Excluding the Irrelevant: Focusing Reinforcement Learning through Continuous Action Masking

Roland Stolz[1,*] **Hanna Krasowski**[1,2,*]

Jakob Thumm[1] **Michael Eichelbeck**[1] **Philipp Gassert**[1,3] **Matthias Althoff**[1]

[1]Technical University of Munich, [2]University of California, Berkeley,
[3]Munich Center for Machine Learning
{roland.stolz, hanna.krasowski}@tum.de

Abstract

Continuous action spaces in reinforcement learning (RL) are commonly defined as multidimensional intervals. While intervals usually reflect the action boundaries for tasks well, they can be challenging for learning because the typically large global action space leads to frequent exploration of irrelevant actions. Yet, little task knowledge can be sufficient to identify significantly smaller state-specific sets of relevant actions. Focusing learning on these relevant actions can significantly improve training efficiency and effectiveness. In this paper, we propose to focus learning on the set of relevant actions and introduce three continuous action masking methods for exactly mapping the action space to the state-dependent set of relevant actions. Thus, our methods ensure that only relevant actions are executed, enhancing the predictability of the RL agent and enabling its use in safety-critical applications. We further derive the implications of the proposed methods on the policy gradient. Using proximal policy optimization (PPO), we evaluate our methods on four control tasks, where the relevant action set is computed based on the system dynamics and a relevant state set. Our experiments show that the three action masking methods achieve higher final rewards and converge faster than the baseline without action masking.

1 Introduction

Reinforcement learning (RL) can solve complex tasks in areas such as robotics [13], games [35], and large language models [26]. Yet, training RL agents is often sample-inefficient due to frequent exploration of actions, which are irrelevant to learning a good policy. Irrelevant actions are actions that are either physically impossible, forbidden due to some formal specification, or evidently counterproductive for solving the task. Since the global action space is typically large in relation to the relevant actions in each state, exploring these actions frequently can introduce unnecessary costs, lead to slow convergence, or even prevent the agent from learning a suitable policy.

Action masking mitigates this problem by constraining the exploration to the set of relevant state-specific actions, which can be obtained based on task knowledge. For example, when there is no opponent within reach in video games, attack actions are masked from the action space [43]. Leveraging task knowledge through action masking usually leads to faster convergence and also improves the predictability of the RL agent, especially when the set of relevant actions has a specific

[*]The first two authors contributed equally to this work.

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

 https://doi.org/10.52202/079017-3013

notion, such as being the set of safe actions. For instance, if the set of relevant actions is a set of verified safe actions, action masking can be used to provide safety guarantees [10, 19].

When the relevant action set is easy to compute, action masking usually benefits RL by improving the sample efficiency and is the quasi-standard for discrete action spaces [34, 46], e.g., in motion planning [8, 18, 23, 30, 42], games [14, 15, 43], and power systems [21, 38]. However, real-world systems operate in continuous space and discretizing it might prevent learning optimal policies. Furthermore, simulation of real-world systems is computationally expensive, and developing RL agents for them often requires additional real-world training [45]. Thus, sample efficiency is particularly valuable for these applications.

In this work, we propose three action masking methods for continuous action spaces. They can employ convex set representations, e.g., polytopes or zonotopes, for the relevant action set. Our action masking methods generalize previous work in [19], which is constrained to intervals as relevant action sets. This extends the applicability of continuous action masking to expressive convex relevant action sets, which is especially useful when action dimensions are coupled, e.g., for a thrust-controlled quadrotor. To integrate the relevant action set into RL, we introduce three methods: the *generator mask*, which exploits the generator representation of a zonotope, the *ray mask*, which projects an action into the relevant action set based on radial directions, and the *distributional mask*, which truncates the policy distribution to the relevant action set. In summary, our main contributions are:

- We introduce continuous action masking based on convex sets representing the state-dependent relevant action sets;
- We present three methods to utilize the relevant action sets and derive their integration in the backward pass of RL with stochastic policies;
- We evaluate our approach on four benchmark environments that demonstrate the applicability of our continuous action masking approaches.

2 Related literature

Action masking has been mainly applied to discrete action spaces [8, 10, 14, 15, 19, 18, 21, 23, 30, 38, 41, 42, 43, 46]. Huang et al. [15] derive implications on policy gradient RL and evaluate their theoretical findings on real-time strategy games. They show that masking actions leads to higher training efficiency and scales better with an increasing number of actions than penalizing the agent for selecting irrelevant actions. Huo et al. [14] have extended [15] to off-policy RL and have observed similar empirical results.

Action masking for discrete action spaces can be categorized by the purpose of the state-dependent relevant action set. Often, the set is obtained by removing irrelevant actions based on task knowledge [8, 14, 15, 21, 30, 42], e.g., executing a harvesting action before goods are produced [15]. While the relevant action sets are usually manually engineered, a recent study [46] takes a data-driven approach and identifies redundant actions based on similarity metrics. Another common interpretation of the relevant action set is that it only includes safe actions [10, 18, 23, 38, 41]. These works typically use the system dynamics to verify the safety of actions. A safe action avoids defined unsafe areas or complies with logic formulas. In our experiments, the relevant action sets are either state-dependent safe action sets or a global relevant action set modeling power supply constraints.

For continuous action spaces, there is work on utilizing action masking with multidimensional intervals (hereafter only referred to as intervals) as relevant action sets [19]. In particular, the proposed continuous action masking represents relevant action sets by intervals that reflect safety constraints and employs straightforward re-normalization to map the action space to the relevant action set. In this paper, we generalize to more expressive relevant action sets and demonstrate the applicability to four benchmark environments.

3 Preliminaries

As the basis for our derivations of the three masking methods, we provide a concise overview of RL with policy gradients. Further, we define the considered system dynamics as well as the set representations used in this work.

3.1 Reinforcement learning with policy gradients

A Markov decision process is a tuple $(\mathcal{S}, \mathcal{A}, T, r, \gamma)$ consisting of the following elements: the observable and continuous state set $\mathcal{S} \subset \mathbb{R}^{n^S}$, the action set $\mathcal{A} \subset \mathbb{R}^N$, the state-transition distribution $T(s'|a, s)$, which is stationary and describes the transition probability to the next state $s' \in \mathcal{S} \subset \mathbb{R}^{n^S}$ from the current state $s \in \mathcal{S}$ when executing action $a \in \mathcal{A}$, the reward $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, and the discount factor γ for future rewards [37]. The goal of RL is to learn a parameterized policy $\pi_\theta(a|s)$ that maximizes the expected reward $\max_\theta \mathbb{E}_{\pi_\theta} \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)$.

For policy gradient algorithms, learning the optimal policy $\pi_\theta^*(a|s)$ is achieved by updating its parameters θ using the policy gradient [36, Thm. 2]

$$\nabla J(\pi_\theta) = \mathbb{E}_{\pi_\theta} \left[\nabla_\theta \log \pi_\theta(a|s) A_{\pi_\theta}(a, s) \right], \tag{1}$$

where $A_{\pi_\theta}(a, s)$ is the advantage function, which represents the expected improvement in reward by taking action a in state s compared to the average action taken in that state according to the policy $\pi_\theta(a|s)$. An estimation of the advantage $A_{\pi_\theta}(a, s)$ is usually provided by a neural network.

3.2 System model and set representations

We consider general continuous-time systems of the form

$$\dot{s} = f(s, a, w), \tag{2}$$

where $w \in \mathcal{W} \subset \mathbb{R}^{n^\mathcal{W}}$ denotes a disturbance. The input is piece-wise constant with a sampling interval Δt. We assume $\mathcal{S}$, $\mathcal{A}$, and $\mathcal{W}$ to be convex.

Zonotopes are a convex set representation well-suited for representing the relevant action set, due to the efficiency of computing their Minkowski sums and linear maps. A zonotope $\mathcal{Z} \subset \mathbb{R}^N$ with center $c \in \mathbb{R}^N$, generator matrix $G \in \mathbb{R}^{N \times P}$, and scaling factors $\beta \in \mathbb{R}^P$ is defined as

$$\mathcal{Z} = \left\{ c + G\beta \,\middle|\, \|\beta\|_\infty \leq 1 \right\} = \langle c, G \rangle_\mathcal{Z}. \tag{3}$$

Additionally, let us denote that $G_{(\cdot,i)}$ returns the i-th column vector of the generator matrix G. The Minkowski addition $\mathcal{Z}_1 \oplus \mathcal{Z}_2$ of two zonotopes $\mathcal{Z}_1$, $\mathcal{Z}_2$ and the linear map $M\mathcal{Z}_1$ of a zonotope $\mathcal{Z}_1$ are given by [2, Eq. 2.1]

$$\mathcal{Z}_1 \oplus \mathcal{Z}_2 = \langle c_1 + c_2, [G_1 \quad G_2] \rangle_\mathcal{Z}, \tag{4a}$$

$$M\mathcal{Z}_1 = \langle Mc_1, MG_1 \rangle_\mathcal{Z}. \tag{4b}$$

4 Continuous action masking

To apply action masking, a relevant action set $\mathcal{A}^r(s) \subseteq \mathcal{A}$ has to be available, which constrains the action space $\mathcal{A}$ based on task knowledge. Let us denote the state-dependent relevant action set as $\mathcal{A}^r(s) \subseteq \mathcal{A}$. From now on, we omit the dependency on the state to simplify notation. For our continuous action masking methods, we specifically require the following two assumptions:

Assumption 1. *The relevant action set $\mathcal{A}^r$ is convex and its center and boundary points are computable.*

Assumption 2. *The policy $\pi_\theta : S \times \mathcal{A} \to \mathbb{R}_+$ of the agent is represented by a parameterized probability distribution $a \sim \pi_\theta(a|s)$.*

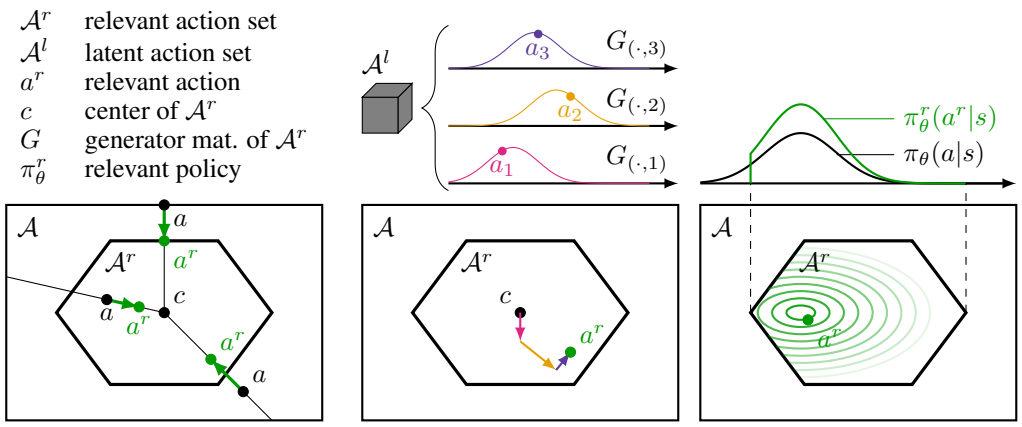
Common convex set representations that fulfill Assumption 1 are polytopes or zonotopes. Our continuous action masking methods transform the policy $\pi_\theta(a|s)$, for which the parameters θ usually specify a neural network, into the relevant policy $\pi_\theta^r : S \times \mathcal{A}^r \to \mathbb{R}_+$ through a functional $h : (\Pi \times \mathcal{P}(\mathcal{A})) \to \Pi^r$. Here, Π is the space of all policies, Π^r is the space of all relevant policies, and $\mathcal{P}(\mathcal{A})$ is the power set of all $\mathcal{A}^r$. The transformation is defined as

$$a^r \sim \pi_\theta^r(a^r|s) = h\big(\pi_\theta(a|s), \mathcal{A}^r\big), \tag{5}$$

and thereby ensures that $a^r \in \mathcal{A}^r$ always holds. Please note that policy gradient methods only require the ability to sample from and to compute the gradient of the policy distribution. Therefore, explicit closed-form expressions for $\pi_\theta^r(a^r|s)$ and h are not necessary.

