**Robotics: Science and Systems 2024**
**Delft, Netherlands, July 15-July 19, 2024**

# POLICEd RL: Learning Closed-Loop Robot Control Policies with Provable Satisfaction of Hard Constraints

Jean-Baptiste Bouvier, Kartik Nagpal, and Negar Mehr
ICON Lab, University of California Berkeley
{bouvier3, kartiknagpal, negar}@berkeley.edu

***Abstract*—In this paper, we seek to learn a robot policy guaranteed to satisfy state constraints. To encourage constraint satisfaction, existing RL algorithms typically rely on Constrained Markov Decision Processes and discourage constraint violations through reward shaping. However, such *soft constraints* cannot offer safety guarantees. To address this gap, we propose *POLICEd RL*, a novel RL algorithm explicitly designed to enforce affine *hard constraints* in closed-loop with a black-box environment. Our key insight is to make the learned policy be affine around the unsafe set and to use this affine region as a repulsive buffer to prevent trajectories from violating the constraint. We prove that such policies exist and guarantee constraint satisfaction. Our proposed framework is applicable to both systems with continuous and discrete state and action spaces and is agnostic to the choice of the RL training algorithm. Our results demonstrate the capacity of POLICEd RL to enforce hard constraints in robotic tasks while significantly outperforming existing methods. Code available at https://iconlab.negarmehr.com/POLICEd-RL/**

## I. Introduction

While reinforcement learning (RL) [52] is widely successful [10, 44, 51], its application to safety-critical tasks is challenging due to its lack of safety guarantees [20]. A common approach towards capturing safety in RL is ensuring the satisfaction of safety constraints which prevent a robot from entering unsafe regions [12]. However, verifying that a learned closed-loop policy *never* leads to any constraint violation is in general a nontrivial problem. To remedy this shortcoming, safe RL has mostly relied on reward shaping to penalize the policy for constraint violations [12, 27, 37]. At a high level this approach corresponds to imposing *soft* safety constraints and does not provide any guarantees of constraint satisfaction by the closed-loop system [27]. However, for many safety-critical tasks, like human-robot interaction, autonomous driving [45], or Airborn Collision Avoidance Systems [34], such safety guarantees are paramount and require maintaining inviolable *hard constraints* in closed loop with the learned policy.

In this paper, we seek to learn a robot control policy guaranteed to satisfy an affine state constraint in a deterministic but black-box environment. The state space region where this constraint is not satisfied corresponds to an unsafe area that must be avoided by the robot. Our key insight is to transform the state space surrounding this unsafe area into a repulsive buffer region as shown in Fig. 1. This buffer is made repulsive by learning a policy whose actions push the robot state away from the unsafe area. To enable analytical verification of the repulsive character of our learned policy, we constrain its outputs to be affine over the buffer region. This key property allows to easily guarantee whether an affine constraint is satisfied.

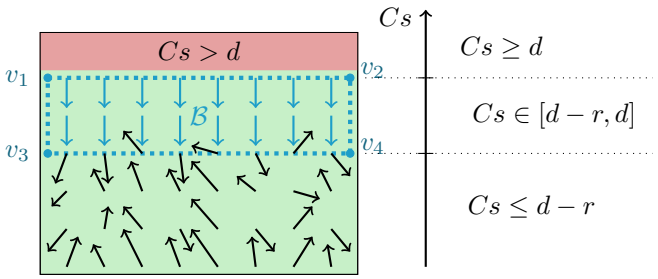



Fig. 1: Schematic illustration of POLICEd RL. To prevent state $s$ from violating an affine constraint represented by $Cs \leq d$, our POLICEd policy enforces $C\dot{s} \leq 0$ in buffer region $\mathcal{B}$ (blue) directly below the unsafe area (red). The POLICEd policy (arrows in the environment) is affine inside buffer region $\mathcal{B}$ (delimited by vertices $v_1, \ldots, v_4$), which allows us to easily verify whether trajectories can violate the constraint.

Our proposed framework is agnostic to the choice of the RL training algorithm but relies on its convergence to guarantee safety. Our method is applicable to both systems with continuous and discrete state and action spaces. Additionally, our approach can accommodate black-box environments by using a local measure of their nonlinearity, which can be numerically estimated. To the best of our knowledge, no other work learns a policy in such a way that the closed-loop system is guaranteed to satisfy a hard constraint after training.

To ensure our policy produces affine outputs over the whole buffer region, we draw from existing literature in constraining neural network outputs. While this topic has been widely investigated [2, 5, 19, 21, 35, 54], we build on the POLICE algorithm proposed in [8] to constrain our learned policy. Indeed, POLICE is capable of ensuring that a deep neural network with continuous piecewise affine activation functions produces affine outputs over a user-specified region. We build on this work and develop a paradigm for enforcing hard constraints in RL which we call POLICEd RL.

Following this approach, we establish analytical conditions under which our learned policy guarantees constraint satisfaction after training. A natural follow-up question arising from these conditions is whether they are so stringent that no constraint-satisfying policy exist. To answer this crucial problem, we transform the question of existence of such a constraint-satisfying policy into a tractable linear problem. We will then demonstrate through a number of numerical examples how our method can be implemented. Then, using a 7DOF robotic arm, we compare the performance of our method with a number of representative baselines and demonstrate that our method outperforms all the baselines both in terms of its constraint satisfaction as well as its expected accumulated reward.

In this work, we focus on constraining only *actuated states* in an effort to lighten the complexity of our theory. Such a setting is commonly used in a wide range of safety research such as the literature using of control barrier functions [4, 42, 59, 64] as well as safe RL [16, 33, 41, 70]. We will provide a detailed review of these works in Section II-C.

In summary, our contributions in this work are as follows.

1) We introduce POLICEd RL, a novel RL framework that can guarantee satisfaction of hard constraints by the closed-loop system composed of a black-box robot model and a trained policy.
2) We transform the question of existence of such a constraint-satisfying policy into a tractable linear problem.
3) We demonstrate the efficiency of our proposed POLICEd RL at guaranteeing the safety of an inverted pendulum and a robotic manipulator in a number of simulations using high-fidelity MuJoCo simulators [53].

The remainder of this work is organized as follows. In Section II we provide a survey of related works. In Section III we describe prior work [8] upon which we build our approach. In Section IV we introduce our problem formulation along with our framework. In Section V we establish our approach and provide its theoretical guarantees to enforce the satisfaction of affine constraints. In Section VI we demonstrate how to implement our proposed POLICEd RL algorithm. In Section VII we present several numerical simulations illustrating our approach. Finally, we conclude the paper in Section VIII.

*Notation:* The characteristic function of a set $\mathcal{S} \subseteq \mathbb{R}^n$ is denoted by $\mathbb{1}_{\mathcal{S}}$. The positive integer interval from $a \in \mathbb{N}$ to $b \in \mathbb{N}$ inclusive is denoted by $[\![a, b]\!]$. Uniform sampling of a variable $x$ in a set $\mathcal{X}$ is denoted by $x \sim \mathcal{U}(\mathcal{X})$.

## II. Related works

### A. Enforcing hard constraints on neural network outputs

First, we review the literature on enforcing hard constraints on the outputs of deep neural networks (DNNs). Adding a post-processing layer or an extra activation function like $tanh$ or $sigmoid$ to a DNN can easily bound its output. Similarly, linear constraints can be enforced by projecting DNN outputs onto a constraint set using quadratic programming optimization layers [5]. To enforce general convex constraints, [2] developed differentiable convex optimization layers that can be incorporated into DNNs. However, evaluating these optimization layers is computationally expensive, which led to imposing linear constraint without any projection in [21] where the constraint is built into the DNN's architecture. This work was recently extended to enforce affine constraints in localized regions [8]. The main advantage of these two works [8, 21] is the absence of computational overhead at deployment where such constrained DNNs become simple multilayer perceptrons. To go beyond affine constraints, [19] used gradient descent along equality constraints until inequality constraints are also satisfied. Observing that gradient descent can require numerous iterations and suffers convergence issues prompted a more efficient method for hard convex constraint enforcement through the offline computation of feasible sets in [54]. In a concurrent work, [35] built DNNs whose outputs satisfy linear and quadratic constraints without solving any optimization problem in the forward pass. All these works enforce constraints on DNN outputs only and do not consider the satisfaction of hard constraints by a trained policy.

### B. Constraints in reinforcement learning

The most common approach to enforce constraints in RL adopts the framework of constrained Markov decision processes (CMDPs) [3]. CMDPs encourage policies to respect constraints by penalizing the expectation of the cumulative constraint violations [37]. Numerous variations of this framework have been developed such as state-wise constrained MDP [68], constrained policy optimization [1], and state-wise constrained policy optimization [70]. These approaches belong to the category of *soft constraints* as the policy is only *encouraged* to respect the constraint and provides no satisfaction guarantees [27]. This category also encompasses work [42] where a control barrier transformer is trained to avoid unsafe actions, but no safety guarantees can be derived.

A *probabilistic* constraint comes with the guarantee of satisfaction with some probability threshold [12] and hence ensures a higher level of safety than soft constraints as shown red in Fig. 2. For instance, [30] derived policies having a high probability of not violating the constraints by more than a small tolerance. Using control barrier functions, [14] guaranteed safe learning with high probability. Since unknown stochastic environments prevent hard constraints enforcement, [59] proposed to learn generative model-based soft barrier functions to encode chance constraints. Similarly, by using a safety index on an unknown environment modeled by Gaussian processes, [69, 33] established probabilistic safety guarantees.

The focus of our work is at the third safety level described in [12] and illustrated in Fig. 2. This level corresponds to inviolable *hard constraints*. Works [16, 47] both learned safe policies thanks to a differentiable safety layer projecting any unsafe action onto the closest safe action. However, since these safety layers only correct actions, a precise model of the robot dynamics model must be known, which is typically unavailable in RL settings. This limitation is also shared by [49, 63],

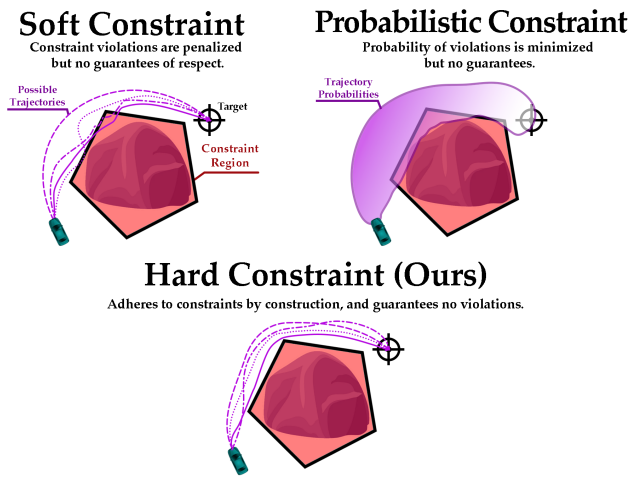



Fig. 2: The three categories of constraint satisfaction with increasing guarantees of satisfaction.

which require some knowledge of the robot dynamics model to compute either backward reachable sets [49] or control barrier functions [63]. To circumvent any knowledge of the robot dynamics model, [67] used an implicit black-box model but is restricted to collision avoidance problems in 2D. To avoid these limitations, [41, 48, 64] learn safety certificates. However, these methods used barrier function approximators and hence cannot guarantee constraint satisfaction.

To sum up, our work differs from all these works as we enforce inviolable *hard* constraints on robot trajectories in closed-loop with a learned control policy while exclusively using a black-box model of the robot dynamics.

### C. Relative degree of constraints

The relative degree of constraints is an important notion in safe RL deserving some introduction. The *relative degree of a constraint* describes how many times a constraint needs to be differentiated before a control input appears in its expression. For instance, a position constraint has relative degree 1 for a velocity-controlled robot, but relative degree 2 for an acceleration-controlled robot. The higher the relative degree, the more inertia the constraint has and the more anticipation is required to respect it.

