Conservative and Adaptive Penalty for Model-Based Safe Reinforcement Learning

Yecheng Jason Ma[*,1] **Andrew Shen**[*,2]
Osbert Bastani[1] **Dinesh Jayaraman**[1]

[1] University of Pennsylvania [2] University of Melbourne

Abstract

Reinforcement Learning (RL) agents in the real world must satisfy safety constraints in addition to maximizing a reward objective. Model-based RL algorithms hold promise for reducing unsafe real-world actions: they may synthesize policies that obey all constraints using simulated samples from a learned model. However, imperfect models can result in real-world constraint violations even for actions that are predicted to satisfy all constraints. We propose Conservative and Adaptive Penalty (CAP), a model-based safe RL framework that accounts for potential modeling errors by capturing model uncertainty and adaptively exploiting it to balance the reward and the cost objectives. First, CAP inflates predicted costs using an uncertainty-based penalty. Theoretically, we show that policies that satisfy this conservative cost constraint are guaranteed to also be feasible in the true environment. We further show that this guarantees the safety of all intermediate solutions during RL training. Further, CAP adaptively tunes this penalty during training using true cost feedback from the environment. We evaluate this conservative and adaptive penalty-based approach for model-based safe RL extensively on state and image-based environments. Our results demonstrate substantial gains in sample-efficiency while incurring fewer violations than prior safe RL algorithms. Code is available at: https://github.com/Redrew/CAP

1 Introduction

Many applications of reinforcement learning (RL) require the agent to satisfy safety constraints in addition to the standard goal of maximizing the expected reward. For example, in robot locomotion, we may want to impose speed or torque constraints to prevent the robot from damaging itself. Since the set of states that violates the imposed constraints is often a priori unknown, a central goal of *safe* reinforcement learning (Pecka and Svoboda 2014; Garcıa and Fernández 2015) is to learn a reward-maximizing policy that satisfies constraints, while incurring as few constraint violations as possible during the agent's training process.

To reduce the cumulative number of constraint violations during training, a promising approach is to incorporate safety considerations into sample-efficient RL algorithms, such as model-based reinforcement learning (MBRL) (Sutton 1990, 1991). MBRL refers to RL algorithms that use learned transition models to directly synthesize policies using simulated samples, thereby reducing the number of real samples needed to train the policy. Given the true environment transition model, it would be trivial to synthesize safe policies without any violations, since we could simply simulate a sequence of actions to evaluate its safety. However, MBRL agents must learn this transition model from finite experience, which induces approximation errors. In this paper, we ask: *can safety be guaranteed during model-based reinforcement learning, despite these model errors?* We prove that this is indeed possible, and design a practical algorithm that permits model-based safe RL even in high-dimensional problem settings.

Specifically, we propose a model-based safe RL framework involving a **c**onservative and **a**daptive cost **p**enalty (**CAP**). We build on a basic model-based safe RL framework, which simply executes a model-free safe RL algorithm inside a learned transition model. We make two important conceptual contributions to improve this basic approach. First, we derive a conservative upper bound on the error in the policy cost computed according to the learned model. In particular, we show that this error is bounded above by a constant factor of an integral probability metric (IPM) (Müller 1997) computed over the true and learned transition models. Based on this bound, we propose to inflate the cost function with an uncertainty-aware penalty function. We prove that all feasible policies with respect to this conservative cost function, including the *optimal* feasible policy (with highest task reward), are guaranteed to be safe in the true environment. A direct consequence is that we can ensure that all intermediate policies are safe and incur zero safety violations during training.

Second, this penalty function, though theoretically optimal, is often too conservative or cannot be computed for high-dimensional tasks. Therefore, in practice, we propose a heuristic penalty term that includes a scale hyperparameter to modulate the degree of conservativeness: higher scales produce behavior that is more averse to risks arising from modeling errors. Thus, different scales may be appropriate for use with different environments and model fidelities.

We observe that this crucial scale hyperparameter need not be manually set and frozen throughout training. Instead,

[*]These authors contributed equally.

we can exploit the fact that the policy receives feedback on its true cost value from the environment, to formulate the entire inflated cost function as a control plant. In this view, the scale hyparparameter is the control input. Then, we can readily apply existing update rules from the control literature to tune the scale. In particular, we use a proportional-integral (PI) controller (Åström and Hägglund 2006), a simpler variant of a PID controller, to adaptively update the scale using cost feedback from the environment.