In the following subsections, we introduce and evaluate three masking methods: generator mask, ray mask, and distributional mask, as shown in Fig. 1. We derive the effect of each masking approach on the gradient of the objective function for stochastic policy gradient methods.

https://doi.org/10.52202/079017-3013



(a) Ray mask (b) Generator mask (c) Distributional mask

Figure 1: Illustration of masking methods in action space $\mathcal{A}$ with a hexagon-shaped relevant action set $\mathcal{A}^r$. The ray mask radially maps the actions towards the center of the relevant action set. The generator mask employs the latent action space $\mathcal{A}^l$, which is the generator space of the zonotope modeling the relevant action set. The distributional mask augments the policy probability density function so that it is zero outside the relevant action set.

4.1 Ray mask

The ray mask maps the action set $\mathcal{A}$ to $\mathcal{A}^r$ by scaling each action a alongside a ray from the center of $\mathcal{A}^r$ to the boundary of $\mathcal{A}$, as shown in Figure 1a. Specifically, the relevant policy π^r_θ results from mapping the action a, sampled from $\pi_\theta(a|s)$, using the function $g_\mathrm{R} : \mathcal{A} \to \mathcal{A}^r$:

$$a^r = g_\mathrm{R}(a) = c + \frac{\lambda_{\mathcal{A}^r}(a)}{\lambda_{\mathcal{A}}(a)}(a - c). \tag{6}$$

Here, $\lambda_{\mathcal{A}}(a)$ and $\lambda_{\mathcal{A}^r}(a)$ denote the distances of the action a to the boundaries of the relevant action set and action space, respectively, measured from the center of the relevant action set c in the direction of a. Note that computing $\lambda_{\mathcal{A}^r}(a)$ for a zonotope requires solving a convex optimization problem, as specified in Appendix A.4. Yet, the ray mask is applicable for all convex sets, for which we can compute the center and boundary points. Since $g_\mathrm{R}(a)$ is bijective (see Appendix A.1 for a detailed proof), we can apply a change of variables [5, Eq. 1.27] to compute the relevant policy

$$\pi^r_\theta(a^r|s) = \pi_\theta\left(g_\mathrm{R}^{-1}(a^r)|s\right)\left|\det\left(\frac{\mathrm{d}}{\mathrm{d}a^r} g_\mathrm{R}^{-1}(a^r)\right)\right|, \tag{7}$$

where $g_\mathrm{R}^{-1}(a^r) = a$ is the inverse of g_R. In general, there is no closed form of the distribution π^r_θ available. However, for stochastic policy gradient-based RL, we only require to sample from π^r_θ and compute its policy gradient. Samples from $\pi^r_\theta(a^r|s)$ are created by sampling from the original policy $a \sim \pi_\theta(a|s)$ followed by computing $a^r = g_\mathrm{R}(a)$. The policy gradient is derived next.

Proposition 1. *Policy gradient for the ray mask.*

The policy gradient of $\pi^r_\theta(a^r|s)$ for the ray mask is

$$\nabla_\theta J\left(\pi^r_\theta(a^r|s)\right) = \mathbb{E}_{\pi^r_\theta}\left[\nabla_\theta \log \pi_\theta(a|s) A_{\pi^r_\theta}(a^r, s)\right], \tag{8}$$

where $A_{\pi^r_\theta}(a^r, s)$ is the advantage function associated with $\pi^r_\theta(a^r|s)$.

Proof. The determinant in (7) is independent of θ, i.e.,

$$\nabla_\theta \det\left(\frac{\mathrm{d}}{\mathrm{d}a^r} g_\mathrm{R}^{-1}(a^r)\right) = 0, \tag{9}$$

which simplifies the score function of $\pi^r_\theta(a^r|s)$ to

$$\nabla_\theta \log \pi^r_\theta(a^r|s) = \nabla_\theta \log \pi_\theta(a|s). \tag{10}$$

Combining (10) and the general form of the policy gradient in (1) for $\pi^r_\theta(a^r|s)$ results in (8).

□

https://doi.org/10.52202/079017-3013

4.2 Generator mask

Zonotopes can be interpreted as the map of a hypercube in the generator space to a lower-dimensional space (see Section 3.2). The generator mask is based on exploiting this interpretation by letting the RL agent select actions in the hypercube of the generator space. Since the size of the output layer of the policy network cannot change during the learning process, we fix the dimension of the generator space. The generator mask requires the following assumption:

Assumption 3. *The relevant action set $\mathcal{A}^r(s)$ is represented by a zonotope $\langle c(s), G(s)\rangle_{\mathcal{Z}}$, with $G \in \mathbb{R}^{N\times P}$ and $c \in \mathbb{R}^N$, and a state-invariant number of generators P.*

Note that in practice, Assumption 3 can often be trivially fulfilled by choosing sufficiently many generators, and the number of generators P is usually the output dimension of the parametrized policy. The domain of the policy $\pi_\theta(a|s)$ is the hypercube $\mathcal{A}^l = [-1, 1]^P$, which can be interpreted as a latent action space, and the domain of the relevant policy $\pi^r_\theta(a^r|s)$ is a subset of the action space $\mathcal{A}^r \subseteq \mathcal{A}$ (see Figure 1b).

To derive the policy gradient of the generator mask, we assume:

Assumption 4. *$\pi_\theta(a|s)$ is a parametrized normal distribution $\mathcal{N}(a; \mu_\theta, \Sigma_\theta)$.*

Proposition 2. *The relevant policy of the generator mask is*

$$\pi^r_\theta(a|s) = \mathcal{N}(a; G\mu_\theta + c, G\Sigma_\theta G^T). \tag{11}$$

Proof. The generator mask $g_{\mathrm{G}} : \mathcal{A}^l \to \mathcal{A}^r$ is:

$$a^r = g_{\mathrm{G}}(a) = c + Ga, \tag{12}$$

which is a linear function. Therefore, the proof directly follows from the linear transformation of multivariate normal distributions [Thm. 3.3.3][40]. □

Note that the ray mask and generator mask are mathematically equivalent to the approach in [19] if the relevant action set is constrained to intervals. Since $g_{\mathrm{G}}(a)$ is not bijective in general, we cannot derive the gradient through a change of variables, as for the ray mask.

Proposition 3. *The policy gradient for $\pi^r_\theta(a^r|s)$ as defined in* (11) *with respect to μ_θ and Σ_θ is*

$$\nabla_{\mu_\theta} \log \pi^r_\theta(a^r|s) = G^T(G\Sigma_\theta G^T)^{-1}(a^r - c - G\mu_\theta), \tag{13}$$

$$\begin{aligned}\nabla_{\Sigma_\theta} \log \pi^r_\theta(a^r|s) = -\frac{1}{2}\big(&G^T(G\Sigma_\theta G^T)^{-1}G - G^T(G\Sigma_\theta G^T)^{-1}(a^r - c - G\mu_\theta)\\ &(a^r - c - G\mu_\theta)^T(G\Sigma_\theta G^T)^{-1}G\big).\end{aligned} \tag{14}$$

Proof. The proposition is proven in Appendix A.2. □

Note that for the special case where G^{-1} exists, i.e., G is square and non-singular, the expressions in (13) and (14) simplify to $\nabla_{\mu_\theta} \log \pi_\theta(a|s)$, and $\nabla_{\Sigma_\theta} \log \pi_\theta(a|s)$, respectively, when Assumption 4 holds (see Proposition 5 in Appendix A.2). While the generator matrix will commonly not be square, as usually $P > N$, there are cases where $P = N$ is a valid choice, e.g., if there is a linear dependency between the action dimensions as for the 2D Quadrotor dynamics (see (40)).

4.3 Distributional mask

The intuition behind the distributional mask comes from discrete action masking, where the probability for irrelevant actions is set to 0 [15]. For continuous action spaces, we aim to achieve the same by ensuring that actions are only sampled from the relevant action set $\mathcal{A}^r$, by setting the density values of the relevant policy distribution $\pi^r_\theta(a^r|s)$ to zero for actions outside the relevant action set (see Figure 1c). For the one-dimensional case, this can be expressed by the truncated distribution [7]. In higher dimensions, the resulting policy distribution is

$$\pi^r_\theta(a^r|s) = \frac{\phi(a, s)\pi_\theta(a|s)}{\int_{\mathcal{A}^r} \pi_\theta(\tilde{a}|s)\mathrm{d}\tilde{a}}, \tag{15}$$

https://doi.org/10.52202/079017-3013

where $\phi(a, s)$ is the indicator function

$$\phi(a, s) = \begin{cases} 1 & \text{if } a \in \mathcal{A}^r, \\ 0 & \text{otherwise.} \end{cases} \tag{16}$$

Since there is no closed form of this distribution, we employ Markov chain Monte Carlo sampling to sample actions from the policy. More specifically, we utilize the random direction hit-and-run algorithm: a geometric random walk that allows sampling from a non-negative, integrable function $f : \mathbb{R}^N \to \mathbb{R}_+$, while constraining the samples to a bounded set [44]. The algorithm iteratively chooses a random direction from the current point, computes the one-dimensional, truncated probability density of f along this direction, and samples a new point from this density. The approach is particularly effective for high-dimensional spaces where other sampling methods might struggle with convergence or efficiency. For the distributional mask, f is the policy $\pi_\theta(a|s)$, and $\mathcal{A}^r$ is the set. As suggested by [22], we execute N^3 iterations before accepting the sample. To estimate the integral in (15), we use numerical integration with cubature [11], which is a method to approximate the definite integral of a function $l : \mathbb{R}^N \to \mathbb{R}$ over a multidimensional geometric set.