Control barrier functions (CBFs), one of the most common tools to enforce constraints, require constraints of relative degree 1 [4]. Only recently have CBFs been specifically extended to handle constraints of higher relative degrees with, for instance, exponential CBFs [46], generalized CBFs [40], and high order CBFs (HOCBF) [62]. However, these CBF methods require knowledge of the robot dynamics model, which is typically not available in RL. To address this limitation, learning a CBF is a prevalent approach in safe RL [42, 48, 59, 64]. These works still choose to focus on constraints of relative degree 1 as do most of the safe RL literature [16, 33, 41, 70]. We follow this common approach and choose to study constraints of relative degree 1 in this work.

### D. Black-box safety with control theory

While CBFs are prevalent in safe RL [42, 48, 59, 64], other approaches backed by control theory have also investigated safety in black-box RL. Most notably, model predictive control (MPC) is particularly well-suited to bring safety guarantees to RL settings [29]. MPC schemes predict safe and optimal control inputs in a receding-horizon fashion by using a model of the dynamics to anticipate their behavior [29]. Implementing MPC for black-box systems typically requires learning either a robust [6, 18] or stochastic [39] model of the dynamics. Then, the theory of robust [9] and stochastic control [56] can provide safety guarantees despite the uncertainty or disturbances in the learned models [29]. Similarly to CBFs, MPC can also serve as a safety filter to enforce constraints on learned policies [29, 57]. While robust control tends to be overly conservative in practice [58] and stochastic MPC offers limited safety guarantees [43], the main limitation of black-box MPC is the unavoidable and notoriously computationally difficult receding-horizon optimization problem to be solved at each time step [29].

More closely related to this work is the literature on optimal control of constrained piecewise affine systems [15]. Indeed, nonlinear dynamics can be locally approximated as piecewise affine to develop robust controllers with provable performance guarantees [24, 28] while enforcing linear temporal logic specifications [31]. However, these works assume knowledge of the robot dynamics.

## III. Prior Work

In this section, we review prior work [8] upon which we build our approach. The POLICE algorithm [8] utilizes the spline theory of DNNs [7] to modify the training process of a DNN by guaranteeing that the DNN outputs are strictly affine within a user-specified region as shown in Fig. 3.

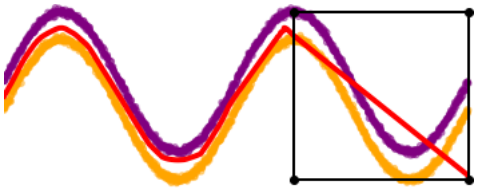


Fig. 3: Classification task of orange versus purple by a learned decision boundary (red) which is required to be affine inside the black square. POLICE [8] guarantees the DNN is affine in the region of interest.

More specifically, consider a DNN represented by a function $f_\theta : \mathbb{R}^{d_0} \to \mathbb{R}^{d_L}$ composed of $L$ layers of width $d_1, ..., d_L$ defined as $f_\theta := f^{(L)}_{\theta_L} \circ \ldots \circ f^{(1)}_{\theta_1}$. Let layer $i \in [\![1, L]\!]$ take the form $f^{(i)}_{\theta_i}(x) = \sigma\big(W^{(i)}x + b^{(i)}\big)$, where $\sigma$ is a pointwise activation function, $W^{(i)} \in \mathbb{R}^{d_i \times d_{i-1}}$ is a weight matrix, and $b^{(i)} \in \mathbb{R}^{d_i}$ is a bias vector. The layer parameters are gathered into $\theta_i := \{W^{(i)},\ b^{(i)}\}$ while the later parameters themselves are gathered as $\theta := \{\theta_1, \ldots, \theta_L\}$.

The framework of [8] assumes that nonlinearities $\sigma$ are continuous piecewise affine functions such as (leaky-)ReLU,

absolute value or max-pooling. This assumption is common in the field of DNN verification [13, 60, 65, 66].

Under this assumption, $f_\theta$ is a composition of affine functions $W^{(i)}x+b^{(i)}$ and piecewise affine functions $\sigma$. As a result, the DNN $f_\theta$ itself is also piecewise affine. Hence, we can partition input space $\mathbb{R}^{d_0}$ into $M$ regions $\mathcal{R}_j$, $j \in [\![1, M]\!]$, over which $f_\theta$ is affine. The nonlinearities of $f_\theta$ occur at the boundaries between regions $\mathcal{R}_j$. Then,

$$f_\theta(x) = \sum_{j=1}^{M} \big(D_j x + e_j\big) \mathbb{1}_{x \in \mathcal{R}_j}, \tag{1}$$

where $D_j \in \mathbb{R}^{d_L \times d_0}$, and $e_j \in \mathbb{R}^{d_L}$ are per-region slope and offset matrices, and regions $\mathcal{R}_j \subseteq \mathbb{R}^{d_0}$ are such that $\bigcup_{j=1}^{M} \mathcal{R}_j = \mathbb{R}^{d_0}$. These regions $\mathcal{R}_j$ are defined by their characteristic functions $\mathbb{1}_{x \in \mathcal{R}_j}$. The formulation of (1) is discussed at length in [55], which also describes how to compute regions $\mathcal{R}_j$ and matrices $D_j$ and $e_j$ based on the parameter vector $\theta$.

The POLICE algorithm [8] builds on this formulation to impose one user-specified polytopic region $\mathcal{R}_{\text{user}}$ into the partition of input space $\mathbb{R}^{d_0}$. As a result, DNN $f_\theta$ becomes affine on this specific region $\mathcal{R}_{\text{user}}$. This affine region is enforced by an extra bias term $b_{\text{extra}}$ calculated for each layer such that the whole region $\mathcal{R}_{\text{user}}$ possesses the same activation pattern. Note that standard unconstrained optimization is still used to train the DNN. Once training is over, a POLICEd DNN behaves similarly to a standard DNN as the affine region enforcement is guaranteed by the stored bias of each layer. Thus, POLICEd DNNs have no overhead at test time compared to standard unconstrained DNNs [8].

## IV. Framework

Let us now introduce the framework of our problem of interest. We consider a robot of state $s$ with deterministic dynamics modeled by a continuous function $f$ and of form

$$\dot{s}(t) = f\big(s(t), a(t)\big), \quad a(t) \in \mathcal{A}, \quad s(0) \sim \rho_0, \tag{2}$$

where $a$ is the action input and $\rho_0$ is the distribution of initial states. The assumption of deterministic dynamics is common in the literature of safe RL with hard constraints as exhibited by [16, 41, 48, 49, 63, 64, 67, 70]. Let state space $\mathcal{S}$ be a compact convex polytope of $\mathbb{R}^n$ and admissible action set $\mathcal{A}$ be a compact convex subset of $\mathbb{R}^m$. Similar assumptions are adopted by [41, 49, 63]. The state of the robot is constrained by a single affine inequality

$$y(t) := Cs(t) \leq d, \quad \text{for all } t \geq 0, \tag{3}$$

with $C \in \mathbb{R}^{1 \times n}$ and $d \in \mathbb{R}$. The robot starts from an initial state $s(0) \sim \rho_0$. At every time instant $t \geq 0$, the robot observes its state $s(t) \in \mathcal{S}$ and takes an action according to its deterministic feedback policy $a(t) = \mu_\theta\big(s(t)\big) \in \mathcal{A}$ modeled by a DNN $\mu_\theta$ parameterized by $\theta$. Then, the robot receives a reward $R\big(s(t), a(t)\big)$. Our objective is to train policy $\mu_\theta$ to maximize the expected discounted reward over trajectories generated by policy $\mu_\theta$ while respecting constraint (3):

$$\max_\theta \mathcal{G}(\mu_\theta) := \mathop{\mathbb{E}}_{s_0 \sim \rho_0} \int_0^\infty \gamma^t R\big(s(t), \mu_\theta(s(t))\big) dt \text{ s.t. (3)}, \tag{4}$$

where $\gamma \in (0, 1]$ is a discount factor. We emphasize that constraint (3) is a *hard constraint* to be respected at all times. We can now formally define our problem of interest.

**Problem 1.** *Given:*

1) *black box continuous control system (2);*
2) *compact convex polytopic state space $\mathcal{S} \subset \mathbb{R}^n$;*
3) *compact convex admissible action set $\mathcal{A} \subset \mathbb{R}^m$;*
4) *affine hard constraint (3);*
5) *DNN policy $\mu_\theta\big(s(t)\big)$ parameterized by $\theta$;*

*our goal is to solve*

$$\theta^* = \arg\max_\theta \mathcal{G}(\mu_\theta) \quad \textit{s.t.} \quad \dot{s}(t) = f\big(s(t), \mu_\theta(s(t))\big),$$
$$s(0) \sim \rho_0, \quad Cs(t) \leq d, \quad \textit{for all } t \geq 0.$$

Our approach focuses on deterministic dynamics with no model mismatch, although it can be readily extended to uncertain dynamics thanks to robust safe control [17]. Additionally, we consider the robot dynamics model $f$ to be an implicit black-box, meaning that we can evaluate $f$ but we do not have access to the equations or analytical form of $f$. This is similar to the online RL setting where $f$ is a simulator or where $f$ is the actual robot.

## V. Constrained Reinforcement Learning

In this section, we establish a method to solve Problem 1. To enforce the satisfaction of affine constraint $Cs \leq d$, we construct a repulsive buffer where $C\dot{s} \leq 0$ around the constraint line $Cs = d$ as illustrated in Fig. 1. This repulsive buffer will then guarantee that closed-loop trajectories of the robot cannot breach the constraint. We will establish the analytical safety guarantees of our method in this section and then in the next section, we discuss the implementation details of our approach.

### A. Guaranteed satisfaction of hard constraints

We start by constructing a repulsive buffer in front of the constraint violation line defined by (3). Let $r > 0$ be the 'radius' of this buffer defined as

$$\mathcal{B} := \big\{s \in \mathcal{S} : Cs \in [d - r, d]\big\}. \tag{5}$$

Note that any state trajectory initially verifying constraint (3), i.e., $Cs(0) \leq d$, has to cross buffer $\mathcal{B}$ before being able to violate the constraint. Therefore, if buffer $\mathcal{B}$ cannot be crossed, then constraint (3) cannot be violated. To design a policy $\mu_\theta$ incapable of crossing buffer $\mathcal{B}$, we need the following result.

**Lemma 1.** *Buffer $\mathcal{B}$ of (5) is a polytope.*

*Proof:* The proof is located in Appendix A. ∎

Then, buffer $\mathcal{B}$ has a finite number $N$ of vertices gathered in the set $\mathcal{V}(\mathcal{B}) := \{v_1, \ldots, v_N\}$.

We choose a deterministic policy modeled by a POLICEd DNN $\mu_\theta : \mathcal{S} \to \mathcal{A}$. We adopt the framework of [8] as discussed in Section III. We assume that the activation functions of $\mu_\theta$ are continuous piecewise affine functions such as (leaky) ReLU, absolute value, or max-pooling. As mentioned in Section III, this POLICEd DNN architecture allows the user to specify a polytopic region $\mathcal{R}_{\text{user}}$ of the state space $\mathcal{S}$ where the outputs of $\mu_\theta$ are affine. We choose $\mathcal{R}_{\text{user}} = \mathcal{B}$ as illustrated in Fig. 1.

Following (1), the affine character of $\mu_\theta$ over region $\mathcal{B}$ is equivalent to the existence of matrices $D_\theta \in \mathbb{R}^{m\times n}$ and $e_\theta \in \mathbb{R}^m$ such that

$$\mu_\theta(s) = D_\theta s + e_\theta \quad \text{for all } s \in \mathcal{B}. \tag{6}$$

Having an affine policy on $\mathcal{B}$, we would like to couple it with affine robot dynamics to obtain a simple constraint enforcement process. However, in general, robot dynamics (2) are nonlinear. We will thus use an affine approximation of the robot dynamics inside buffer $\mathcal{B}$ using the following definition.

**Definition 1.** *An* approximation measure $\varepsilon$ *of dynamics* (2) *with respect to constraint* (3) *and buffer* (5) *is any* $\varepsilon \geq 0$ *for which there exists any matrices* $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$, *and* $c \in \mathbb{R}^n$ *such that*

$$\big|Cf(s,a) - C(As + Ba + c)\big| \leq \varepsilon, \tag{7}$$

*for all* $s \in \mathcal{B}$, *and all* $a \in \mathcal{A}$.

Despite this approximation, note that we will guarantee the satisfaction of constraint (3) with actual dynamics (2). To help us easily compute a value for $\varepsilon$ using system identification techniques [38] like linear least square [25], we take advantage of the following property of approximation measures.

**Lemma 2.** *Given dynamics* $f$ *of* (2), *constraint matrix* $C$ *of* (3) *and buffer* $\mathcal{B}$ *of* (5), *any* $\varepsilon$ *sufficiently large is an approximation measure in the sense of Definition 1.*

*Proof:* The proof is located in Appendix B. ∎

Note that intuitively, the value of $\varepsilon$ quantifies the quality of the approximation over buffer $\mathcal{B}$ of possibly nonlinear robot dynamics (2) by affine system

$$\dot{s}(t) = As(t) + Ba(t) + c, \quad a(t) \in \mathcal{A}. \tag{8}$$

We will show how to *guarantee* satisfaction of constraint (3) by black-box environment (2) armed *only* with an approximation measure $\varepsilon$ and *without* knowing matrices $A$, $B$, $c$ or dynamics $f$.

We define the safe states as $\mathcal{S}_s := \{s \in \mathcal{S} : Cs < d\}$. We only consider trajectories remaining in state space $\mathcal{S}$, which we define as $\tau^{\mathcal{S}}\big(s_0, a(\cdot)\big) := \big\{s(t) : s(t) \in \mathcal{S} \text{ and follows } (2)\big\}$ for all $s_0 \in \mathcal{S}$ and adequate actions $a(\cdot) \in \mathcal{A}$. We can now state our central result relating repulsive buffer $\mathcal{B}$ to trajectories $\tau^{\mathcal{S}}$ continuously respecting the constraints.

**Theorem 1.** *If for some approximation measure* $\varepsilon$, *repulsion condition*

$$Cf\big(v, \mu_\theta(v)\big) \leq -2\varepsilon, \tag{9}$$

*holds for all* $v \in \mathcal{V}(\mathcal{B})$, *then* $\tau^{\mathcal{S}}(s_0, \mu_\theta) \subseteq \mathcal{S}_s$ *for all* $s_0 \in \mathcal{S}_s$.

*Proof:* The intuition behind this proof is to use (9) and approximation (7) to show that $C\dot{s} \leq 0$ for all $s \in \mathcal{B}$, which in turn prevents trajectory from crossing buffer $\mathcal{B}$ and hence from violating the constraint.

Since $\varepsilon$ is an approximation measure, there exists matrices $A$, $B$ and $c$ verifying (7). We can now extend repulsion condition (9) from the vertices of buffer $\mathcal{B}$ to the whole set $\mathcal{B}$. For $v \in \mathcal{V}(\mathcal{B})$,

$$\begin{aligned} &C\big(Av + B\mu_\theta(v) + c\big) \\ &\leq \big|C\big(Av + B\mu_\theta(v) + c - f(v, \mu_\theta(v))\big)\big| + Cf\big(v, \mu_\theta(v)\big) \\ &\leq \varepsilon + C\dot{v} \leq \varepsilon - 2\varepsilon \leq -\varepsilon, \end{aligned} \tag{10}$$

where the first inequality is a triangular inequality, the second follows from (7) and (2), and the third inequality follows from (9). Using the fact that an affine function is uniquely determined by its values on the vertices of a full-dimensional polytope [26], [28], we will show that (10) is also valid all over polytope $\mathcal{B}$ and not just at its vertices $\mathcal{V}(\mathcal{B})$. More specifically, since $\mathcal{B}$ is the convex hull of its vertices $\{v_1, \ldots, v_N\} = \mathcal{V}(\mathcal{B})$ [26], for all $s \in \mathcal{B}$, there exists $\alpha_1, \ldots, \alpha_N \in \mathbb{R}$ such that $\alpha_k \geq 0$, $\sum_{k=1}^N \alpha_k = 1$ and $s = \sum_{k=1}^N \alpha_k v_k$. Then, (6) yields

$$\begin{aligned} &C\big(As + B\mu_\theta(s) + c\big) = C\big(As + B(D_\theta s + e_\theta) + c\big) \\ &= C(A + BD_\theta)s + C(Be_\theta + c) \\ &= C(A + BD_\theta)\sum_{k=1}^N \alpha_k v_k + C(Be_\theta + c)\sum_{k=1}^N \alpha_k \\ &= \sum_{k=1}^N \alpha_k C\big((A + BD_\theta)v_k + Be_\theta + c\big) \\ &= \sum_{k=1}^N \alpha_k C\big(Av_k + B\mu_\theta(v_k) + c\big) \leq \sum_{k=1}^N \alpha_k(-\varepsilon) = -\varepsilon, \end{aligned} \tag{11}$$

where the inequality comes from (10) applied on each vertex $v_k$. For any state $s \in \mathcal{B}$, we have

$$\begin{aligned} C\dot{s} &= Cf\big(s, \mu_\theta(s)\big) \\ &\leq \big|C\big(f(s, \mu_\theta(s)) - As - B\mu_\theta(s) - c\big)\big| \\ &\qquad + C\big(As + B\mu_\theta(s) + c\big) \\ &\leq \varepsilon - \varepsilon \leq 0, \end{aligned} \tag{12}$$

where we first use the triangular inequality, then (7) and (11). Having proved (12), we will now show that it prevents all trajectories $\tau^{\mathcal{S}}(s_0, \mu_\theta)$ from exiting safe set $\mathcal{S}_s$ when $s_0 \in \mathcal{S}_s$.

We prove this by contradiction. Assume that trajectory $\tau^{\mathcal{S}}(s_0, \mu_\theta) \not\subseteq \mathcal{S}_s$ for some $s_0 \in \mathcal{S}_s$. Since $\tau^{\mathcal{S}}(s_0, \mu_\theta) \subset \mathcal{S}$, there exists some $T > 0$ such that $s(T) \in \mathcal{S}\backslash\mathcal{S}_s$. Then, by definition of $\mathcal{S}_s$, $y(T) = Cs(T) \geq d$. Since $s_0 \in \mathcal{S}_s$, $y(0) = Cs_0 < d$. By continuity of the function $y$, there exists a time $t_2 \in (0, T]$ at which $y(t_2) = Cs(t_2) = d$. Let $t_0 \geq 0$ be the last time at which $\tau^{\mathcal{S}}(s_0, \mu_\theta)$ entered $\mathcal{B}$, so that $s(t) \in \mathcal{B}$ for all $t \in [t_0, t_2]$.

Note that $y(t) = Cs(t)$ is continuously differentiable since dynamics (2) are continuous. Then, according to the Mean

Value Theorem [32], there exists $t_1 \in (t_0, t_2)$ such that $\dot{y}(t_1) = \big(y(t_2) - y(t_0)\big)/(t_2 - t_0)$. By construction of $t_0$ and $t_2$, we have $t_2 - t_0 > 0$ and $y(t_2) - y(t_0) > 0$. Therefore, $\dot{y}(t_1) = C\dot{s}(t_1) > 0$ with $s(t_1) \in \mathcal{B}$, which contradicts (12). Therefore, all trajectories $\tau^{\mathcal{S}}(\cdot, \mu_\theta)$ starting in safe set $\mathcal{S}_s$ remain in $\mathcal{S}_s$. ∎

Theorem 1 guarantees that trajectories remaining in $\mathcal{S}$ and steered by $a(t) = \mu_\theta\big(s(t)\big)$ satisfy constraint (3) at all times as long as repulsion condition (9) is satisfied. This condition (9) guarantees that trajectories cannot cross buffer $\mathcal{B}$ and hence cannot violate constraint (3).

**Remark 1.** *In order to guarantee satisfaction of constraint (3) through Theorem 1, we need an approximation measure $\varepsilon$. In practice, we can use data-driven techniques for estimating $\varepsilon$. Our safety guarantees are valid as long as our estimator $\tilde{\varepsilon}$ overshoots the true $\varepsilon$ since that will help make $\tilde{\varepsilon}$ an approximation measure according to Lemma 2.* [1]

Theorem 1 highlights the major strength of POLICEd RL: to ensure constraint satisfaction we only need to check whether (9) holds at the vertices of $\mathcal{B}$. Without POLICE, $\mu_\theta$ would not be affine over $\mathcal{B}$, and repulsion condition (9) would need to be verified at numerous intermediary points depending on the smoothness of $\mu_\theta$ and the size of the buffer. Such an approach is similar to the $\delta$-completeness guarantees of [13, 23] and would be much more computationally intensive than our approach.

Theorem 1 and more specifically Condition (9) implicitly assume that the robot has access to $\dot{v}$, the derivative of the state, which is usually not the case in typical RL environments [11]. To address this, we provide a straightforward extension of Theorem 1 to the widespread framework of discrete-time setting. We define the discrete-time state of the robot as

$$s_{j+1} := s_j + f(s_j, a_j)\delta t, \tag{13}$$

for all $s_j \in \mathcal{S}$, $a_j \in \mathcal{A}$, $j \in \mathbb{N}$, and some time step $\delta t > 0$. The discrete trajectory of policy $\mu_\theta$ starting from $s_0 \in \mathcal{S}$ and remaining in state space $\mathcal{S}$ is defined as

$$\tau_d^{\mathcal{S}}(s_0, \mu_\theta) := \big(s_0, s_1, \dots\big) \in \mathcal{S}^{\mathbb{N}}, \tag{14}$$

such that $s_{j+1} = s_j + f(s_j, \mu_\theta(s_j))\delta t \in \mathcal{S}$ for all $j \in \mathbb{N}$.

**Corollary 1.** *If for some approximation measure $\varepsilon$, the following discrete repulsion condition holds*

$$Cf(s_j, \mu_\theta(s_j)) \leq -2\varepsilon \quad \textit{for all } s_j \in \mathcal{V}\big(\mathcal{B}\big), \tag{15}$$

*with $a_j = \mu_\theta(s_j)$, and buffer $\mathcal{B}$ is wide enough such that*

$$\max\big\{C\big(s_{j+1} - s_j\big) : s_j \in \mathcal{S}_s,\ a_j = \mu_\theta(s_j)\big\} \leq r, \tag{16}$$

*then $\tau_d^{\mathcal{S}}(s_0, \mu_\theta) \subset \mathcal{S}_s$ for all $s_0 \in \mathcal{S}_s$.*

*Proof:* The proof is similar to that of Theorem 1 and can be found in Appendix C. ∎

[1] However, in practice an excessively large value of $\varepsilon$ will be detrimental to the learning performance of $\mu_\theta$ since repulsion condition (9) will become harder to enforce as $\varepsilon$ increases.

Note that condition (16) requires the buffer to be wide enough not to be 'jumped' over in a single time step by the discrete dynamics. Condition (15) makes buffer $\mathcal{B}$ repulsive so that discrete trajectories $\tau_d^{\mathcal{S}}$ cannot leave safe set $\mathcal{S}_s$.

### B. Existence conditions

While Theorem 1 and Corollary 1 provide a way to verify the satisfaction of hard constraint (3), they do not entirely solve Problem 1. A natural remaining piece is the following existence condition.

**Problem 2.** *Under what conditions does there exist an admissible policy $\mu_\theta$ satisfying Theorem 1?*

Indeed, if the conditions of Theorem 1 cannot be satisfied, then training $\mu_\theta$ will fail to solve Problem 1, which is why Problem 2 is crucial. To address this issue, we first reformulate the conditions of Theorem 1 into a tractable existence problem.

**Proposition 1.** *Finding an admissible policy $\mu_\theta$ satisfying Theorem 1 is equivalent to finding a matrix $D_\theta \in \mathbb{R}^{m \times n}$ and a vector $e_\theta \in \mathbb{R}^m$ which satisfy the following conditions for all vertices $v \in \mathcal{V}\big(\mathcal{B}\big)$*

$$C\big((A + BD_\theta)v + Be_\theta + c\big) \leq -\varepsilon, \tag{17a}$$
$$D_\theta v + e_\theta \in \mathcal{A}. \tag{17b}$$

*Proof:* Following definition (6) of policy $\mu_\theta$, (17a) is equivalent to (10). Similarly, a policy $\mu_\theta$ satisfying Theorem 1 also verifies (10).