Proposition 4. *The policy gradient for the distributional mask is*

$$\nabla_\theta \log \pi^r_\theta(a^r|s) = \nabla_\theta \log \pi_\theta(a|s) - \nabla_\theta \log \int_{\mathcal{A}^r} \pi_\theta(\tilde{a}|s) \mathrm{d}\tilde{a}. \tag{17}$$

Proof. Equation (17) is obtained by calculating the gradient of the logarithm of (15). The indicator function $\phi(a, s)$ is not continuous and differentiable, which would necessitate the use of the subgradient for learning. However, since $a^r \in \mathcal{A}^r$ always holds, the gradient has to be computed for the continuous part of $\pi^r_\theta(a^r|s)$ only, and thus $\phi(a, s)$ can be omitted. □

Since the gradient of the numeric integral $\int_{\mathcal{A}^r} \pi_\theta(\tilde{a}|s) \mathrm{d}\tilde{a}$ is intractable for zonotopes, we treat the integral as a constant in practice, and use $\nabla_\theta \log \pi^r_\theta(a^r|s) \approx \nabla_\theta \log \pi_\theta(a|s)$. We discuss potential improvements in Section 5.3.

5 Numerical experiments

We compare the three continuous action masking methods in four different environments: The simple and intuitive Seeker Reach-Avoid environment, the 2D Quadrotor environment to demonstrate the generalization from the continuous action masking approach in [19], and the 3D Quadrotor and Mujoco Walker2D environment to show the applicability to action spaces of higher dimension. Because the derivation of the relevant action set is not trivial in practice, we selected four environments for which we could compute intuitive relevant action sets. In particular, for the Seeker and Quadrotor environments, the relevant action set is a safe action set since it is computed so that the agent does not collide (Seeker) or leave a control invariant set (2D and 3D Quadrotor). For the Walker2D environment, the relevant action set is state-independent and models a power supply constraint for the actuators.

For the experiments, we extend the stable-baseline3 [29] implementation of proximal policy optimization (PPO) [33] by our masking methods. PPO is selected because it is a widely used algorithm in the field of RL and fulfills both Assumptions 2 and 4 by default. Apart from the masking agents, we also train a baseline agent with standard PPO that uses the action space $\mathcal{A}$ and a replacement agent, for which an action outside of $\mathcal{A}^r$ is replaced by a uniformly sampled action from $\mathcal{A}^r$ (see [19] for details). The replacement agent is an appropriate comparison to the masking agents since only relevant actions are executed while the replacement is implemented as part of the environment, which is usually easier than an implementation as part of the policy as for the masking methods. We conduct a hyperparameter optimization with 50 trials for each masking method and environment. The resulting hyperparameters are reported in Appendix A.9. All experiments are run on a machine with a Intel(R) Xeon(R) Platinum 8380 2.30 GHz processor and 2 TB RAM.

5.1 Environments

We briefly introduce the three environments and their corresponding relevant action sets $\mathcal{A}^r$. Parameters and dynamics for the environments are detailed in Appendix A.5.

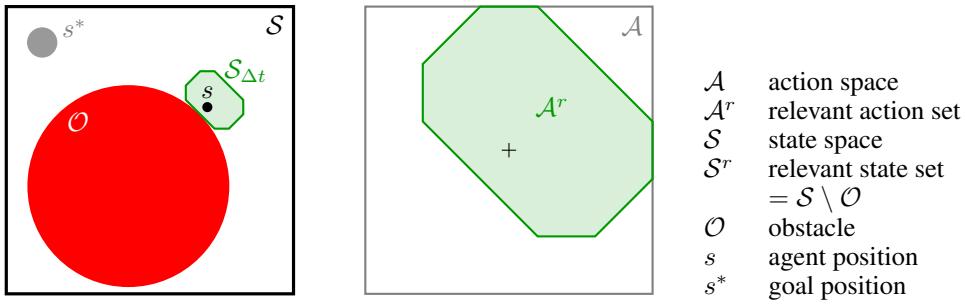


Figure 2: The Seeker Reach-Avoid environment with state and action space. The agent (black) has to reach the goal (gray) while avoiding the obstacle (red). The center of the action space is illustrated by a cross and the relevant action set $\mathcal{A}^r$ for the current state is shown in green. The state set reachable at the next time step, by the relevant action set, is $\mathcal{S}_{\Delta t}$.

5.1.1 Seeker Reach-Avoid

This episodic environment features an agent navigating a 2D space, tasked with reaching a goal while avoiding a circular obstacle (see Figure 2). It is explicitly designed to provide an intuitive relevant action set $\mathcal{A}^r$. The system is defined as in Section 3.2: The dynamics for the position of the agent $s = [s_x, s_y]^T$ and the action $a = [a_x, a_y]^T$ is $\dot{s} = a$, and there are no disturbances.

The environment is characterized by the position of the agent, the goal position s^*, the obstacle position o, and the obstacle radius r_o. These values are pseudo-randomly sampled at the beginning of each episode, with the constraints that the goal cannot be inside the obstacle and the obstacle blocks the direct path between the initial position of the agent and the goal. The reward for each time step is

$$r(a, s) = \begin{cases} 100 & \text{if goal reached,} \\ -100 & \text{if collision occurred,} \\ -1 - \|s^* - s\|_2 & \text{otherwise.} \end{cases} \tag{18}$$

We compute the relevant action set $\mathcal{A}^r$ so that all actions that cause a collision with the obstacle or the boundary are excluded (see Appendix A.3.3).

5.1.2 2D Quadrotor

The 2D Quadrotor environment models a stabilization task and employs an action space where the two action dimensions are coupled, i.e., rotational movement is originating from differences between the action values and vertical movement is proportional to the sum of the action values. The relevant action set is computed based on the system dynamics and a relevant state set (see Appendix, Eq. (33)). The reward function is defined as

$$r(a, s) = \exp\left(-\|s - s^*\|_2 - \frac{0.01}{2}\left\|\left[\frac{a_1 - a_{1,\min}}{a_{1,\text{range}}}, \frac{a_2 - a_{2,\min}}{a_{2,\text{range}}}\right]\right\|_1\right), \tag{19}$$

where $s^* = \mathbf{0}$ is the stabilization goal state, $a = [a_1, a_2]$ is the two-dimensional action, $a_{i,\min}$ is the lower bound for the actions in dimension i, and $a_{i,\text{range}}$ is the absolute difference between the lower and upper bound for the actions in dimension i.

5.1.3 3D Quadrotor

The third environment models a stabilization task for a quadrotor defined in [16]. The quadrotor has four action dimensions. We use the same reward (see (19)) and the same calculation approach for the relevant action set (see Appendix, Eq. (33)) as for the 2D Quadrotor.

5.1.4 Mujoco Walker2D

The relevant action sets of the two Quadrotor and the Seeker environments are sets that only contain safe actions for the current state. Computing safe action sets requires considerable domain knowledge

https://doi.org/10.52202/079017-3013

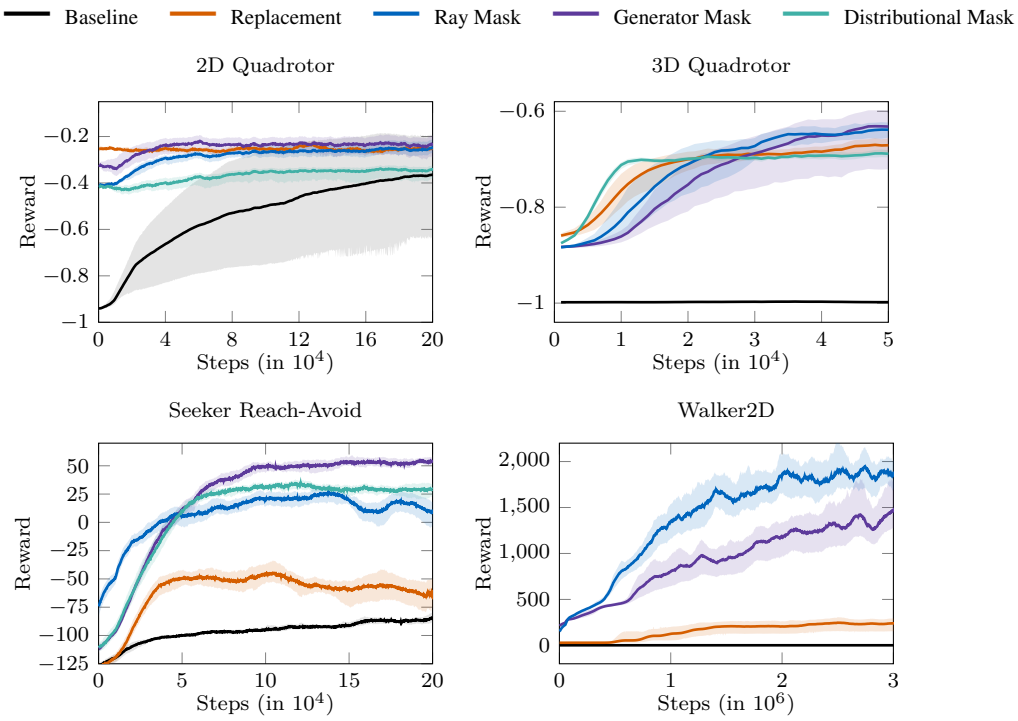


Figure 3: Average reward curves for benchmarks with transparent bootstrapped 95% confidence interval.

and for our case solving an optimization problems (see Appendix A.3). However, action masking is not restricted to safe relevant action sets, which we aim to demonstrate on the Mujoco Walker2D environment [39]. We extend the environment with a termination criterion, which ends an episode, when the the action violates the constraint $\|a\|_2 \leq \alpha_p$. This can be viewed as constraining the cumulative power output on all joints to a maximum value α_p. We motivate this constraint by having a battery with a maximum power output that should not be exceeded in practice. Accordingly, we define the relevant action set as the static set

$$\mathcal{A}^r = \{a \big| \|a\|_2 \leq \alpha_p\}. \tag{20}$$

We under-approximate this relevant action set with a zonotope that consists of 36 generators.

5.2 Results

The reported training results are based on ten random seeds for each configuration. Figure 3 shows the mean reward and the bootstrapped 95% confidence intervals [27] for the four environments and Table 1 depicts the mean and standard deviation of the episode return during deployment. First, we present the results for the Seeker and Quadrotor environments for which the relevant action set is state-dependent and only includes safe actions. Then, we detail the results for the Walker2D environment with a static relevant action set.

For the Seeker and Quadotor environments, the baseline, i.e., PPO, converges significantly slower or not at all (see Figure 3). Additionally, the initial rewards when using action masking are significantly higher than for the baseline, indicating that the exploration is indeed constrained to relevant task-fulfilling actions. During deployment (see Table 1), the baseline episode returns are significantly lower than for action masking, while the three masking methods perform similarly. More specifically, for the Seeker environment, the relative volume of the relevant action set compared to the global action space is on average 70%, and all action masking methods converge significantly faster to high rewards than the baseline. The generator mask and the distributional mask achieve the highest final reward. Further, action replacement performs better than the baseline but significantly worse than

Table 1: Mean and standard deviation of episode return for ten runs per trained model.

	Seeker	2D Quad.	3D Quad.	Walker2D
Baseline	-71.03 ± 19.67	-0.80 ± 0.15	-1.00 ± 0.00	1.19 ± 0.06
Replacement	-60.21 ± 19.98	-0.25 ± 0.03	-0.68 ± 0.09	234.93 ± 129.5
Ray	-20.45 ± 16.39	-0.26 ± 0.05	-0.63 ± 0.03	1941.82 ± 992.83
Generator	18.60 ± 22.02	-0.25 ± 0.02	-0.68 ± 0.07	1443.62 ± 702.7
Distributional	-13.66 ± 19.97	-0.23 ± 0.02	-0.66 ± 0.02	–

masking. For the 2D Quadrotor, the relative volume is on average 28%, which is much smaller than for the Seeker environment. In the 2D Quadrotor environment, the ray mask, generator mask, and action replacement achieve the highest reward. While the baseline converges significantly slower, the final average reward is similar to the one of the distributional mask. Please note that the confidence interval for the baseline is significantly larger, because three of the ten runs do not converge and, on average, exhibit a reward of one throughout training. Additionally, we observed that if we constrain the relevant action set to intervals for this environment, our optimization problem in (33) often renders infeasible, because the maximal relevant action set cannot be well approximated by an interval. Thus, the masking method proposed in [19] is not suitable for the 2D Quadrotor task. The results on the 3D Quadrotor are similar to those on the 2D Quadrotor; again, the generator mask converges the fastest but yields a final reward similar to that of the ray mask and action replacement. The relative volume of the relevant action set to the global action space is on average 25%. Notably, in this environment, the baseline does not learn a meaningful policy. Based on these three environments with state-dependent relevant action sets, it seems that action masking performs better than action replacement when the relative volume is not too small.

The training results of the Walker2D experiment are shown in Figure 3. While the generator and ray mask both learn a performant policy, the ray mask outperforms by a significant margin. The lower performance of the generator mask is likely due to the high-dimensional generator space with 36 dimensions. This is supported by initial experiments with 12 generators (i.e., a more conservative under-approximation of the L_2-norm) where the generator mask performed better compared to the ray mask. Replacement performs significantly worse than the two masking approaches, and the PPO baseline does not learn a meaningful policy since the environment is frequently reset due to violations of the power constraint in (20). The frequent terminations occur since in six action dimensions the relative volume of the unit ball compared to the unit box is $\approx 8\%$, i.e., more than 92% of actions are violating the power constraint. To compare masking to a learning baseline, we also evaluated standard PPO without constraints, which performs slightly better than ray masking but almost always uses actions outside $\mathcal{A}^r$ (see Appendix A.7). The deployment results in Table 1 reflect similar results as the training; the ray mask achieves better rewards than the generator mask, followed by replacement, and the baseline performs the worst. We excluded the distributional mask for the Walker2D, since its computation time is approximately 170 times slower than the baseline, compared to a 1.6 increase for the generator, 2.7 for the ray mask, and 2.5 for action replacement (see Table 3). The severely increased computational cost for the distributional mask arises from the geometric random walks, which scale cubically with action dimensions.

5.3 Discussion and limitations

Our experimental results indicate that continuous action masking with zonotopes can improve both the sample efficiency and the final policy of PPO. While the sample efficiency is higher in our experiments, computing the relevant action set adds computational load as shown by the increased computation times (see Appendix A.6). Thus, in practice, a tight relevant action set might require more computation time than the additional samples for standard RL algorithms. Yet, if the relevant action set provides guarantees, e.g., is a set of verified safe actions, this increased computation time is often acceptable. Additionally, the computational effort for the masks differs. Given a relevant action zonotope, the generator mask adds a matrix-vector multiplication, which scales quadratically with action dimensions, the ray mask is dominated by the computation of the boundary points, which scales polynomially with action dimensions [20], and the distributional mask scales cubically with the dimension of the action space due to the mixing time of the hit-and-run algorithm [22]. Note that

 https://doi.org/10.52202/079017-3013

for the Walker2D environment with six action dimensions, the computation time for the hit-and-run algorithm is so high that the distributional mask evaluation was infeasible.

The ray mask and generator mask are based on functions g_{R} and g_{G} that map to relevant actions. There are two different approaches of incorporating these functions into RL algorithms. One is to apply the mapping as part of the environment on an action that is sampled by a standard RL policy. The second option, which we use, is to consider the masking as part of the policy, which creates the relevant policy as in (5). This has three main benefits over integrating masking as part of the environment. First, the actions passed to the environment are better interpretable, e.g., for the generator mask, adding the masking mapping function to the environment leads to a policy that samples actions from the generator space, which commonly does not have an intuitive interpretation. Second, more formulations of the functional h are possible, e.g., the distributional mask, and, third, the mapping function can be included in the gradient calculation. For mathematically sound masking as part of the policy, the correct gradient needs to be derived for each RL algorithm, and the standard RL algorithms need to be adapted accordingly. However, the empirical benefit could be minor. Thus, future work should investigate the significance of the correct gradient on a variety of tasks. Note that we showed in Proposition 1 that for the ray mask, the PPO gradient with respect to the original policy and relevant policy is the same. Thus, for the ray mask, it does not matter if it is viewed as part of the policy or the environment.

PPO is a common RL algorithm. However, off-policy algorithms such as Twin Delayed DDPG (TD3) [9] and Soft Actor-Critic (SAC) [12] are frequently employed as well. The ray mask and generator mask are conceptually applicable for deterministic policies as used in TD3. Yet, the implications on the gradient must be derived for each RL algorithm and are subject to future work. For the distributional mask, treating the integral in (15) as constant with respect to θ is a substantial simplification, which might be an explanation for the slightly worse convergence of the distributional mask since this introduces an off-policy bias. To address this in future work, one could approximate the integral with a neural network, which has the advantage that it is easily differentiable.

We focus this work on the integration of a convex relevant action set into RL and assume that an appropriate relevant action set can be obtained. While convex sets are a significant generalization from previous work [19], they might not be sufficient for some applications, e.g., tasks where relevant action sets are disjoint. Thus, future work could include investigating hybrid RL approaches [25] to increase the applicability to multiple convex sets or non-convex sets, such as constrained polynomial zonotopes [17]. Further, obtaining the relevant action set can be a major challenge in practice, in particular when the relevant action set is different for each state. Such a high state-dependency is likely when the notion of relevance is safety, while for other definitions of action relevance, the relevant action set might be easy to pre-compute, e.g., excluding high steering angles at high velocities. Additionally, there might be an optimal precision of the relevant action set due to two opposing mechanisms. On the one hand, the larger the relevant action set is with respect to the action space, the smaller the sample efficiency gain from action masking might get. On the other hand, a tight relevant action set might require significant computation time to obtain. Thus, future work should investigate efficient methods to obtain sufficiently tight relevant action sets.

6 Conclusion

We propose action masking methods for continuous action spaces that focus the exploration on the relevant part of the action set. In particular, we extend previous work on continuous action masking from using intervals as relevant action sets to using convex sets. To this end, we have introduced three masking methods and have derived their implications on the gradient of PPO. We empirically evaluated our methods on four benchmarks and observed that the generator mask and ray mask perform best. If the relevant action set can be described by a zonotope with fixed generator dimensions and the policy follows a normal distribution, the generator mask is straightforward to implement. If the assumptions for the generator mask cannot be fulfilled, the ray mask is recommended based on our experiments. Because of subpar performance and longer computation time, the distributional mask needs to be further improved. Future work should also investigate a broad range of benchmarks to identify the applicability and limits of continuous action masking with convex sets more clearly.

Acknowledgments and Disclosure of Funding

We thank Matthias Killer for conducting preliminary experiments. We gratefully acknowledge that this project was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – SFB 1608 – 501798263, AL 1185/9-1, AL 1185/33-1, and the Bavarian Research Foundation project STROM (Energy - Sector coupling and microgrids).

References

[1] Takuya Akiba et al. "Optuna: A Next-Generation Hyperparameter Optimization Framework". In: *25th ACM SIGKDD Int. Conf. on Knowledge Discovery & Data Mining*. 2019, pp. 2623–2631.

[2] Matthias Althoff. "Reachability Analysis and its Application to the Safety Assessment of Autonomous Cars". PhD thesis. Technische Universität München, 2010.

[3] Matthias Althoff and Goran Frehse. "Combining zonotopes and support functions for efficient reachability analysis of linear systems". In: *IEEE Conf. on Decision and Control (CDC)*. 2016, pp. 7439–7446.

[4] Dimitris Bertsimas and John N. Tsitsiklis. *Introduction to linear optimization*. Athena Scientific, 1997.

[5] Christopher M. Bishop. *Pattern recognition and machine learning*. Information science and statistics. 2006.

[6] Stephen Boyd et al. "A tutorial on geometric programming". In: *Optimization and engineering* 8 (2007), pp. 67–127.

[7] J. Burkardt. *The Truncated Normal Distribution*. Department of Scientific Computing Website, Florida State University, Tallahassee. Online resource. 2018.

[8] Xutao Feng et al. "Autonomous Decision Making with Reinforcement Learning in Multi-UAV Air Combat". In: *IEEE Int. Conf. on Systems, Man, and Cybernetics (SMC)*. 2023, pp. 2874–2879.

[9] Scott Fujimoto, Herke Van Hoof, and David Meger. "Addressing Function Approximation Error in Actor-Critic Methods". In: *Proc. of the Int. Conf. on Machine Learning (ICML)*. 2018, pp. 2587–2601.

[10] Nathan Fulton and André Platzer. "Safe Reinforcement Learning via Formal Methods: Toward Safe Control Through Proof and Learning". In: *Proc. of the AAAI Conf. on Artificial Intelligence (AAAI)*. 2018, pp. 6485–6492.

[11] Alan Genz and Ronald Cools. "An adaptive numerical cubature algorithm for simplices". In: *ACM Trans. Math. Softw.* 29.3 (2003), pp. 297–308.

[12] Tuomas Haarnoja et al. "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor". In: *Proc. of the Int. Conf. on Machine Learning (ICML)*. 2018, pp. 1861–1870.

[13] Dong Han et al. "A Survey on Deep Reinforcement Learning Algorithms for Robotic Manipulation". In: *Sensors* 23.7 (2023).

[14] Yueqi Hou et al. "Exploring the Use of Invalid Action Masking in Reinforcement Learning: A Comparative Study of On-Policy and Off-Policy Algorithms in Real-Time Strategy Games". In: *Applied Sciences* 13.14 (2023).

[15] Shengyi Huang and Santiago Ontañón. "A Closer Look at Invalid Action Masking in Policy Gradient Algorithms". In: *Int. Florida Artificial Intelligence Research Society Conf. Proc. (FLAIRS)* 35 (2022).