The only statement left to prove is the equivalence between (17b) and policy $\mu_\theta$ being admissible over buffer $\mathcal{B}$. Admissibility is formalized as $\mu_\theta(s) \in \mathcal{A}$ for all $s \in \mathcal{B}$, which combined with (6), trivially implies (17b).

Conversely, assume (17b) holds and let us prove that $\mu_\theta$ is admissible. Let $s \in \mathcal{B}$. Since $\mathcal{V}\big(\mathcal{B}\big) = \{v_1, \dots, v_N\}$ are the vertices of convex set $\mathcal{B}$, there exists $\alpha_1, \dots, \alpha_N \in \mathbb{R}$ such that $\alpha_k \geq 0$, $\sum_{k=1}^N \alpha_k = 1$ and $s = \sum_{k=1}^N \alpha_k v_k$. Then, (6) yields

$$\begin{aligned}\mu_\theta(s) = D_\theta s + e_\theta &= D_\theta \left(\sum_{k=1}^N \alpha_k v_k\right) + e_\theta \left(\sum_{k=1}^N \alpha_k\right) \\ &= \sum_{k=1}^N \alpha_k (D_\theta v_k + e_\theta).\end{aligned}$$

Then, (17b) coupled with the convexity of $\mathcal{A}$ yields $\mu_\theta(s) \in \mathcal{A}$ for all $s \in \mathcal{B}$, hence completing the proof. ∎

Proposition 1 translates the feasibility of Theorem 1 into two sets of $N$ linear conditions to verify. Additionally, if the input set $\mathcal{A}$ is a polytope, (17b) can be simplified to a linear existence problem. Hence, the question of the existence of a policy $\mu_\theta$ satisfying Theorem 1 can be answered efficiently with linear programming. Notice that (17a) admits plenty of solutions as long as $CB \neq 0$. This observation is in fact related to the concept of *relative degree* whose definition we now formalize.

**Definition 2.** *The* relative degree $\gamma$ *of system* (2) *with output* (3) *is the order of its input-output relationship, i.e.,* $\gamma := \min\left\{r \in \mathbb{N} : \frac{\partial}{\partial a}\frac{\partial^r y(t)}{\partial t^r} \neq 0\right\}$.

In simpler words, the relative degree is the minimal number of times output $y$ has to be differentiated until input $a$ appears.

**Proposition 2.** *If affine dynamics* (8) *with output* (3) *have a relative degree* 1*, then condition* (17a) *admits solutions. Otherwise,* (17a) *admits solutions if and only if* $c_v^* \leq -\varepsilon$ *where* $c_v^* := \max\left\{CAv + Cc : v \in \mathcal{V}(\mathcal{B})\right\}$.

*Proof:* Following Definition 2 with dynamics (8) and constraint (3), we calculate

$$\frac{\partial}{\partial a}\frac{\partial y(t)}{\partial t} = \frac{\partial}{\partial a}\frac{\partial Cs(t)}{\partial t} = \frac{\partial}{\partial a}C\dot{s}(t)$$
$$= \frac{\partial}{\partial a}C\big(As(t) + Ba(t) + c\big) = CB.$$

Then, a relative degree 1 entails $CB \neq 0$, i.e., there exists $j \in [\![1, m]\!]$ such that the $j^{th}$ component of $CB$ is non-zero. Then, we choose $e_\theta$ to be the zero vector of $\mathbb{R}^m$ except for its $j^{th}$ component to be $\frac{-c_v^* - \varepsilon}{[CB]_j}$, so that $CBe_\theta = -c_v^* - \varepsilon$. We also choose $D_\theta = 0 \in \mathbb{R}^{m \times n}$. Then, for all $v \in \mathcal{V}(\mathcal{B})$ the left-hand side of (17a) simplifies to

$$CAv - c_v^* - \varepsilon + Cc \leq c_v^* - c_v^* - \varepsilon = -\varepsilon,$$

where the first inequality comes from the definition of $c_v^*$. Therefore, $(D_\theta, e_\theta)$ is a solution to (17a).

On the other hand, if dynamics (8) with output (3) have relative degree larger than 1, then $CB = 0$, i.e., the policy has no direct impact on output $y$. In that case, (17a) simplifies to $C(Av + c) \leq -\varepsilon$, i.e., $c_v^* \leq -\varepsilon$. ∎

To check the existence of a policy satisfying Theorem 1, we cannot rely solely on the hope that the constraint will enforce itself. Thus, in practice, we need $CB \neq 0$, i.e., the relative degree of system (8) must be 1.

Since affine system (8) is only an approximation of nominal dynamics (2), they do not necessarily have the same relative degree. However, such a preventable discrepancy only hampers practical implementation of Theorem 1 by rendering $\varepsilon$ larger than necessary. Indeed, work [61] discusses how to infer the relative degree of a black box system from first principles. Hence, affine approximation (8) can and should be designed to match the relative degree of nominal dynamics (2) to make $\varepsilon$ as small as possible. This reasoning prompts the following practical consideration.

**Remark 2.** *In practice, to find a policy satisfying Theorem 1, output* (3) *of system* (2) *needs relative degree* 1.

The intuition behind Remark 2 is that the derivative of the constrained states must be actuated to allow immediate corrective actions. A relative degree 1 is also required by CBFs [4] and by numerous works in constrained RL [33, 41, 42, 59] as discussed in Section II-C.

**Remark 3.** *Our foundational theory can be generalized in future work to enforce multiple constraints of high relative degrees by creating an affine buffer for each constraint where the policy dissipate their inertia.*

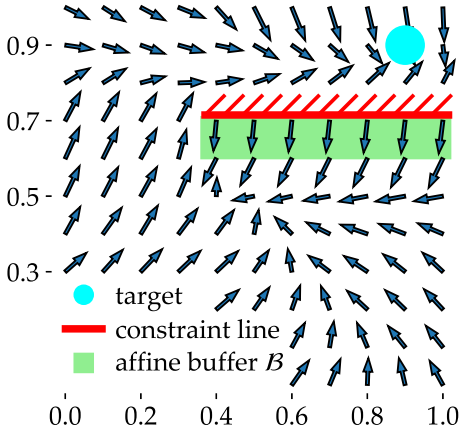



Fig. 4: State space $\mathcal{S}$ with arrows denoting state transitions under POLICEd policy $\mu_\theta$ for linear environment (18). The affine buffer $\mathcal{B}$ (green) pushes states away from the constraint line (red) before heading towards the target (cyan).

## VI. Implementation

In this section, we demonstrate how our method can be implemented in a continuous 2D environment where a point mass moves according to the following discrete-time dynamics

$$s(t + \delta t) = s(t) + a(t)\delta t, \tag{18}$$

with actions $a(t) \in \mathcal{A} := [-1, 1]^2$, states $s(t) \in \mathcal{S} := [0, 1]^2$ and time step $\delta t = 0.1$. We assume dynamics (18) to be a black-box from the controller's perspective. The reward signal is proportional to the distance between the state and a target state located at $(0.9, 0.9)$, as illustrated in Fig. 4.

We consider the line joining states $(0.4, 0.7)$ to $(1., 0.7)$ to form an obstacle. We will guarantee that trajectories do not cross this constraint line from below by placing our buffer right under it, as shown in Fig. 4. Note that a policy trained to reach the target will not cross the constraint line from above as that would move the state away from the target as seen in Fig. 4. Thus, we only need to prevent crossings from below. The constraint is parameterized by $C = [0, 1]$ and $d = 0.7$ in the notations of (3), i.e., $s_2 \leq 0.7$ when $s_1 \geq 0.4$.

The first step of our approach is to estimate the size of our buffer. To do so, we uniformly sample states $s(t) \sim \mathcal{U}(\mathcal{S})$, actions $a(t) \sim \mathcal{U}(\mathcal{A})$, and the corresponding next states $s(t + \delta t)$ from (18) to be stored in a dataset $\mathcal{D}$. With this dataset we can calculate the minimal buffer radius $r$ following condition (16) and we find that for this setting $r = 0.1$, which we can analytically verify with (18).

Following (5) we define buffer $\mathcal{B}$ delimited by vertices $\mathcal{V}(\mathcal{B}) = \big\{(0.4, 0.7), (1, 0.7), (1, 0.6), (0.4, 0.6)\big\}$ which will be provided to our algorithm to learn a POLICEd policy $\mu_\theta$.

The next step is to determine an approximation measure $\varepsilon$ of the system's nonlinearity. Since dynamics (18) are linear, affine approximation (7) is exact with $\varepsilon = 0$.

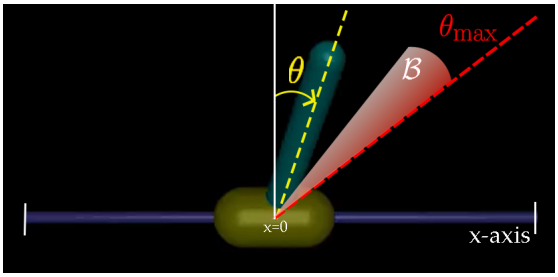


Fig. 5: The inverted pendulum Gym environment [11] annotated with cart position $x$, pendulum angle $\theta$, and buffer $\mathcal{B}$.

Finally, we train the POLICEd policy $\mu_\theta$ to ensure repulsion condition (15) holds, i.e., $s_2(t+\delta t) - s_2(t) \leq 0$ for $s(t) \in \mathcal{V}(\mathcal{B})$. This is illustrated by the arrows pointing down in the affine buffer of Fig. 4. As suggested in [8], once training is completed, we create a copy $\kappa_\theta$ of the POLICEd DNN $\mu_\theta$, where the POLICEd extra-bias $b_{\text{extra}}$ (discussed at the end of Section III) are directly embedded into the standard bias of $\kappa_\theta$. Thus, copy $\kappa_\theta$ is a standard DNN without overhead.

We summarize the POLICEd RL process in Algorithm 1.

**Algorithm 1** POLICEd RL

**Require:** Environment (13), constraint (3), transition dataset $(s, a, s') \sim \mathcal{D} \subset \mathcal{S} \times \mathcal{A} \times \mathcal{S}$

1: Calculate buffer radius $r$ with (16) from dataset $\mathcal{D}$
2: Determine buffer $\mathcal{B}$ and its vertices $\mathcal{V}(\mathcal{B})$ with (5)
3: Sample transitions $(s, a, s') \sim \mathcal{D}$ s.t. $s \in \mathcal{B}$ and use least-square approximation to get $\varepsilon$ from (7)
4: Train the POLICEd RL agent $\mu_\theta$ until repulsion condition (15) holds on the polytopic buffer's vertices $\mathcal{V}(\mathcal{B})$
5: Copy $\mu_\theta$ into a standard DNN $\kappa_\theta$ of identical weights and biases increased by the POLICEd extra-bias $b_{\text{extra}}$ of $\mu_\theta$

**Ensure:** Trajectories $\tau_d^{\mathcal{S}}(s_0, \mu_\theta)$ of (14) starting from safe state $s_0 \in \mathcal{S}_s$ do not leave safe set $\mathcal{S}_s$ and copied policy $\kappa_\theta = \mu_\theta$ has no inference-time overhead.

## VII. Simulations

We will now test our POLICEd RL framework through more challenging realistic simulations to answer the following questions:

- **Q1** Can POLICEd RL produce safer policies than a baseline?
- **Q2** Can POLICEd RL generate policies achieving higher rewards than a baseline, while guaranteeing constraint satisfaction?
- **Q3** Can POLICEd RL be expanded to higher-dimensional systems with realistic dynamics?

### A. Inverted pendulum experiment

We begin by testing POLICEd RL on the OpenAI inverted pendulum environment [11] which uses the MuJoCo physics engine [53]. The inverted pendulum environment has a 4-dimensional observation space with cart position $x$, pendulum angle $\theta$, linear cart velocity $\dot{x}$, and angular velocity $\dot{\theta}$ as illustrated in Fig. 5. The action space is the force applied to the cart $a$ proportional to $\ddot{x}$.