[16] Shahab Kaynama et al. "Scalable Safety-Preserving Robust Control Synthesis for Continuous-Time Linear Systems". In: *IEEE Trans. on Automatic Control* 60.11 (2015), pp. 3065–3070.

[17] Niklas Kochdumper and Matthias Althoff. "Constrained polynomial zonotopes". In: *Acta Informatica* 60.3 (2023), pp. 279–316.

[18] Hanna Krasowski, Xiao Wang, and Matthias Althoff. "Safe Reinforcement Learning for Autonomous Lane Changing Using Set-Based Prediction". In: *Proc. of the IEEE Int. Intelligent Transportation Systems Conf. (ITSC)*. 2020, pp. 1–7.

[19] Hanna Krasowski et al. "Provably Safe Reinforcement Learning: Conceptual Analysis, Survey, and Benchmarking". In: *Trans. on Machine Learning Research* (2023).

 https://doi.org/10.52202/079017-3013

[20] Adrian Kulmburg and Matthias Althoff. “On the co-NP-Completeness of the Zonotope Containment Problem”. In: *European Journal of Control* 62 (2021), pp. 84–91.

[21] Lingyu Liang et al. “Enhancement of Distribution Network Resilience: A Multi-Buffer Invalid-Action-Mask Double Q-Network Approach for Distribution Network Restoration”. In: *3rd Int. Conf. on New Energy and Power Engineering (ICNEPE)*. 2023, pp. 1055–1060.

[22] László Lovász and Santosh Vempala. “Hit-and-Run from a Corner”. In: *SIAM Journal on Computing* 35.4 (2006), pp. 985–1005.

[23] Branka Mirchevska et al. “High-level Decision Making for Safe and Reasonable Autonomous Lane Changing using Reinforcement Learning”. In: *Proc. of the IEEE Int. Intelligent Transportation Systems Conf. (ITSC)*. 2018, pp. 2156–2162.

[24] Ian M. Mitchell, Jacob Budzis, and Andriy Bolyachevets. *Invariant, Viability and Discriminating Kernel Under-Approximation via Zonotope Scaling*. 2019. arXiv: 1901.01006.

[25] Michael Neunert et al. “Continuous-discrete reinforcement learning for hybrid control in robotics”. In: *Proc. of the Conference on Robot Learning*. 2020, pp. 735–751.

[26] Long Ouyang et al. “Training language models to follow instructions with human feedback”. In: *Advances in Neural Information Processing Systems*. Ed. by S. Koyejo et al. Vol. 35. Curran Associates, Inc., 2022, pp. 27730–27744.

[27] Andrew Patterson et al. *Empirical Design in Reinforcement Learning*. 2023. arXiv: 2304.01315.

[28] André Platzer and Edmund M. Clarke. “The Image Computation Problem in Hybrid Systems Model Checking”. In: *Hybrid Systems: Computation and Control*. 2007.

[29] Antonin Raffin et al. “Stable-Baselines3: Reliable Reinforcement Learning Implementations”. In: *Journal of Machine Learning Research* 22.268 (2021), pp. 1–8.

[30] Thomas Rudolf et al. “Fuzzy Action-Masked Reinforcement Learning Behavior Planning for Highly Automated Driving”. In: *Int. Conf. on Control, Automation and Robotics (ICCAR)*. 2022, pp. 264–270.

[31] Sadra Sadraddini and Russ Tedrake. “Linear encodings for polytope containment problems”. In: *IEEE Conf. on Decision and Control (CDC)*. 2019, pp. 4367–4372.

[32] Lukas Schäfer, Felix Gruber, and Matthias Althoff. “Scalable Computation of Robust Control Invariant Sets of Nonlinear Systems”. In: *IEEE Trans. on Automatic Control* 69.2 (2024), pp. 755–770.

[33] John Schulman et al. *Proximal Policy Optimization Algorithms*. 2017. arXiv: 1707.06347.

[34] Ashish Kumar Shakya, Gopinatha Pillai, and Sohom Chakrabarty. “Reinforcement learning algorithms: A brief survey”. In: *Expert Systems with Applications* 231 (2023).

[35] Kun Shao et al. *A Survey of Deep Reinforcement Learning in Video Games*. 2019. arXiv: 1912.10944.

[36] Richard S Sutton et al. “Policy Gradient Methods for Reinforcement Learning with Function Approximation”. In: *Advances in Neural Information Processing Systems*. Ed. by S. Solla, T. Leen, and K. Müller. Vol. 12. MIT Press, 1999.

[37] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. 2nd ed. MIT Press, 2018.

[38] Daniel Tabas and Baosen Zhang. “Computationally Efficient Safe Reinforcement Learning for Power Systems”. In: *Proc. of the American Control Conf. (ACC)*. 2022, pp. 3303–3310.

[39] Emanuel Todorov, Tom Erez, and Yuval Tassa. “MuJoCo: A physics engine for model-based control”. In: *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2012, pp. 5026–5033.

[40] Y. L. Tong. *The Multivariate Normal Distribution*. Springer Series in Statistics. 1990.

[41] G Varricchione et al. “Pure-past action masking”. In: *Proc. of the AAAI Conf. on Artificial Intelligence (AAAI)*. Vol. 38. 19. 2024, pp. 21646–21655.

[42] Yang Xiaofei et al. “Global path planning algorithm based on double DQN for multi-tasks amphibious unmanned surface vehicle”. In: *Ocean Engineering* 266 (2022).

[43] Deheng Ye et al. “Mastering Complex Control in MOBA Games with Deep Reinforcement Learning”. In: *Proc. of the AAAI Conf. on Artificial Intelligence (AAAI)* 34.04 (2020), pp. 6672–6679.

[44] Zelda B. Zabinsky and Robert L. Smith. “Hit-and-Run Methods”. In: *Encyclopedia of Operations Research and Management Science*. Ed. by Saul I. Gass and Michael C. Fu. Boston, MA: Springer US, 2013, pp. 721–729.

[45] Wenshuai Zhao, Jorge Peña Queralta, and Tomi Westerlund. “Sim-to-Real Transfer in Deep Reinforcement Learning for Robotics: a Survey”. In: *IEEE Symposium Series on Computational Intelligence (SSCI)*. 2020, pp. 737–744.

[46] Dianyu Zhong, Yiqin Yang, and Qianchuan Zhao. “No Prior Mask: Eliminate Redundant Action for Deep Reinforcement Learning”. In: *Proc. of the AAAI Conf. on Artificial Intelligence (AAAI)*. Vol. 38. 15. 2024, pp. 17078–17086.

 https://doi.org/10.52202/079017-3013

A Appendix

A.1 Proof of bijectivity of the ray mapping function.

Lemma 1. *The mapping function $g : \mathcal{A} \to \mathcal{A}^r$ of the ray mask $g(a) = c + \frac{\lambda_{\mathcal{A}^r}(a)}{\lambda_{\mathcal{A}}(a)}(a - c)$ is bijective.*

Proof. We show that $g(a)$ is bijective by showing that the function is both injective and surjective.

For any convex set $\mathcal{A}^r$ with center c, we can construct a ray from c through any $a^r \in \mathcal{A}^r$ as $r_\gamma(t) = c + \gamma t$, where $\gamma = \frac{a-c}{\|a-c\|_2}$, and $t \in [0, t_{max}]$. Since $\mathcal{A}^r \subseteq \mathcal{A}$, this also holds $\forall a \in \mathcal{A}$. By construction, any two distinct rays only intersect in c. For all points on a given ray, the scaling factors $\lambda_{\mathcal{A}}$ and $\lambda_{\mathcal{A}^r}$ are constants, allowing us to rewrite $g(a)$ for all $a = r_\gamma(t)$ as the linear function:

$$g\left(r_\gamma(t)\right) = c + \frac{\lambda_{\mathcal{A}^r}}{\lambda_{\mathcal{A}}}\left(r_\gamma(t) - c\right).$$

To show that $g(a)$ is injective, i.e. $\forall a_1, a_2 \in \mathcal{A}$, if $a_1 \neq a_2 \implies g(a_1) \neq g(a_2)$, consider the following two cases for $a_1 \neq a_2$.

Case 1: If a_1 and a_2 are on the same ray, then, $a_1 = r_\gamma(t_1)$ and $a_2 = r_\gamma(t_2)$ with $t_1 \neq t_2$. Since $g\left(r_\gamma(t)\right)$ is linear and thereby monotonic, it follows that $g\left(r_\gamma(t_1)\right) \neq g\left(r_\gamma(t_2)\right)$ and consequently $g(a_1) \neq g(a_2)$.

Case 2: Otherwise, a_1 and a_2 are on different rays and $a_1 \neq a_2 \neq c$, so $g(a_1) \neq g(a_2)$ follows directly as the rays only intersect in c.

Thus, $g(a)$ is injective.

For showing $g(a)$ to be surjective, it is sufficient to show that $\forall a^r \in \mathcal{A}^r$, there exists an $a \in \mathcal{A}$, for which $g(a) = a^r$.

Consider the ray $r_\gamma(t)$ passing through a^r. We know that $g\left(r_\gamma(0)\right) = g(c) = c$, and $g\left(r_\gamma(t_{max})\right)$ is the boundary point of $\mathcal{A}^r$ from c in the direction d. Moreover, a^r lies on the line segment between c and $g\left(r_\gamma(t_{\max})\right)$. Since $g\left(r_\gamma(t)\right)$ is linear and continuous, according to the intermediate value theorem, $\exists t^* \in [0, t_{\max}]$, for which $g\left(r_\gamma(t^*)\right) = a^r$ for any $a^r \in \mathcal{A}^r$ along the ray $r_\gamma(t)$. As we can construct such a ray through any point in $\mathcal{A}^r$, we have shown that for every $a^r \in \mathcal{A}^r$, there exists an $a \in \mathcal{A}$ such that $g(a) = a^r$, thus proving surjectivity. □

A.2 Policy gradient for the generator mask

The log probability density function of the relevant policy for the generator mask is

$$\log \pi^r_\theta(a^r|s) = -\frac{d}{2}\log(2\pi) - \frac{1}{2}\log|G\Sigma_\theta G^T| - \frac{1}{2}(a^r - c - G\mu_\theta)^T(G\Sigma_\theta G^T)^{-1}(a^r - c - G\mu_\theta). \quad (21)$$

We can derive the gradient w.r.t μ_θ as

$$\begin{aligned}\nabla_{\mu_\theta}\log\pi^r_\theta(a^r|s) &= -\frac{1}{2}\nabla_{\mu_\theta}(a^r - c - G\mu_\theta)^T(G\Sigma_\theta G^T)^{-1}(a^r - c - G\mu_\theta)\\ &= -\frac{1}{2}\left(-2(G\Sigma_\theta G^T)^{-1}(a^r - c - G\mu_\theta)\right)\nabla_{\mu_\theta}(a^r - c - G\mu_\theta)\\ &= G^T(G\Sigma_\theta G^T)^{-1}(a^r - c - G\mu_\theta),\end{aligned} \quad (22)$$

and w.r.t Σ_θ as

$$\begin{aligned}&\nabla_{\Sigma_\theta}\log\pi^r_\theta(a^r|s) =\\ &= -\frac{1}{2}\nabla_{\Sigma_\theta}\log|G\Sigma_\theta G^T| - \frac{1}{2}\nabla_{\Sigma_\theta}(a^r - c - G\mu_\theta)^T(G\Sigma_\theta G^T)^{-1}(a^r - c - G\mu_\theta)\\ &= -\frac{1}{2}(G\Sigma_\theta G^T)^{-1}\nabla_{\Sigma_\theta}G\Sigma_\theta G^T\\ &\quad + \frac{1}{2}(G\Sigma_\theta G^T)^{-1}(a^r - c - G\mu_\theta)(a^r - c - G\mu_\theta)^T(G\Sigma_\theta G^T)^{-1}\nabla_{\Sigma_\theta}G\Sigma_\theta G^T\\ &= -\frac{1}{2}\left(G^T(G\Sigma_\theta G^T)^{-1}G - G^T(G\Sigma_\theta G^T)^{-1}(a^r - c - G\mu_\theta)(a^r - c - G\mu)^T(G\Sigma_\theta G^T)^{-1}G\right)\end{aligned} \quad (23)$$

We can state the following for the special case when the inverse of G exists.

Proposition 5. *If G is invertible, $\nabla_\theta \log \pi^r_\theta(a^r|s) = \nabla_\theta \log \pi_\theta(a|s)$ holds for the generator mask.*

Proof. If G is invertible, we can further simplify (22) to

$$\begin{aligned}\nabla_{\mu_\theta} \log \pi^r_\theta(a^r|s) &= G^T G^{-T} \Sigma_\theta^{-1} G^{-1}(a^r - c - G\mu_\theta) \\ &= \Sigma_\theta^{-1}\left(G^{-1}(a^r - c) - G^{-1}G\mu_\theta\right) \\ &= \Sigma_\theta^{-1}\left(a - \mu_\theta\right) = \nabla_{\mu_\theta} \log \pi_\theta(a|s),\end{aligned} \tag{24}$$

and (23) to

$$\begin{aligned}\nabla_{\Sigma_\theta} \log \pi^r_\theta(a^r|s) &= -\frac{1}{2}\left(\Sigma_\theta^{-1} - \Sigma_\theta^{-1} G^{-1}(a^r - c - G\mu_\theta)(a^r - c - G\mu_\theta)^T G^{-T}\Sigma_\theta^{-1}\right) \\ &= -\frac{1}{2}\left(\Sigma_\theta^{-1} - \Sigma_\theta^{-1}(a - \mu_\theta)(a - \mu_\theta)^T\Sigma_\theta^{-1}\right) = \nabla_{\Sigma_\theta} \log \pi_\theta(a|s),\end{aligned} \tag{25}$$

by using $a = g^{-1}(a^r) = G^{-1}(a^r - c)$, and thus proving the statement. □

A.3 Computation of the relevant action set

A.3.1 General case

Given an initial state s_0, an input trajectory $a_{(\cdot)}$, and a disturbance trajectory $w_{(\cdot)}$, we denote the solution of (2) at time t as $\xi_t(s_0, a_{(\cdot)}, w_{(\cdot)})$. Assuming a sampled controller with piecewise-constant input a, we define the reachable set of (2) after one time step Δt given a set of initial states $\mathcal{S}_0$ and a set of possible inputs $\tilde{\mathcal{A}} \subseteq \mathcal{A}$ as

$$\mathcal{R}^e_{\Delta t}(\mathcal{S}_0, \tilde{\mathcal{A}}) = \left\{\xi_{\Delta t}(s_0, a, w_{(\cdot)}) \,\middle|\, \exists x_0 \in \mathcal{S}_0, \exists a \in \tilde{\mathcal{A}}, \forall t\colon \exists w_t \in \mathcal{W}\right\}. \tag{26}$$

The reachable set over the time interval $[0, \Delta t]$ is defined as

$$\mathcal{R}^e_{[0,\Delta t]}(\mathcal{S}_0, \tilde{\mathcal{A}}) = \bigcup_{\tau \in [0,\Delta t]} \mathcal{R}^e_\tau(\mathcal{S}_0, \tilde{\mathcal{A}}). \tag{27}$$

For many system classes it is impossible to compute the reachable set exactly [28], which is why we generally compute overapproximations $\mathcal{R}(\cdot) \supseteq \mathcal{R}^e(\cdot)$.

To guarantee constraint satisfaction over an infinite time horizon, we use a relevant state set $\mathcal{S}^r$. In this work, we choose $\mathcal{S}^r$ to be a robust control invariant set in the sense that we can always find an input which guarantees that the reachable set at the next time step is contained in $\mathcal{S}^r$ and that all state constraints are satisfied in the time interval in between [32]:

$$\exists a \in \mathcal{A}\colon \mathcal{R}_{\Delta t}(\mathcal{S}^r, a) \subseteq \mathcal{S}^r, \mathcal{R}_{[0,\Delta t]}(\mathcal{S}^r, a) \subseteq \mathcal{S}. \tag{28}$$

We compute the relevant action set as the largest set of inputs that allows us to keep the system in the relevant state set in the next time step. We define a parameterized action set $\mathcal{A}^r(p)$, where $p \in \mathbb{R}^{n^p}$ is a parameter vector. The relevant action set is then computed with the optimal program

$$\begin{aligned}\max_p \quad & \widetilde{\texttt{Vol}}\left(\mathcal{A}^r(p)\right) \\ \text{subject to} \quad & \mathcal{A}^r(p) \subseteq \mathcal{A} \\ & \mathcal{R}_{\Delta t}(\mathcal{S}_0, \mathcal{A}^r(p)) \subseteq \mathcal{S}^r \\ & \mathcal{R}_{[0,\Delta t]}(\mathcal{S}_0, \mathcal{A}^r(p)) \subseteq \mathcal{S},\end{aligned} \tag{29}$$

where $\widetilde{\texttt{Vol}}(\cdot)$ is a proxy function for the volume in the sense that a maximization of $\widetilde{\texttt{Vol}}(\cdot)$ is suboptimally maximizing the volume. In the following, we provide a detailed formulation of (29) as an exponential cone program.

https://doi.org/10.52202/079017-3013

A.3.2 Exponential cone program

We consider the discrete-time linearization of our system

$$s_{k+1} = As_k + Ba_k + w'_k, \tag{30}$$

where $w'_k \in \mathcal{W}'(s_k)$ additionally contains linearization errors and the enclosure of all possible trajectory curvatures between the two discrete time steps [2]. Furthermore, we assume $\mathcal{S}$, $\mathcal{A}$, $\mathcal{W}$, and $\mathcal{S}^r$ to be zonotopes.

With regard to the input set parameterization, we consider a template zonotope with a predefined template generator matrix $\tilde{G}$. We use the vector $\tilde{p} \in \mathbb{R}^P_{>0}$ of generator scaling factors to scale the generator matrix. The parameterized template zonotope is then given by

$$\mathcal{A}^r(p) = \langle c, \tilde{G}\, diag(\tilde{p}) \rangle_{\mathcal{Z}}, \tag{31}$$

where $p = \begin{bmatrix} c & \tilde{p} \end{bmatrix}^\top$. In the following, we denote the center and generator matrix of a specific zonotope $\mathcal{Z}$ by $c^{\mathcal{Z}}$ and $G^{\mathcal{Z}}$, respectively. A zonotope $\mathcal{Z}_1$ is contained in a zonotope $\mathcal{Z}_2$ if there exist $\Gamma \in \mathbb{R}^{P^{\mathcal{Z}_2} \times P^{\mathcal{Z}_1}}$, $\beta \in P^{\mathcal{Z}_2}$, such that [31, Corollary 4]

$$\begin{aligned} G^{\mathcal{Z}_1} &= G^{\mathcal{Z}_2}\Gamma \\ c^{\mathcal{Z}_2} - c^{\mathcal{Z}_1} &= G^{\mathcal{Z}_2}\beta \\ \|[\Gamma, \beta]\|_\infty &\leq 1. \end{aligned} \tag{32}$$

Based on the linearized system dynamics in (30), the definitions from (4), and using $\mathcal{R}$ as a shorthand for $\mathcal{R}_{\Delta t}(\cdot)$, we have $G^{\mathcal{R}} = \begin{bmatrix} AG^{\mathcal{S}_0} & BG^{\mathcal{A}^r} & G^{\mathcal{W}} \end{bmatrix}$ and $c^{\mathcal{R}} = \begin{bmatrix} Ac^{\mathcal{S}_0} & Bc^{\mathcal{A}^r} & c^{\mathcal{W}} \end{bmatrix}$. Since we already consider trajectory curvature in the disturbance, we only need to guarantee that the relevant input set is contained in the feasible input set and that the reachable set of the next time step is contained in the relevant state set. Using the containment condition from (32), we define the auxiliary variables $\Gamma^{\mathcal{R}}$, $\beta^{\mathcal{R}}$, $\Gamma^{\mathcal{A}^r}$, and $\beta^{\mathcal{A}^r}$ and solve

$$\begin{aligned} \max_{p} \quad & \widetilde{\texttt{Vol}}\left(\mathcal{A}^r(p)\right) \\ \text{subject to} \quad & G^{\mathcal{R}} - G^{\mathcal{S}^r}\Gamma^{\mathcal{R}} = \mathbf{0} \\ & c^{\mathcal{S}^r} - c^{\mathcal{R}} - G^{\mathcal{S}^r}\beta^{\mathcal{R}} = \mathbf{0} \\ & \| \begin{bmatrix} \Gamma^{\mathcal{R}} & \beta^{\mathcal{R}} \end{bmatrix} \|_\infty \leq 1 \\ & G^{\mathcal{A}^r} - G^{\mathcal{A}}\Gamma^{\mathcal{A}^r} = \mathbf{0} \\ & c^{\mathcal{A}} - c^{\mathcal{A}^r} - G^{\mathcal{A}}\beta^{\mathcal{A}^r} = \mathbf{0} \\ & \| \begin{bmatrix} \Gamma^{\mathcal{A}^r} & \beta^{\mathcal{A}^r} \end{bmatrix} \|_\infty \leq 1, \end{aligned} \tag{33}$$

where $\mathbf{0}$ are zero matrices of appropriate dimensions. By vectorizing $\Gamma^{\mathcal{R}}$ and $\Gamma^{\mathcal{A}^r}$ and resolving the infinity norm constraints [4, Sec. 1.3], we can obtain a formulation with purely linear constraints. Since computing the exact volume of a zonotope is computationally expensive, we approximate it by computing the geometric mean of the parameterization vector:

$$\widetilde{\texttt{Vol}}(\mathcal{A}^r(p)) = \left(\prod_i \tilde{p}_i \right)^{\frac{1}{P}}. \tag{34}$$

With the cost function from (34), the problem in (33) is an exponential cone program, which is convex and can be efficiently computed [6, Sec. 2.5].