We investigate different safety constraints to prevent the pendulum from falling. First, we consider a constraint of relative degree 1 where we want to enforce $\dot{\theta}(t) \leq 0$ for $\theta(t) \in [0.1, 0.2]\,rad$ to push $\theta$ away from its upper limit $\theta_{max} = 0.2\,rad$. With state $s(t) = \big(x(t), \theta(t), \dot{x}(t), \dot{\theta}(t)\big)$, the constraint is $Cs(t) \leq d$ with $d = 0$ and $C = \begin{bmatrix}0 & 0 & 0 & 1\end{bmatrix}$. We now need to create a buffer wide enough such that it cannot be 'jumped' over by the robot in a single time-step. Following (5), we build a buffer of width $r$ as $\mathcal{B} = \big\{(x, \theta, \dot{x}, \dot{\theta}) : x \in [-0.9, 0.9],\ \theta \in [0.1, 0.2],\ \dot{x} \in [-1, 1],\ \dot{\theta} \in [-r, 0]\big\}$. Buffer $\mathcal{B}$ stays clear off $x = \pm 1$ since these locations cause a large state discontinuity preventing stabilization. Following step 1 of Algorithm 1, we uniformly sample states $s \sim \mathcal{U}(\mathcal{S})$, actions $a \sim \mathcal{U}(\mathcal{A})$, corresponding next state $s'$ and we use (16) to compute $r \approx 1.03$ for action magnitudes $|a| \leq 1$. To guarantee $\dot{\theta} \leq 0$, Theorem 1 leads us to enforce $\ddot{\theta} \leq 0$ for all states in buffer $\mathcal{B}$.

We can now compute an approximation measure $\varepsilon$ from (7). We uniformly sample states $s \sim \mathcal{U}(\mathcal{B})$, actions $a \sim \mathcal{U}(\mathcal{A})$, we get the corresponding next state $s'$ and approximate $\dot{s} \approx (s' - s)/\delta t$ with $\delta t = 0.02\,s$ which is the environment time-step. A least-square fit following (7) yields a value $\varepsilon \approx 0.7$.

Note that training to satisfy a constraint is often adversarial to accomplishing the task, and thus often causes training instability. Throughout our development process, we discovered techniques to greatly improve training time and sample complexity, which are further discussed in Appendix D. For example, with the inverted pendulum scenario, we train policy $\mu_\theta$ by iterating over two phases. The first phase is a standard RL training where the initial state is regularly reset either around the origin or inside the affine buffer. Once a reward threshold is met, the *constraint training* phase starts. The state is iteratively reset to all the vertices of buffer $\mathcal{B}$ and the trajectory is propagated for a single time step to evaluate whether repulsion condition (15) holds. Only the experiences where (15) is *not* verified are added to the replay memory with negative rewards in order to promote the respect of (15). After several rounds of these updates, the first training phase resumes and the process begins anew until condition (15) holds everywhere on the vertices of $\mathcal{B}$ and the maximal reward is achieved. We summarize this training process in Algorithm 2.

We use Proximal Policy Optimization (PPO) [50] to learn both a baseline and a POLICEd policy. The baseline is a standard PPO policy that does not have the enforced affine buffer $\mathcal{B}$ of the POLICEd policy. They both follow the same training procedure described above in the same environment where they receive identical penalties for constraint violations. Each episode has a maximal length of 1000 time steps with a reward of 1 if the pole is upright and 0 otherwise. The reward curves of Fig. 6 show that both methods achieve maximal reward.

During training we measure the proportion of buffer $\mathcal{B}$ where repulsion condition (15) is satisfied and report these proportions in Fig. 7. Since the POLICEd policy achieves maximal reward while completely satisfying repulsion condition (15) at episode 1180 it stops training, whereas Fig. 7

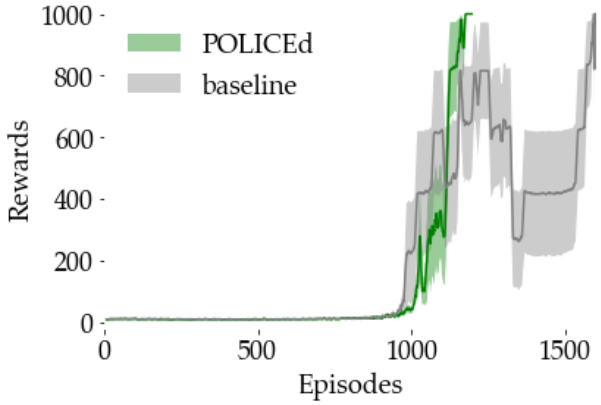



Fig. 6: Reward curves for the inverted pendulum (max=1000). The solid lines correspond to the average and the shaded regions to the 95% confidence interval over 5 runs.

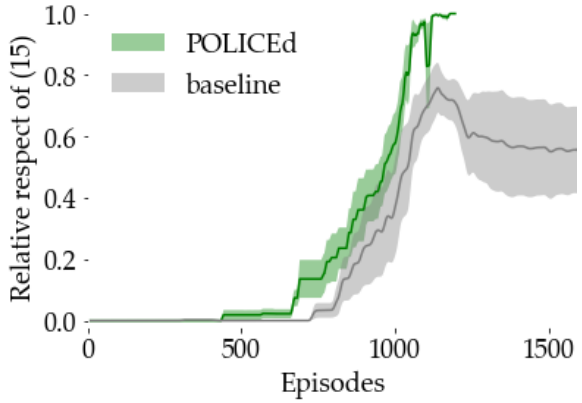
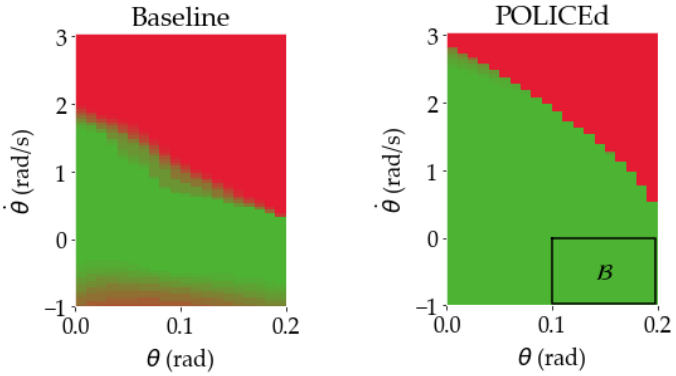



Fig. 7: Relative portion of buffer $\mathcal{B}$ where repulsion condition (15) is respected. The baseline never succeeds in entirely enforcing (15) hence allowing possible constraint violations. The solid lines correspond to the average and the shaded regions to the 95% confidence interval over 5 runs.

shows that the baseline never succeeds in enforcing (15) over the entire buffer. Moreover, the constraint training phase causes a large drop in the baseline rewards as seen in Fig. 6. Our POLICEd policy guarantees the satisfaction of constraint $\dot{\theta} \leq 0$ by enforcing repulsion condition (15), i.e., $\ddot{\theta} \leq 0$. Consequently, it guarantees that trajectories starting from $\dot{\theta}(0) < 0$ will never reach any state where $\dot{\theta}(t) \geq 0$.

We will now empirically evaluate the performance of POLICEd RL at ensuring satisfaction of a constraint of relative degree 2. Although our theory cannot provide any safety guarantees, we will show that POLICEd RL performs better and is safer than the baseline.

We want the pendulum to maintain $|\theta| \leq \theta_{\max} = 0.2\,rad$ (See Fig. 5). This position constraint is of relative degree 2 since the action (pushing the cart) only impacts directly the angular acceleration $\ddot{\theta}$. Our empirical evaluation consists in resetting the state of the inverted pendulum at a variety of initial conditions $\left(\theta, \dot{\theta}\right)$, and seeing how often policies fail to maintain $|\theta| \leq 0.2\,rad$. Some initial conditions *cannot be stabilized* since the controller does not have direct control action on $\theta$, only on $\dot{\theta}$. Indeed $\theta(\delta t) = \theta_0 + \delta t\,\dot{\theta}_0 > 0.2\,rad$, if $\theta_0 = 0.2\,rad$ and $\dot{\theta}_0 > 0$ not matter the control action. We present our results in Fig. 8 where we can see that the POLICEd policy stabilizes a span of initial states much larger than both buffer $\mathcal{B}$ and the ones stabilized by the baseline.




(a) Baseline (b) POLICEd

Fig. 8: Success rates for the policies to maintain $|\theta| \leq 0.2\,rad$ for the inverted pendulum given an initial state $\left(0, \theta, 0, \dot{\theta}\right)$. Green: stabilized. Red: failure. The black box shows buffer $\mathcal{B}$ where the POLICEd policy guarantees $\ddot{\theta} \leq 0$. Position constraint $|\theta| \leq 0.2\,rad$ is of relative degree 2. Some initial conditions pictured here *cannot be stabilized* since the controller does not have direct control action on $\theta$, only on $\dot{\theta}$. However, we see that our POLICEd policy maintains safety in a larger region of the state space than the baseline.

Then, POLICED RL has shown to be empirically effective in increasing the set of stabilizable initial conditions even with constraints of high relative degree. Our results indicate that both **Q1** and **Q2** are answered positively as POLICEd RL produces safer policies achieving higher rewards than our baseline in the inverted pendulum environment.

### B. Robotic arm

We further showcase our method on a robotic arm with 7 degrees of freedom. To implement our approach, we rely on the high-fidelity MuJoCo simulator [53]. We developed a custom environment where a KUKA robotic arm aims to reach a 3D target with its end-effector while avoiding an unsafe region as illustrated in Fig. 9. More specifically, we define the state of the environment $s \in \mathbb{R}^{10}$ as the arm's joint angles $s_{joints} \in \mathbb{R}^7$ and the X-Y-Z coordinates of the end-effector $s_{end} \in \mathbb{R}^3$. At each timestep, the policy provides a change in joint angles $a \in \mathbb{R}^7$. The target is kept constant across all episodes. At the start of each episode, the starting state of the arm is uniformly sampled from the joint space.

At each timestep, the robotic arm is assigned a reward function $R(s, a)$ which is composed of four terms: the straight line distance $R_d(s)$ between the end-effector and the target, a bonus $R_b(s, a)$ for reaching the target, a penalty $R_{\text{inf}}(a)$ for generating an infeasible joint state, and a penalty $R_{\text{unsf}}(s, a)$ for violating the constraint.

We perform an ablation study to showcase the necessity

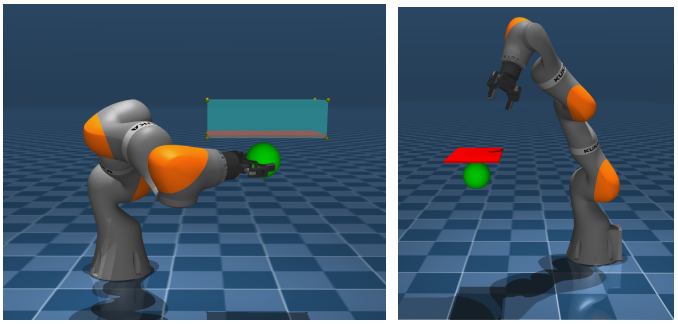


Fig. 9: Robotic arms tasked to bring their end-effectors to the target (green) while avoiding the constraint surface (red). The POLICEd method uses a buffer region (cyan). The arms are shown in two random configurations.

of our POLICEd method to ensure both high rewards and constraint satisfaction. To begin, we learn a POLICEd policy using Twin Delayed DDPG (TD3) [22], an improved version of Deep Deterministic Policy Gradient (DDPG) [36]. After training each model, we deployed the fully-trained policy onto 500 episodes on our Safe Arm Task which is illustrated in Fig. 9. From these episodes we collected a series of metrics which we summarize in Table I:

1) *Completion %:* The percentage of episodes where the policy reaches the target to evaluate task completion rate.
2) *Completion % w/o Violation:* The percentage of episodes where the policy reaches the target *without* violating the constraint, to evaluate its safety capabilities.
3) *Average Reward:* The average reward earned to evaluate how efficiently the policy completes the task.
4) *Average Constraint Satisfaction:* The percentage of episodes where the constraint is never violated to evaluate the safety capabilities of the policy.