A.3.3 Seeker Reach-Avoid environment

Because of the dynamics of the Seeker Reach-Avoid environment, we can simplify the relevant action set computation to the following optimization problem:

$$\begin{aligned} \max_{p} \quad & \widetilde{\texttt{Vol}}(\mathcal{A}^r(p)) \\ \text{subject to} \quad & \mathcal{A}^r(p) \subseteq \mathcal{A} \\ & \mathcal{S}^r_{\Delta t} = s \oplus \mathcal{A}^r(p) \\ & \mathcal{S}^r_{\Delta t} \subseteq [-10, 10]^2 \\ & \mathcal{S}^r_{\Delta t} \cap \mathcal{O} = \emptyset, \end{aligned} \tag{35}$$

where $\mathcal{S}_{\Delta t}$ is the reachable set of the agent in the next time step, the set $\mathcal{O}$ represents the obstacle, and the box $[-10, 10]^2$ is the outer boundary of the state space. The constraints represent three intuitive geometric constraints. First, containment of the relevant action set in the action set (usually, the interval $[-1, 1]^2$, second, the containment of the relevant state set of the next time step in the environment boundaries, and the last constraint enforces that there is no collision with the obstacle possible. To enforce these constraints for zonotopes, we use the support function [3, Lemma 1]

$$\rho_{\mathcal{Z}}(l) = l^T c + \sum_{i=1}^{P} \left| l^T G_{(\cdot,i)} \right|, \tag{36}$$

where l is the vector in the direction of interest. For the first constraint, we need to ensure that the support function for the basis vectors e_1, and e_2 (and there negative versions) are less than or equal to 1, i.e.,

$$\begin{bmatrix} \rho_{\mathcal{A}^r}(e_1) \\ \rho_{\mathcal{A}^r}(e_2) \end{bmatrix} = \begin{bmatrix} c_1^{\mathcal{A}^r} + \sum_{i=1}^{P} \left| G_{(1,i)}^{\mathcal{A}^r} \right| \\ c_2^{\mathcal{A}^r} + \sum_{i=1}^{P} \left| G_{(2,i)}^{\mathcal{A}^r} \right| \end{bmatrix} = c^{\mathcal{A}^r} + \sum_{i=1}^{P} \left| G_{(\cdot,i)}^{\mathcal{A}^r} \right| \leq \begin{bmatrix} 1 \\ 1 \end{bmatrix}. \tag{37}$$

From (36) it is apparent that using the negative basis vectors $-e_1$ and $-e_2$, just flips the sign of $c^{\mathcal{A}^r}$ in (37). The constraints for $\mathcal{S}_{\Delta t}^r \subseteq [-10, 10]^2$ can be constructed similarly. For simplicity, we express the element-wise inequality in (37) using the L_∞-norm, and write the optimization problem as

$$\begin{aligned} \max_{p} \quad & \widetilde{\texttt{Vol}}(\mathcal{A}^r(p)) \\ \text{subject to} \quad & \| c^{\mathcal{A}^r} + \sum_i |G_{(\cdot,i)}^{\mathcal{A}^r}| \|_\infty \leq 1 \\ & \| - c^{\mathcal{A}^r} + \sum_i |G_{(\cdot,i)}^{\mathcal{A}^r}| \|_\infty \leq 1 \\ & \| c^{\mathcal{A}^r} + s + \sum_i |G_{(\cdot,i)}^{\mathcal{A}^r}| \|_\infty \leq 10 \\ & \| - c^{\mathcal{A}^r} + s + \sum_i |G_{(\cdot,i)}^{\mathcal{A}^r}| \|_\infty \leq 10 \\ & n^T (c^{\mathcal{A}^r} + s) + \sum_i |n^T G_{(\cdot,i)}^{\mathcal{A}^r}| \leq b. \end{aligned} \tag{38}$$

The last constraint represents an under-approximation of $\mathcal{S}_{\Delta t}^r \cap \mathcal{O} = \emptyset$ enforcing the containment of $\mathcal{A}^r$ in the halfspace $\{x | n^T x \leq b\}$ through the support function $\rho_{\mathcal{A}^r}(n)$. The directional vector $n = \frac{o-s}{\|o-s\|}$ where o is the center of the obstacle, and the offset $b = n^T(o - n)r$ where r is the radius of the obstacle. This ensures that the intersection between the reachable set $\mathcal{S}_{\Delta t}^r$ and the obstacle $\mathcal{O}$ will be empty since the halfspace is constructed tangential to the obstacle at the intersection point of the line from the agent to the center of the obstacle.

A.4 Computation of zonotope boundary points

The boundary point $p \in \mathbb{R}^N$ on a zonotope $\langle c, G \rangle_{\mathcal{Z}} \subset \mathbb{R}^N$ in a direction $d \in \mathbb{R}^N$ starting from a point $x \in \mathbb{R}^N$ is obtained by solving the linear program

$$\begin{aligned} \min_{\alpha \in \mathbb{R}, \gamma \in \mathbb{R}^N} \quad & \alpha \\ \text{subject to} \quad & x + \alpha d = c + G\gamma \\ & \|\gamma\|_\infty \leq 1 \end{aligned} \tag{39}$$

and computing $p = x + \alpha d$ [20].

A.5 Parameters and dynamics for environments

We provide the action space bounds, the state space bounds, and the generator template matrix $\tilde{G}$ for the Seeker and Quadrotor environments in Table 2.

https://doi.org/10.52202/079017-3013

Table 2: Parameters for important sets of environments.

Parameter	Seeker	2D Quadrotor	3D Quadrotor
Lower bound actions	[-1, -1]	[6.83, 6.83]	[-9.81, -0.5, -0.5, -0.5]
Upper bound actions	[1, 1]	[8.59, 8.59]	[2.38, 0.5, 0.5, 0.5]
Lower bound states	[-10, -10]	[-1.7, 0.3, -0.8, ... -1, -π/12, -π/2]	[-3, -3, -3, -3, -3, -3, ... -π/4, -π/4, -π, -3, -3, -3]
Upper bound states	[10, 10]	[1.7, 2.0, 0.8, ... 1.0, π/12, π/2]	[3, 3, 3, 3, 3, 3, ... π/4, π/4, π, 3, 3, 3]
Template matrix $\tilde{G}$	$\begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & -1 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & 1 & 1 \\ 1 & -1 & 0 \end{bmatrix}$	I_4

The system dynamics of the 2D Quadrotor is:

$$\dot{s} = \begin{pmatrix} \dot{x} \\ \dot{z} \\ (a_1 + a_2)k\sin(\theta) \\ -g + (a_1 + a_2)k\cos(\theta) \\ \dot{\theta} \\ -d_0\theta - d_1\dot{\theta} + n_0(-a_1 + a_2) \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ w_1 \\ w_2 \\ 0 \\ 0 \end{pmatrix}, \tag{40}$$

where the state is $s = \left[x, z, \dot{x}, \dot{z}, \theta, \dot{\theta}\right]^T$, the action is $a = [a_1, a_2]^T$, and the disturbance is $w = [w_1, w_2]^T$. The dynamics are adapted from [24, Eq. 43] so that the actions are two independent thrusts. Additionally, the parameters used in our experiments are $g = 9.81\,\mathrm{m\,s^{-2}}$, $k = 1\,1/\mathrm{kg}$, $d_0 = 70$, $d_1 = 17$, $n_0 = 55$, and $\mathcal{W} = [-0.08, -0.08] \times [0.08, 0.08]$.

The 3D Quadrotor is modeled in a twelve-dimensional state space with state $s = [x, y, z, \dot{x}, \dot{y}, \dot{z}, \theta, \phi, \psi, \dot{\theta}, \dot{\phi}, \dot{\psi}]^T$ and four-dimensional action space with action $a = [a_1, a_2, a_3, a_4]^T$. The system dynamics are $\dot{s} = [\dot{x}, \dot{y}, \dot{z}, -9.81\phi, 9.81\theta, a_1, \dot{\theta}, \dot{\phi}, \dot{\psi}, a_2, a_3, a_4]^T$ [16].

For the Walker2D environment, we use the standard parameters of the gymnasium implementation[2]. The relevant action set is a zonotope under-approximation of the relevant action set stated in (20) with 36 generators and $\alpha_P = 1$.

A.6 Computational time for training

Note that the increased computation time of the masking approaches compared to the baseline is mainly due to the computation of the relevant action sets. Since the relevant action set is a zonotope, the generator mask does not require expensive additional computations. The increased runtime for the ray mask mainly originates from the computation of the boundary points (see Appendix A.4). For the distributional mask, the increased runtime is mostly caused by the Markov chain Monte Carlo sampling of the action.

Table 3: Mean and standard deviation of runtime in hours for training runs on the utilized machine.

	Seeker	2D Quadrotor	3D Quadrotor	Walker2D
Baseline	0.045 ± 0.001	1.049 ± 0.197	0.493 ± 0.077	0.350 ± 0.007
Ray mask	0.865 ± 0.013	3.140 ± 0.097	2.031 ± 0.258	0.962 ± 0.043
Generator mask	0.699 ± 0.016	2.533 ± 0.021	1.528 ± 0.010	0.557 ± 0.021
Distributional mask	1.620 ± 0.017	4.083 ± 0.080	3.130 ± 0.045	≈ 60
Replacement	0.823 ± 0.015	2.782 ± 0.048	1.765 ± 0.033	0.880 ± 0.039

[2] `https://gymnasium.farama.org/environments/mujoco/walker2d/`

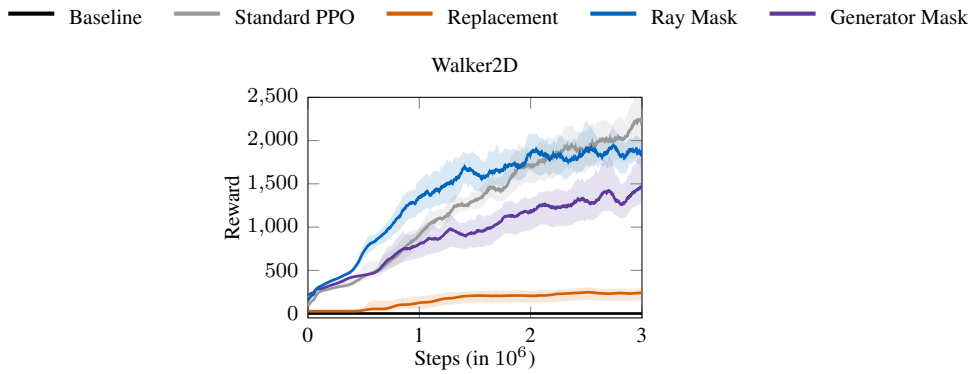


Figure 4: Average reward curves for Walker2D with transparent bootstrapped 95% confidence interval including standard PPO as additional baseline.