We proceeded with the ablation study by training and evaluating variations of our initial policy where we either removed our POLICE module or the reward penalties. We began by training a traditional TD3 policy without our POLICE module and evaluating it as described. We then switched to a reward function without $R_{\text{unsf}}$ and trained a TD3 policy without both the POLICE module and without the constraint violation penalties. We also trained a POLICEd policy without the penalty term in order to showcase how reward shaping is necessary to tune the affine region of the policy to avoid the constraint region.

While we believe ours is the first paper to provably enforce hard constraints with black-box environments, safe RL has produced remarkable works in soft constraints and learned safety certificate methods. As such, we also compare our approach with the soft constraint method Constrained Policy Optimization (CPO) [1], as well as the learned control barrier function approach PPO-Barrier of [64].

As seen in Table I, our POLICEd policy is the only algorithm to *guarantee* constraint satisfaction. The soft constraint CPO [1] and the learned safety certificate PPO-Barrier [64] baselines provide better constraint adherence than standard RL algorithms like TD3, but still fall far short of guaranteed satisfaction. In comparison, our POLICEd approach has the highest task completion percentage without violations by a $40\%$ margin, while also being the highest reward earning policy by nearly a margin of 3 times.

Through our ablation study, we confirm that our POLICEd approach is necessary for constraint satisfaction, as seen by the poor average constraint satisfaction by the TD3 and TD3 trained without penalty policies. Furthermore, it follows intuitively that the constraint violation penalties guide the policy to avoid the region, a concept extensively studied in soft constraint works [1, 27]. As expected, training without penalizing constraint violations is vastly detrimental to the performance of both the TD3 and POLICEd policies. In the POLICEd case, the reward shaping is necessary for the policy to appropriately tune the affine region to avoid the constraint. We can now answer questions **Q1**, **Q2**, and **Q3** as the POLICEd policy achieves highest average reward while *guaranteeing constraint satisfaction* even on this relatively high-dimensional, high-fidelity robotic arm environment.

We additionally observed that our POLICEd approach exhibited significantly greater sample efficiency compared to our baseline methods. The POLICEd policy converged within 4000 episodes, each consisting of 100 steps. In contrast, CPO frequently failed to converge even after 20,000 episodes of the same length. While PPO-Barrier achieved convergence within 200 iterations, these iterations encompassed numerous episodes of uncapped length, resulting in nearly double the number of environment samples required compared to POLICEd. For additional details see Appendix E-B1.

## VIII. Conclusion

**Summary.** We proposed *POLICEd RL*, a novel algorithm explicitly designed to enforce hard safety constraints for a black-box robot in closed loop with a RL policy. Our key insight was to build a repulsive buffer around the unsafe area with a locally affine learned policy to guarantee that trajectories never leave the safe set. Our experiments showed that POLICEd RL can enforce hard constraints in high-dimensional, high-fidelity robotics tasks while significantly outperforming existing methods.

**Limitations.** With Proposition 1, we can verify whether there exists a safe POLICEd policy. However, as in standard RL, there are no guarantees that the training will converge to this safe policy. Moreover, in high dimensional environments, the exponential number of vertices of buffer region $\mathcal{B}$ $\left(2^n \text{ for a box in dimension } n\right)$ will become computationally prohibitive. However, once trained, a POLICEd DNN is just as fast as a standard unconstrained DNN [8]. Finally, we would like to point out that if the estimate of the approximation measure $\varepsilon$ is too low, then the safety guarantees of Theorem 1 and Corollary 1 will not hold. We would like to investigate how an upper bound of $\varepsilon$ can be estimated for general high-dimensional settings.

**Future Directions.** We are excited by our findings and believe our method is only the first step towards enforcing hard constraints on RL policy. For future work, we plan to

| Models | Completion % (↑) | Completion % w/o violation (↑) | Average reward ±95% CI (↑) | Average % constraint satisfaction ±95% CI (↑) |
|---|---|---|---|---|
| TD3 trained and evaluated w/o penalty | 100 | – | $-11.07 \pm 0.59$ | $69.2 \pm 4.1$ |
| POLICEd (ours) | 93.4 | **93.4** | $\mathbf{-16.22 \pm 0.68}$ | $\mathbf{100 \pm 0.0}$ |
| TD3 | 75.8 | 12.0 | $-45.20 \pm 3.23$ | $28.4 \pm 3.9$ |
| CPO | 2.0 | 2.0 | $-96.71 \pm 3.45$ | $89.9 \pm 2.7$ |
| PPO-Barrier | **100** | 86.2 | $-41.26 \pm -2.30$ | $86.2 \pm 3.0$ |
| POLICEd trained w/o penalty | 48.0 | 41.6 | $-70.09 \pm 1.22$ | $41.6 \pm 4.3$ |
| TD3 trained w/o penalty | 99.8 | 48.8 | $-45.69 \pm 16.61$ | $53.4 \pm 4.4$ |

TABLE I: Metrics comparison for different methods based on a 500 episode deployment with the fully-trained policies on the safe arm task illustrated in Fig. 9. The top row (gray) provides an upper bound on the completion rate and maximal reward achievable as TD3 is evaluated without penalties for constraint violation (i.e. without $R_{\text{unsf}}$). We compare our POLICEd method against the soft-constraint baseline CPO [1] and the learned safety certificate baseline PPO-Barrier [64]. We also report the metrics for TD3 trained with and without the penalty as well as our POLICEd method trained without penalty as part of our ablation study. The bold numbers denote the highest values achieved when constraint violations are appropriately penalized. The completion task only assess whether the target is eventually reached, even if the constraint is not properly respected. For all metrics higher is better (↑).

investigate the case of higher relative-degree constraints by taking ideas from the works that extended CBFs to higher-relative degrees [40, 46, 62, 63]. We would like to further consider enforcing multiple constraints simultaneously. This extension would require minimal changes to our theory but would mostly involve extending the POLICE algorithm [8] to enforce several affine regions instead of just one as initially designed. Another direction of research could be to guarantee safety during training.

## Acknowledgments

This work is supported by the National Science Foundation, under grants ECCS-2145134, CAREER Award, CNS-2423130, and CCF-2423131.

## References

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

[2] Akshay Agrawal, Brandon Amos, Shane Barratt, Stephen Boyd, Steven Diamond, and J Zico Kolter. Differentiable convex optimization layers. *Advances in Neural Information Processing Systems*, 32, 2019.

[3] Eitan Altman. *Constrained Markov decision processes*. Routledge, 2021.

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

[5] Brandon Amos and J Zico Kolter. OptNet: Differentiable optimization as a layer in neural networks. In *International Conference on Machine Learning*, pages 136–145. PMLR, 2017.

[6] Anil Aswani, Humberto Gonzalez, S Shankar Sastry, and Claire Tomlin. Provably safe and robust learning-based model predictive control. *Automatica*, 49(5):1216–1226, 2013.

[7] Randall Balestriero and Richard Baraniuk. A spline theory of deep learning. In *35th International Conference on Machine Learning*, volume 80, pages 374–383. PMLR, 2018.

[8] Randall Balestriero and Yann LeCun. POLICE: Provably optimal linear constraint enforcement for deep neural networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 1–5. IEEE, 2023.

[9] Alberto Bemporad and Manfred Morari. Robust model predictive control: A survey. In *Robustness in identification and control*, pages 207–226. Springer, 2007.

[10] Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Dębiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, et al. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*, 2019.

[11] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. OpenAI Gym. *arXiv preprint arXiv:1606.01540*, 2016.

[12] Lukas Brunke, Melissa Greeff, Adam W Hall, Zhaocong Yuan, Siqi Zhou, Jacopo Panerati, and Angela P Schoellig. Safe learning in robotics: From learning-based control to safe reinforcement learning. *Annual Review of Control, Robotics, and Autonomous Systems*, 5:411 – 444, 2022.

[13] Ya-Chien Chang, Nima Roohi, and Sicun Gao. Neural Lyapunov control. *Advances in Neural Information Processing Systems*, 32, 2019.

[14] Richard Cheng, Gábor Orosz, Richard M Murray, and Joel W Burdick. End-to-end safe reinforcement learning

through barrier functions for safety-critical continuous control tasks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3387–3395, 2019.

[15] Frank J Christophersen. *Optimal Control of Constrained Piecewise Affine Systems*. Springer, 2007.

[16] Gal Dalal, Krishnamurthy Dvijotham, Matej Vecerik, Todd Hester, Cosmin Paduraru, and Yuval Tassa. Safe exploration in continuous action spaces. *arXiv preprint arXiv:1801.08757*, 2018.

[17] Ersin Daş and Richard M Murray. Robust safe control synthesis with disturbance observer-based control barrier functions. In *61st Conference on Decision and Control*, pages 5566–5573. IEEE, 2022.

[18] Federico Di Palma and Lalo Magni. A multi-model structure for model predictive control. *Annual reviews in control*, 28(1):47–52, 2004.

[19] Priya L Donti, David Rolnick, and J Zico Kolter. DC3: A learning method for optimization with hard constraints. In *International Conference on Learning Representations*, 2020.

[20] Gabriel Dulac-Arnold, Nir Levine, Daniel J Mankowitz, Jerry Li, Cosmin Paduraru, Sven Gowal, and Todd Hester. Challenges of real-world reinforcement learning: Definitions, benchmarks and analysis. *Machine Learning*, 110(9):2419 – 2468, 2021.

[21] Thomas Frerix, Matthias Nießner, and Daniel Cremers. Homogeneous linear inequality constraints for neural network activations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 748 – 749, 2020.

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

[23] Sicun Gao, Jeremy Avigad, and Edmund M Clarke. $\delta$-complete decision procedures for satisfiability over the reals. In *International Joint Conference on Automated Reasoning*, pages 286–300. Springer, 2012.

[24] A. Girard and S. Martin. Synthesis for constrained nonlinear systems using hybridization and robust controllers on simplices. *IEEE Transactions on Automatic Control*, 57(4), 2012.

[25] Gene Golub. Numerical methods for solving linear least squares problems. *Numerische Mathematik*, 7:206–216, 1965.

[26] Branko Grünbaum, Volker Kaibel, Victor Klee, and Gtinter M. Ziegler. *Convex Polytopes*. Springer Science & Business Media, 2003.

[27] Shangding Gu, Long Yang, Yali Du, Guang Chen, Florian Walter, Jun Wang, Yaodong Yang, and Alois Knoll. A review of safe reinforcement learning: Methods, theory and applications. *arXiv preprint arXiv:2205.10330*, 2022.

[28] L. Habets and J. H. Van Schuppen. A control problem for affine dynamical systems on a full-dimensional polytope. *Automatica*, 40:21–35, 2004.

[29] Lukas Hewing, Kim P Wabersich, Marcel Menner, and Melanie N Zeilinger. Learning-based model predictive control: Toward safe learning in control. *Annual Review of Control, Robotics, and Autonomous Systems*, 3:269–296, 2020.

[30] Krishna C Kalagarla, Rahul Jain, and Pierluigi Nuzzo. A sample-efficient algorithm for episodic finite-horizon MDP with constraints. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 8030–8037, 2021.

[31] M. Kloetzer and C. Belta. A fully automated framework for control of linear systems from temporal logic specifications. *IEEE Transactions on Automatic Control*, 53 (1), 2008.

[32] Anthony W Knapp. *Basic Real Analysis*. Springer Science & Business Media, 2005.

[33] Craig Knuth, Glen Chou, Necmiye Ozay, and Dmitry Berenson. Planning with learned dynamics: Probabilistic guarantees on safety and reachability via Lipschitz constants. *IEEE Robotics and Automation Letters*, 6(3): 5129–5136, 2021.

[34] Mykel J Kochenderfer, Jessica E Holland, and James P Chryssanthacopoulos. Next generation airborne collision avoidance system. *Lincoln Laboratory Journal*, 19(1): 17–33, 2012.

[35] Andrei V Konstantinov and Lev V Utkin. A new computationally simple approach for implementing neural networks with output hard constraints. *arXiv preprint arXiv:2307.10459*, 2023.

[36] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. In *International Conference on Learning Representations*, 2016.

[37] Zuxin Liu, Zijian Guo, Yihang Yao, Zhepeng Cen, Wenhao Yu, Tingnan Zhang, and Ding Zhao. Constrained decision transformer for offline safe reinforcement learning. *arXiv preprint arXiv:2302.07351*, 2023.