A.7 Quantitative results for Walker2D environment compared to standard PPO

If we remove the power constraint that resets the environment whenever an action outside of $\mathcal{A}^r$ is selected, the standard PPO is able to learn a policy that performs slightly better than the ray mask towards the end of training while converging slightly slower (see Figure 4). However, during deployment, only 0.05% of the actions from this policy are from within $\mathcal{A}^r$. Yet, the reward that standard PPO achieves during deployment is on average 2177.57 with standard deviation 1238.42. This is slightly higher than the deployment results for the ray mask (see Table 1).

A.8 Qualitative results for the Seeker environment

Figure 5 presents ten example trajectories for a trained agent using each of the five approaches under consideration. Notably, the continuous action masking agents solve the task by safely reaching the goal while the PPO baseline still collides with the obstacle. The replacement approach is ineffective, with the agent frequently failing to reach the goal. In fact, for the ten displayed rollouts, no replacement agent reaches the goal. This discrepancy with respect to the training performance where the replacement agent reaches the goal frequently is due to using the policy in a deterministic setting for deployment. The simple dynamics of the Seeker environment make it possible to directly visualize the relevant action set $\mathcal{A}^r$ at each time step along the trajectory, which is depicted in the lower half of Figure 5. For the generator mask, the least amount of $\mathcal{A}^r$ is plotted, which indicates that the generator mask agent reaches the goal most efficiently, i.e., the lowest amount of steps are required.

A.9 Hyperparameters for learning algorithms

We specify the hyperparameters for the three masking approaches, replacement and baseline in the Seeker (Table 4), the 2D Quadrotor (Table 5), and the 3D Quadrotor environment (Table 6). These were obtained through hyperparameter optimization with 50 trials using the tree-structured Parzen estimator of Optuna with the default parameters [1].

https://doi.org/10.52202/079017-3013

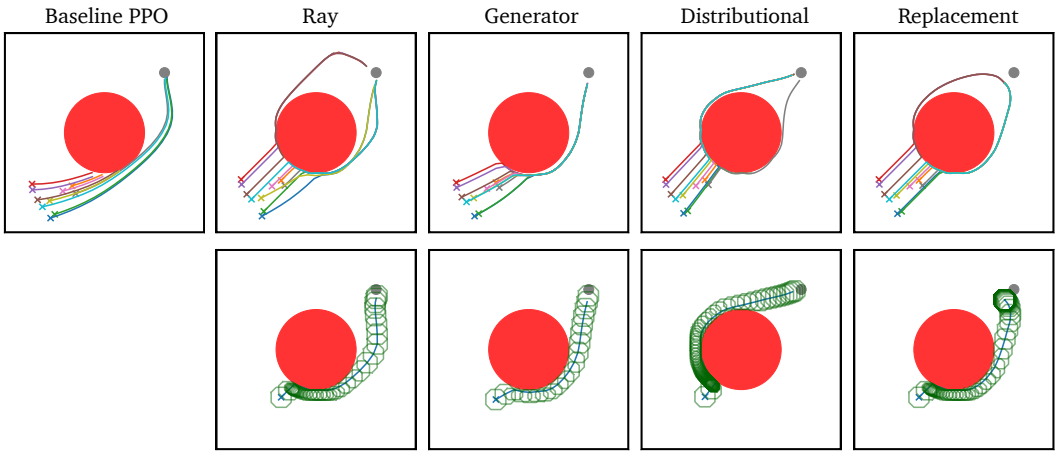


Figure 5: Qualitative deployment results for ten initial states and one goal-obstacle configuration for the Seeker environment. The top half shows ten trajectories with randomly sampled starting states. The lower half depicts the relevant action set (green polygon) for each time step along one trajectory.

Table 4: PPO hyperparameters for the Seeker environment.

Parameter	Baseline	Ray	Generator	Distributional	Replacement
Learning rate	$5.43\text{E}-5$	$8.25\text{E}-4$	$3.45\text{E}-4$	$3.85\text{E}-5$	$1.92\text{E}-6$
Discount factor γ	0.98	0.98	0.98	0.98	0.98
Steps per update	32	256	2084	32	128
Optimization epochs	4	8	16	4	4
Minibatch size	8	128	256	8	128
Max gradient norm	0.9	0.9	0.9	0.9	0.9
Entropy coefficient	$4.71\text{E}-5$	$1.66\text{E}-7$	$6.61\text{E}-7$	$3.33\text{E}-6$	$1.83\text{E}-7$
Initial log std dev	-1.183	-0.010	-0.255	-1.213	-1.064
Value function coeff.	0.5	0.5	0.5	0.5	0.5
Clipping range	0.1	0.1	0.1	0.1	0.1
GAE λ	0.9	0.9	0.9	0.9	0.9
Activation function	ReLU	ReLU	ReLU	ReLU	ReLU
Hidden layers	2	2	2	2	2
Neurons per layer	32	32	32	32	32

Table 5: PPO hyperparameters for the 2D Quadrotor environment.

Parameter	Baseline	Ray	Generator	Distributional	Replacement
Learning rate	$1.24\text{E}-4$	$7.92\text{E}-4$	$4.34\text{E}-3$	$3.94\text{E}-4$	$1.13\text{E}-4$
Discount factor γ	0.99	0.99	0.99	0.99	0.99
Steps per update	256	1024	1024	1024	512
Optimization epochs	32	8	8	8	8
Minibatch size	64	128	128	128	128
Max gradient norm	0.9	0.9	0.9	0.9	0.9
Entropy coefficient	$8.9\text{E}-2$	$5.65\text{E}-2$	$5.08\text{E}-2$	$5.99\text{E}-3$	$5.42\text{E}-3$
Initial log std dev	-0.437	-0.784	-1.251	-1.217	-1.019
Value function coeff.	0.5	0.5	0.5	0.5	0.5
Clipping range	0.1	0.1	0.1	0.1	0.1
GAE λ	0.95	0.95	0.95	0.95	0.95
Activation function	ReLU	ReLU	ReLU	ReLU	ReLU
Hidden layers	2	2	2	2	2
Neurons per layer	256	256	256	256	256

Table 6: PPO hyperparameters for the 3D Quadrotor environment.

Parameter	Baseline	Ray	Generator	Distributional	Replacement
Learning rate	$2.38\text{E}-4$	$1.08\text{E}-3$	$9.24\text{E}-5$	$7.88\text{E}-4$	$6.25\text{E}-4$
Discount factor γ	0.98	0.98	0.98	0.98	0.98
Steps per update	32	128	128	64	128
Optimization epochs	8	4	16	4	4
Minibatch size	16	32	16	64	64
Max gradient norm	0.9	0.9	0.9	0.9	0.9
Entropy coefficient	$5.85\text{E}-5$	$1.14\text{E}-7$	$3.41\text{E}-7$	$2.75\text{E}-6$	$1.88\text{E}-6$
Initial log std dev	-3.609	-1.793	-1.363	-1.880	-1.582
Value function coeff.	0.5	0.5	0.5	0.5	0.5
Clipping range	0.1	0.1	0.1	0.1	0.1
GAE λ	0.9	0.9	0.9	0.9	0.9
Activation function	ReLU	ReLU	ReLU	ReLU	ReLU
Hidden layers	2	2	2	2	2
Neurons per layer	32	32	32	32	32

Table 7: PPO hyperparameters for the Walker2D environment.

Parameter	Baseline	Replacement	Ray mask	Generator mask
Learning rate	$6.992\text{E}-5$	$4.700\text{E}-3$	$2.607\text{E}-4$	$1.719\text{E}-4$
Discount factor γ	0.99	0.99	0.99	0.99
Steps per update	2048	2048	2048	2048
Minibatch size	128	64	16	32
Max gradient norm	0.603	0.336	0.156	0.152
Entropy coefficient	$6.559\text{E}-7$	$5.960\text{E}-6$	$4.992\text{E}-6$	$7.488\text{E}-5$
Clipping range	0.165	0.131	0.102	0.192
GAE λ	0.970	0.944	0.919	0.957
Value function coefficient	0.259	0.330	0.181	0.500
Activation function	ReLU	ReLU	ReLU	ReLU
Hidden layers	2	2	2	2
Neurons per layer	64	64	64	64

https://doi.org/10.52202/079017-3013

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We claim that focusing on state-dependent relevant actions can improve training efficiency and effectiveness of reinforcement learning significantly, which we demonstrate with four experiments in Section 5.2. We formally introduce three action masking methods, which can be used with convex relevant action sets, and derive their implications on the gradient in Section 4.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We highlighted the assumptions of the proposed action masking methods in Section 4. Additionally, we discuss limitations of our experimental results and the applicability of our methods in Section 5.3.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: For all three methods, we assume convex relevant action sets and a stochastic policy. Although, as mentioned in Section 5.3, the ray mask and generator mask could be adapted to non-stochastic policies. For the generator mask, we additionally assume the set representation to be a zonotope. We propose the adapted policy gradient for each method in Propositions 1, 3, and 4, and provide proofs for all three.

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We describe the necessary details to implement the three different masking approaches in Sections 4.1, 4.2, and 4.3. We provide information needed to re-implement the environments in Sections 5.1 and A.5. The computation of the relevant action set is specified in A.3. Further, the hyperparameters used for the training runs are reported in A.9. Any remaining ambiguity can be resolved by the code attached to this submission.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.

https://doi.org/10.52202/079017-3013

(c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We attached the code for the experiments including a detailed readme in the supplementary material to this submission.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: We describe the utilized method for hyperparameter tuning in Section 5, and specify the used hyperparameters, and optimizer in Appendix A.9.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: We train each configuration with ten random seeds and report the training results with 95% bootstrapped confidence intervals in transparent color in Figure 3. For the reported deployment results and computational runtime (see Table 1 and 3, respectively), we report the mean and 1-sigma standard deviation.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We reported the compute resources utilized in Section 5. The average computation times are reported in Section A.6. Note that the compute resource used for the experiments is a shared resource where also other researcher ran their experiments at the same time. Thus, runtime evaluations will only be partially reproducible.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: Our experiments do not involve human subjects or datasets. Regarding safety and security, our approaches can be directly used to prevent agents from performing unsafe actions. We cannot identify any potentially harmful consequences of our algorithms and work in general.

https://doi.org/10.52202/079017-3013

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: The paper introduces a method to exploit task knowledge via relevant action sets to achieve more efficient and effective reinforcement learning. Given the limited scope of our experiments, a detailed discussion on broader societal impact would be very speculative. However, we believe our approach has the potential to expand the use of reinforcement learning in applications where training opportunities are limited but task knowledge is easily transferable to relevant action sets and where enforcing hard constraints is important.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: We do not work with, or release any datasets, and our models have no risk for potential misuse.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.

- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: As mentioned in Section 5, we use the stable-baseline3 implementation of PPO. Furthermore, we clearly state the origin of the used Quadrotor environments in Sections 5.1.2 and 5.1.3.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: We do not release any new assets.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

https://doi.org/10.52202/079017-3013

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.