[38] Lennart Ljung. System identification. In *Signal analysis and prediction*, pages 163–173. Springer, 1998.

[39] Matthias Lorenzen, Fabrizio Dabbene, Roberto Tempo, and Frank Allgöwer. Stochastic MPC with offline uncertainty sampling. *Automatica*, 81:176–183, 2017.

[40] Haitong Ma, Jianyu Chen, Shengbo Eben, Ziyu Lin, Yang Guan, Yangang Ren, and Sifa Zheng. Model-based constrained reinforcement learning using generalized control barrier function. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4552–4559. IEEE, 2021.

[41] Haitong Ma, Changliu Liu, Shengbo Eben Li, Sifa Zheng, and Jianyu Chen. Joint synthesis of safety certificate and safe control policy using constrained reinforcement learning. In *Learning for Dynamics and Control Conference*, pages 97–109. PMLR, 2022.

[42] Yue Meng, Sai Vemprala, Rogerio Bonatti, Chuchu Fan, and Ashish Kapoor. ConBaT: Control barrier

transformer for safe policy learning. *arXiv preprint arXiv:2303.04212*, 2023.

[43] Ali Mesbah. Stochastic model predictive control: An overview and perspectives for future research. *IEEE Control Systems Magazine*, 36(6):30–44, 2016.

[44] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[45] Khan Muhammad, Amin Ullah, Jaime Lloret, Javier Del Ser, and Victor Hugo C de Albuquerque. Deep learning for safe autonomous driving: Current challenges and future directions. *IEEE Transactions on Intelligent Transportation Systems*, 22(7):4316–4336, 2020.

[46] Quan Nguyen and Koushil Sreenath. Exponential control barrier functions for enforcing high relative-degree safety-critical constraints. In *2016 American Control Conference*, pages 322 – 328. IEEE, 2016.

[47] Tu-Hoa Pham, Giovanni De Magistris, and Ryuki Tachibana. Optlayer - practical constrained optimization for deep reinforcement learning in the real world. In *2018 IEEE International Conference on Robotics and Automation*, pages 6236–6243. IEEE, 2018.

[48] Zengyi Qin, Dawei Sun, and Chuchu Fan. Sablas: Learning safe control for black-box dynamical systems. *IEEE Robotics and Automation Letters*, 7(2):1928–1935, 2022.

[49] Nicholas Rober, Sydney M Katz, Chelsea Sidrane, Esen Yel, Michael Everett, Mykel J Kochenderfer, and Jonathan P How. Backward reachability analysis of neural feedback loops: Techniques for linear and nonlinear systems. *IEEE Open Journal of Control Systems*, 2:108–124, 2023.

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

[51] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science*, 362(6419):1140–1144, 2018.

[52] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

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

[54] Jesus Tordesillas, Jonathan P How, and Marco Hutter. RAYEN: Imposition of hard convex constraints on neural networks. *arXiv preprint arXiv:2307.08336*, 2023.

[55] Joseph A Vincent and Mac Schwager. Reachable polyhedral marching (RPM): A safety verification algorithm for robotic systems with deep neural network components. In *IEEE International Conference on Robotics and Automation*, pages 9029–9035. IEEE, 2021.

[56] Kim P Wabersich, Lukas Hewing, Andrea Carron, and Melanie N Zeilinger. Probabilistic model predictive safety certification for learning-based control. *IEEE Transactions on Automatic Control*, 67(1):176–188, 2021.

[57] Kim Peter Wabersich and Melanie N Zeilinger. A predictive safety filter for learning-based control of constrained nonlinear dynamical systems. *Automatica*, 129:109597, 2021.

[58] Le Yi Wang and Ji-Feng Zhang. Fundamental limitations and differences of robust and adaptive control. In *Proceedings of the 2001 American Control Conference*, volume 6, pages 4802–4807. IEEE, 2001.

[59] Yixuan Wang, Simon Sinong Zhan, Ruochen Jiao, Zhilu Wang, Wanxin Jin, Zhuoran Yang, Zhaoran Wang, Chao Huang, and Qi Zhu. Enforcing hard constraints with soft barriers: Safe reinforcement learning in unknown stochastic environments. In *International Conference on Machine Learning*, pages 36593–36604. PMLR, 2023.

[60] Lily Weng, Huan Zhang, Hongge Chen, Zhao Song, Cho-Jui Hsieh, Luca Daniel, Duane Boning, and Inderjit Dhillon. Towards fast computation of certified robustness for ReLU networks. In *International Conference on Machine Learning*, pages 5276–5285. PMLR, 2018.

[61] Tyler Westenbroek, David Fridovich-Keil, Eric Mazumdar, Shreyas Arora, Valmik Prabhu, S Shankar Sastry, and Claire J Tomlin. Feedback linearization for uncertain systems via reinforcement learning. In *2020 IEEE International Conference on Robotics and Automation*, pages 1364 – 1371. IEEE, 2020.

[62] Wei Xiao and Calin Belta. Control barrier functions for systems with high relative degree. In *58th Conference on Decision and Control*, pages 474 – 479. IEEE, 2019.

[63] Wei Xiao, Tsun-Hsuan Wang, Ramin Hasani, Makram Chahine, Alexander Amini, Xiao Li, and Daniela Rus. BarrierNet: Differentiable control barrier functions for learning of safe robot control. *IEEE Transactions on Robotics*, 39(3):2289 – 2307, 2023.

[64] Yujie Yang, Yuxuan Jiang, Yichen Liu, Jianyu Chen, and Shengbo Eben Li. Model-free safe reinforcement learning through neural barrier certificate. *IEEE Robotics and Automation Letters*, 8(3):1295–1302, 2023.

[65] Huan Zhang, Tsui-Wei Weng, Pin-Yu Chen, Cho-Jui Hsieh, and Luca Daniel. Efficient neural network robustness certification with general activation functions. *Advances in Neural Information Processing Systems*, 31, 2018.

[66] Huan Zhang, Pengchuan Zhang, and Cho-Jui Hsieh. RecurJac: An efficient recursive algorithm for bounding Jacobian matrix of neural networks and its applications. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 5757–5764, 2019.

[67] Weiye Zhao, Tairan He, and Changliu Liu. Model-free safe control for zero-violation reinforcement learning. In *5th Annual Conference on Robot Learning*, pages 784–

793. PMLR, 2022.

[68] Weiye Zhao, Rui Chen, Yifan Sun, Tianhao Wei, and Changliu Liu. State-wise constrained policy optimization. *arXiv preprint arXiv:2306.12594*, 2023.

[69] Weiye Zhao, Tairan He, and Changliu Liu. Probabilistic safeguard for reinforcement learning using safety index guided Gaussian process models. In *Learning for Dynamics and Control Conference*, pages 783–796. PMLR, 2023.

[70] Weiye Zhao, Yifan Sun, Feihan Li, Rui Chen, Tianhao Wei, and Changliu Liu. Learn with imagination: Safe set guided state-wise constrained policy optimization. *arXiv preprint arXiv:2308.13140*, 2023.

## APPENDIX A
## PROOF OF LEMMA 1

*Proof:* We want to prove that buffer $\mathcal{B}$ of (5) is a polytope. By definition (5), buffer $\mathcal{B}$ can be written as $\mathcal{B} = C^{-1}\big([d-r,d]\big) \cap \mathcal{S}$, where $C^{-1}\big([d-r,d]\big) := \{s : Cs \in [d-r,d]\}$ denotes the inverse image of the interval $[d-r,d]$. The inverse image of a set is always defined, even if matrix $C$ is not invertible. Therefore, $\mathcal{B}$ is the intersection of affine variety $C^{-1}\big([d-r,d]\big)$ and polytope $\mathcal{S}$ and hence $\mathcal{B}$ is a polytope according to result 4 of section 3.1 of [26]. ∎

## APPENDIX B
## PROOF OF LEMMA 2

*Proof:* We want to prove that any $\varepsilon$ sufficiently large is an approximation measure in the sense of Definition 1. We will first determine the infimum of all approximation measures before proving that any $\varepsilon$ larger than this infimum is an approximation measure.

For every matrices $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$, and $c \in \mathbb{R}^n$ introduce

$$\varepsilon^*_{ABc} := \min\left\{\varepsilon : \big|Cf(s,a) - C(As+Ba+c)\big| \leq \varepsilon, \atop \text{for all } a \in \mathcal{A}, \text{ and all } s \in \mathcal{B}\right\}.$$

Note that the minimum in $\varepsilon^*_{ABc}$ exists since the function to minimize is continuous ($f$ is assumed continuous at (2)) and sets $\mathcal{A}$ and $\mathcal{B}$ are compact. Then, all $\varepsilon^*_{ABc}$ are approximation measures as they satisfy Definition 1. Define

$$\varepsilon^* := \inf\left\{\varepsilon^*_{ABc} : A \in \mathbb{R}^{n\times n},\ B \in \mathbb{R}^{n\times m},\ c \in \mathbb{R}^n\right\}.$$

We will now show that any $\varepsilon$ larger than $\varepsilon^*$ is in fact an approximation measure. Let $\varepsilon > \varepsilon^*$. Since $\varepsilon^*$ is the infimum of all approximation measures following Lemma 3, there exists an approximation measure $\alpha$ such that $\varepsilon^* \leq \alpha < \varepsilon$. Since $\alpha$ is an approximation measure, there exists a triplet $(A,B,c)$ for which (7) holds with upper bound $\alpha$. Note that (7) also holds for this same triplet but with upper bound $\varepsilon$ since $\varepsilon > \alpha$. Then $\varepsilon$ is an approximation measure. ∎

**Lemma 3.** *The infimum of all approximation measures is $\varepsilon^*$.*

*Proof:* To show that $\varepsilon^*$ is the infimum of all approximation measures, we need to show that it is their largest lower bound. If $\varepsilon$ is an approximation measure, then it has a triplet $(A,B,c)$ satisfying (7). By definition of $\varepsilon^*_{ABc}$ and $\varepsilon^*$, we then have $\varepsilon \geq \varepsilon^*_{ABc} \geq \varepsilon^*$. Thus, $\varepsilon^*$ is a lower bound to all approximation measures.

Let $\alpha$ be a lower bound to all approximation measures. Since all the $\varepsilon^*_{ABc}$ are approximation measures, $\alpha \leq \varepsilon^*_{ABc}$ for all $A,B,c$, i.e., $\alpha \leq \varepsilon^*$ by definition of $\varepsilon^*$. Therefore, $\varepsilon^*$ is the largest lower bound of all approximation measures. ∎

## APPENDIX C
## PROOF OF COROLLARY 1

*Proof:* As in the proof of Theorem 1, we take advantage of the linearity of dynamics (8) to extend (15) from the vertices of $\mathcal{B}$ to the whole set $\mathcal{B}$. Note that for any $s_t \in \mathcal{V}(\mathcal{B})$, combining repulsion condition (15) with (7) yields

$$\begin{aligned} &C\big(As_t + B\mu_\theta(s_t) + c\big) \\ &\leq \big|C\big(As_t + B\mu_\theta(s_t) + c\big) - Cf\big(s_t, \mu_\theta(s_t)\big)\big| \\ &\qquad + Cf\big(s_t, \mu_\theta(s_t)\big) \\ &\leq \varepsilon + \frac{1}{\delta t} C\big(s_{t+1} - s_t\big) \leq \varepsilon - 2\varepsilon\delta t \frac{1}{\delta t} \leq -\varepsilon. \end{aligned}$$

As in the proof of Theorem 1, we use the convexity of polytope $\mathcal{B}$ of vertices $\mathcal{V}(\mathcal{B})$ and the linearity of approximation (8) to obtain

$$C\big(As + B\mu_\theta(s) + c\big) \leq -\varepsilon, \quad \text{for all } s \in \mathcal{B}. \tag{19}$$

We can now revert this inequality to the discrete dynamics. For $s_t \in \mathcal{B}$,

$$\begin{aligned} &C\frac{s_{t+1} - s_t}{\delta t} \\ &\leq \left|C\frac{s_{t+1} - s_t}{\delta t} - C\big(As_t + B\mu_\theta(s_t) + c\big)\right| \\ &\qquad + C\big(As_t + B\mu_\theta(s_t) + c\big) \\ &\leq \big|Cf\big(s_t, \mu_\theta(s_t)\big) - C\big(As_t + B\mu_\theta(s_t) + c\big)\big| - \varepsilon \\ &\leq \varepsilon - \varepsilon \leq 0. \end{aligned} \tag{20}$$

Note that the first inequality follows from the triangular inequality, the second from the definition of $s_{t+1}$ and (19), and the third inequality stems from (7). We will now show that (20) prevents all trajectories $\tau_d^{\mathcal{S}}(s_0, \mu_\theta)$ from exiting safe set $\mathcal{S}_s$ when $s_0 \in \mathcal{S}_s$.

We assume for contradiction purposes that trajectory $\tau_d^{\mathcal{S}}(s_0, \mu_\theta) \notin \mathcal{S}_s$ for some $s_0 \in \mathcal{S}_s$. Since $\tau_d^{\mathcal{S}}(s_0, \mu_\theta) \in \mathcal{S}$, there exists some $t \in \mathbb{N}$ such that $s_t \in \mathcal{S}_s$ and $s_{t+1} \in \mathcal{S} \backslash \mathcal{S}_s$. Thus, $Cs_t < d$ and $Cs_{t+1} \geq d$. Now (16) yields

$$C(s_{t+1} - s_t) \leq r,$$

since $s_t \in \mathcal{S}_s$ and $\mu_\theta(s_t) \in \mathcal{A}$. Then, $Cs_t \geq Cs_{t+1} - r \geq d - r$. Thus, $s_t \in \mathcal{B}$.

Since $Cs_t < d$ and $Cs_{t+1} \geq d$, we have

$$C\frac{s_{t+1} - s_t}{\delta t} > 0,$$

which contradicts (20). Therefore, all trajectories $\tau_d^{\mathcal{S}}$ starting in safe set $\mathcal{S}_s$ remain in $\mathcal{S}_s$. ∎

## APPENDIX D
## POLICED RL TRAINING TIPS

During our numerous implementations we learned a few tips to help any POLICEd actor to learn a safe policy faster. However, as in standard RL training, there are no guarantees that training will converge to a feasible policy.

To ensure the admissibility of policy $\mu_\theta$, i.e., $\mu_\theta(s) \in \mathcal{A}$ for all $s \in \mathcal{S}$, the classical approach would be to add a final layer to the neural network constituted of a bounded function like the hyperbolic tangent or sigmoid. However, these functions are not piecewise affine, which would invalidate the assumptions of [8]. A clipping of the output would conserve the

piecewise affine character of $\mu_\theta$ but would modify the partition $\mathcal{R}$ of (1), hence preventing the POLICE algorithm to guarantee the affine character of $\mu_\theta$ on buffer $\mathcal{B}$. Instead, we addressed this issue by adding a large penalty to the reward function when $\mu_\theta(s) \notin \mathcal{A}$. This approach has been very successful in all our experiments.

Another helpful trick to improve the training of a POLICEd network comes from manual curriculum learning. We start the training with an affine buffer $\mathcal{B}$ of size zero. Once a reward threshold is met, we iteratively increase the size of the buffer until it reaches the specified size. This minimizes the impact of POLICE in slowing the learning process.

Similarly, we noticed that resetting the initial state of some trajectories inside the buffer helped the POLICEd policy learn repulsion condition (9) or (15). We would typically reset 1 in 10 initial states inside the buffer during training.

It is also useful to note that for an actor-critic algorithm, only the actor DNN is POLICEd, and the critic is not modified.

We tested several different approaches for the environment behavior with respect to constraint violations.

- *Penalized Constraints*: when the agent violates the constraint, it is allowed to keep going but incurs a penalty.
- *Terminal Constraints*: when the agent violates the constraint, the episode ends with a penalty.
- *Bouncing Constraints*: when the agent tries to step into the constraint, it incurs a penalty and its state remains at its last constraint-abiding value.

The inverted pendulum environment of Section VII-A implements terminal constraints, while the linear environment of Section VI and the robotic arm environment of Section VII-B rely on bouncing constraints.

## Appendix E
## Implementation Details

Our framework of closed-loop constrained RL is illustrated with Fig. 10.

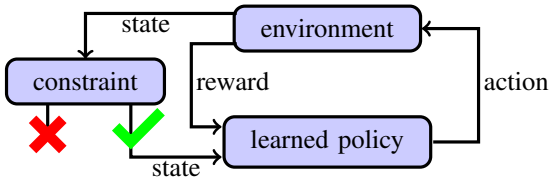



Fig. 10: Illustration of closed-loop constrained RL.

### A. Safe inverted pendulum

In the inverted pendulum experiment, we choose to limit the buffer to $x \in [-0.9, 0.9]$, whereas the cart position $x$ can in fact reach $-1$ and 1. However, when reaching these extremal positions the cart bounces back which provokes a severe discontinuity in the velocities. Our framework can handle discontinuities through the $\varepsilon$ term of (7) Moreover, a safe policy must be able to stabilize the inverted pendulum without pushing the cart to its extremal positions. For all these reasons, we decided to stop our buffer short of $x = \pm 1$.

**Algorithm 2** POLICEd RL training process

**Require:** Environment $E$ (13), constraint (3), reward threshold $r_T$

Let $\mathcal{R}_b$ denote the replay buffer, $\mu_\theta$ our policy, $\mathcal{B}$ our affine buffer, and $\mathcal{U}$ the uniform sampling function.
▷ Main training loop
**while** reward < $r_T$ **and** (15) not satisfied on $\mathcal{B}$ **do**
  **while** reward < $r_T$ **do** ▷ Standard RL training
    Reset $s(0) = 0$ or $s(0) \sim \mathcal{U}(\mathcal{B})$
    **while** not $done$ **do**
      Take action $a(t) = \mu_\theta(s(t))$
      Get reward $r(t) = R(s(t), a(t))$
      Get $(s(t+\delta t), respect, done)$ from $E$
      Add $\big(s(t), a(t), s(t+\delta t), r(t), done\big)$ to $\mathcal{R}_b$
    **end while**
    Update policy $\mu_\theta$ with $\mathcal{R}_b$ ▷ Note: This often reduces the constraint satisfaction of the policy
  **end while**
  ▷ Constraint Training Loop
  **while** (15) not satisfied on $\mathcal{B}$ **do**
    Reset $s(0) \sim \mathcal{U}(\mathcal{V}(\mathcal{B}))$ ▷ Reset the initial state only on the buffer vertices
    Take action $a(0) = \mu_\theta(s(0))$
    Get reward $r(t) = R(s(0), a(0))$
    Get $(s(\delta t), respect, done)$ from $E$
    **if** not $respect$ **then**
      Add $\big(s(0), a(0), s(\delta t), r(0), done\big)$ to $\mathcal{R}_b$
    **end if**
    Update policy $\mu_\theta$ with $\mathcal{R}_b$ ▷ Note: This often reduces the reward earned by the policy
  **end while**
**end while**

**Ensure:** Trajectories $\tau_d^{\mathcal{S}}(s_0, \mu_\theta)$ of (14) starting from safe state $s_0 \in \mathcal{S}_s$ do not leave safe set $\mathcal{S}_s$

### B. Safe robotic arm

At each timestep $k$, the actions $a_k \in \mathbb{R}^7$ outputted by the policy are promoted to only produce small joint changes in each step. For the KUKA Arm, the maximum joint angles are $[\pm 2.97, \pm 2.09, \pm 2.97, \pm 2.09, \pm 2.97, \pm 2.09, \pm 3.05]$, and the agent incurs a penalty of 3 for choosing an action which would go outside of this range.

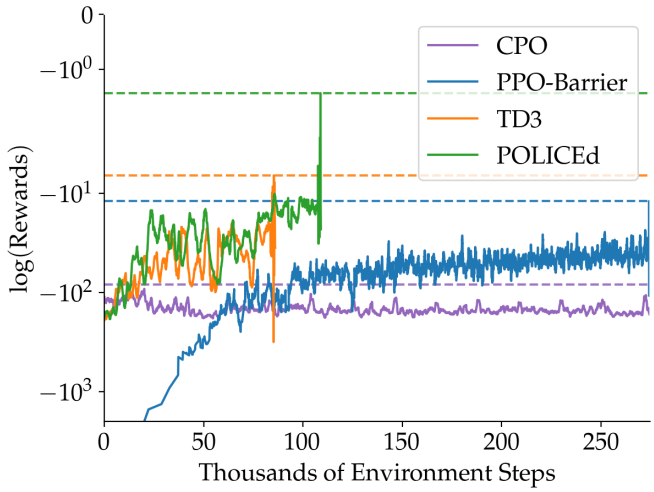
The target is always placed at X-Y-Z coordinates $[0.5, 0.5, 0.5]$, the constraint is a rectangular prism region placed above the target and centered at $[0.5, 0.5, 0.60]$ with side-lengths $[0.16, 0.30, 0.06]$, respectively. We choose this constraint because it prevents the robotic arm from easily reaching the target and from fully minimizing the reward. Indeed, the optimal policy is modified by the existence of the constraint for a large set of initial states $s(0)$. The buffer $\mathcal{B}$ is larger than the constraint and surrounds it on all sides. The buffer is centered at $[0.5, 0.5, 0.62]$ and is also a rectangular prism with sizes $[0.18, 0.32, 0.10]$. These placements are all illustrated in Fig. 9. These choice of side-lengths were

calculated based on the equations presented in Section IV.

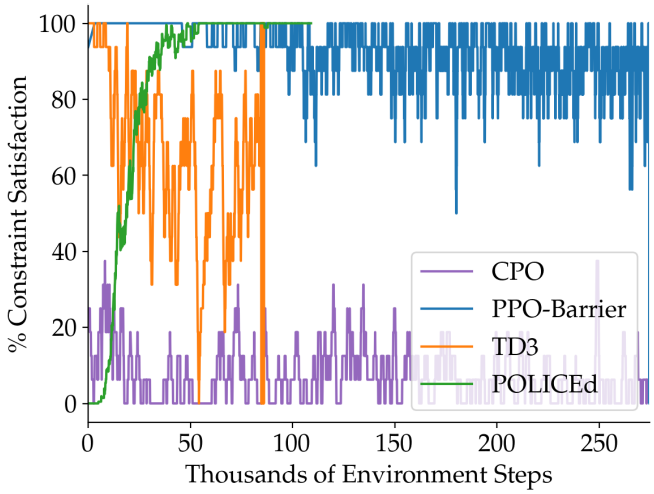
Additionally, during training, we also prevented the agent from taking actions that would violate the constraint during training, which involves both assigning a penalty and leaving the state unchanged. This choice was made to ensure that the episodes that involved violation did not end too quickly, as longer episodes that did not violate the constraint might take more steps and incur more reward penalties. When evaluating the baselines, both training with violations allowed and training without were tested, and the case with better performance was reported.

*1) Comparison with the literature:* While building Table I we noticed several interesting facts about other approaches. We found that the CPO [1] method becomes increasingly incapable of approaching the target. Additionally, given that the state space is very large, it can often experience new initial states during deployment, causing it to still violate the constraint in some cases.

Meanwhile, the PPO-Barrier [64] approach often gathers many thousands of episodes and environment samples in its initial "iterations", allowing it to appear to train quickly (around 250,000 environment steps for convergence). In comparison, our POLICEd approach primarily converges in around 150,000 environment samples. We further showcase these findings in the training curves showcased in Fig. 11.




(a) Averaged log-reward




(b) Averaged percentage of constraint satisfaction

Fig. 11: Reward and Constraint Adherence curves for the safe arm scenario. For the average reward curve, the solid lines correspond to the reward per episode averaged and $log(\cdot)$ over 5 training cycles of each method. The dashed lines are the highest average rewards achieved by each method. Meanwhile for the constraint satisfaction, the constraint respect per episode was averaged over 5 training cycles of each method. Note that the x-axis on both curves is in thousands of environment samples, and each curve ends when it was considered converged. Note that CPO was trained for an average of 4 million environment steps and has been cut off, but the reported maximum reward is over the entire dataset. 95% CI bounds were omitted for clarity, but can be provided on request.