Our overall CAP framework incorporates a conservative penalty term into predicted costs in the basic model-based safe RL framework, and adapts its scale to ensure the penalty is neither too aggressive nor too modest. To evaluate CAP, we first illustrate its proposed benefits in simple tabular gridworld environments using a linear programming-based instantiation of CAP; there, we show that CAP indeed achieves zero training violations and exhibits effective adaptive behavior. For state and image-based control environments, we evaluate a second instantiation of CAP, using a cost constraint-aware variant (Wen and Topcu 2020) of cross entropy method (CEM) (De Boer et al. 2005) coupled with state-of-art dynamics models (Chua et al. 2018; Hafner et al. 2019) to optimize action sequences. Through extensive experiments, we show that our practical algorithms substantially reduce the number of real environment samples and unsafe episodes required to learn feasible, high-reward policies compared to model-free baselines as well as ablations of CAP. In summary, our main contributions are:

- an uncertainty-aware cost penalty function that can guarantee the safety of all training policy iterates
- an automatic update rule for dynamically tuning the degree of conservativeness during training.
- a linear program formulation of CAP that achieves near-optimal policies in tabular gridworlds while incurring zero training violation
- and finally, scalable implementations of CAP that learn safe, high-reward actions in continuous control environments with high-dimensional states, including images.

2 Related Work

Safe RL Our work is broadly related to the safe reinforcement learning and control literature; we refer interested readers to (Garcıa and Fernández 2015; Brunke et al. 2021) for surveys on this topic. A popular class of approaches incorporates Lagrangian constraint regularization into the policy updates in policy-gradient algorithms (Achiam et al. 2017; Ray, Achiam, and Amodei 2019; Tessler, Mankowitz, and Mannor 2018; Dalal et al. 2018; Cheng et al. 2019; Zhang, Vuong, and Ross 2020; Chow et al. 2019). These methods build on model-free deep RL algorithms (Schulman et al. 2017b a), which are often sample-inefficient, and do not guarantee that intermediate policies during training are safe. These safe RL algorithms are therefore liable to perform large numbers of unsafe maneuvers during training.

Model-Based Safe RL Model-based safe RL approaches, instead, learn to synthesize a policy through the use of a transition model learned through data. A distinguishing factor among model-based approaches is their assumption on what is known or safe in the environment. Most works assume partially known dynamics (Berkenkamp et al. 2017; Koller et al. 2019) or safe regions (Bastani 2021; Li and Bastani 2020; Bansal et al. 2017; Akametalu et al. 2014), and come with safety guarantees that are tied to these assumptions. In comparison, our work targets the more general setting, obtaining safety guarantees in a data-driven manner. In tabular MDP settings, we prove a high probability guarantee on the safety of any feasible solution under the conservative objective; we subsequently extend this result to ensure the safety of all training episodes. On more complex domains, we provide approximate and practically effective implementations for high-dimensional inputs, such as images, on which previous methods cannot be applied.

Our core idea of using uncertainty estimates as penalty terms to avoid unsafe regions has been explored in several prior works (Kahn et al. 2017; Berkenkamp et al. 2017; Zhang et al. 2020). However, our work provides the first theoretical treatment of the uncertainty-based cost penalty that is independent of the type of the cost (e.g., binary cost) and the parametric choice of the transition model. Our theoretical analysis is similar to that of Yu et al. (2020), though we extend their results, originally in the offline constraint-free setting, to the online constrained MDP setting, and introduce a new result guaranteeing safety for all training episodes. Furthermore, our framework permits the cost penalty weight to automatically adjust to transition model updates, using environment cost feedback during MBRL training.

3 Preliminaries

In safe reinforcement learning, one common problem formulation is to consider an infinite-horizon constrained Markov Decision Process (CMDP) (Altman 1999) $\mathcal{M} = (\mathcal{S}, \mathcal{A}, T, r, c, \gamma, \mu_0)$. Here, $\mathcal{S}, \mathcal{A}$ are the state and action spaces, $T(s' \mid s, a)$ is the transition distribution, $r(s, a)$ is the reward function, $c(s, a)$ is the cost function, $\gamma \in (0, 1)$ is the discount factor, and $s_0 \sim \mu_0(s_0)$ is the initial state distribution; we assume that both $r(s, a)$ and $c(s, a)$ are bounded. A policy $\pi : \mathcal{S} \to \Delta(\mathcal{A})$ is a mapping from state to distribution over actions. Given a fixed policy π, its state-action occupancy distribution is defined to be $\rho_T^\pi(s, a) \coloneqq (1 - \gamma) \sum_{t=0}^{\infty} \gamma^t \Pr^\pi(s_t = s, a_t = a)$, where $\Pr^\pi(s_t = s, a_t = a)$ is the probability of visiting (s, a) at timestep t when executing π in $\mathcal{M}$ starting at $s_0 \sim \mu_0$. The objective in this safe RL formulation is to find the optimal feasible policy π^* that solves the following constrained optimization problem:

$$\begin{aligned} \max_{\pi} \quad & J(\pi) \coloneqq \mathbb{E}\Big[\sum_{t=0} \gamma^t r(s_t, a_t)\Big] \\ \text{s.t.} \quad & J_c(\pi) \coloneqq \mathbb{E}\Big[\sum_{t=0} \gamma^t c(s_t, a_t)\Big] \leq C \end{aligned} \tag{1}$$

where the expectation is over $s_0 \sim \mu_0(\cdot), s_t \sim T(s_t \mid s_{t-1}, a_{t-1}), a_t \sim \pi(\cdot \mid s_t)$, and C is a cumulative constraint threshold that should not be exceeded. We say that a policy π is *feasible* if it does not violate the constraint, and the

optimization problem is feasible if there exists at least one feasible solution (i.e., policy).

Unlike unconstrained MDPs, constrained MDPs cannot be solved by dynamic programming; instead, a common approach is to consider the dual of Eq (1) (Altman 1999):

$$\begin{aligned} \max_{\rho(s,a)\geq 0} \quad & \frac{1}{1-\gamma}\sum_{s,a}\rho(s,a)r(s,a) \\ \text{s.t.} \quad & \frac{1}{1-\gamma}\sum_{s,a}\rho(s,a)c(s,a) \leq C \\ & \sum_{a}\rho(s,a) = (1-\gamma)\mu_0(s) + \gamma\sum_{s',a'}T(s \mid s',a')\rho(s',a'), \forall s \end{aligned} \tag{2}$$

The dual problem Eq (2) is a linear program over occupancy distributions, and can be solved using standard LP algorithms; the second constraint defines the space of valid occupancy distributions by ensuring a "conservation of flow" property among the distributions. Given its solution ρ^*, the optimal policy can be defined as $\pi^*(a \mid s) = \arg\max \rho^*(s,a)$, or equivalently, $\pi^*(a \mid s) = \rho^*(s,a)/\sum_a \rho^*(s,a)$ (if the optimal policy is unique).

Typically, the transition function T is not known to the agent; thus, the optimal policy π^* cannot be directly computed through LP. In model-based reinforcement learning (MBRL), the lack of known T is directly addressed by learning an estimated transition function $\hat{T}$ through data $\mathcal{D} \coloneqq \{(s,a,r,c,s')\}$. Then, we can define a *surrogate* objective to Eq (2) by simply replacing T with $\hat{T}$ and solving Eq (2) as before. Likewise, we can replace $J(\pi)$ with $\hat{J}(\pi)$, and $J_c(\pi)$ with $\hat{J}_c(\pi)$, to obtained model-based objectives in Eq (1). Putting all this together, we may define a basic model-based safe RL framework (Berkenkamp et al. 2017; Brunke et al. 2021) that iterates among three steps: (1) solving for $\hat{\pi}^*$ approximately, (2) collecting data (s,a,r,c,s') from $\hat{\pi}^*$, and (3) updating $\hat{T}$ using all collected data so far. However, at any fixed training iteration, the modeling error may lead to sub-optimal, potentially infeasible $\hat{\pi}^*$. This motivates our approach, described in the following sections.

4 CAP: Conservative and Adaptive Penalty

Next, we introduce **c**onservative and **a**daptive cost-**p**enalty (CAP), our proposed uncertainty and feedback-aware model-based safe RL framework. First, we precisely characterize the downstream effect of the model prediction error on the cost estimate $\hat{J}_c(\pi)$ by providing an upper bound on the true cost $J_c^*(\pi)$, which allows us to derive a penalty function based on the epistemic uncertainty of the model. To this end, we adapt the return simulation lemma results in (Luo et al. 2021; Yu et al. 2020) to the cost setting and derive the following upper bound on the true policy cost $\frac{1}{1-\gamma}\sum_{s,a}\rho_T^\pi(s,a)c(s,a)$ with respect to the estimated policy cost $\frac{1}{1-\gamma}\sum_{s,a}\rho_{\hat{T}}^\pi(s,a)c(s,a)$.

4.1 Cost Penalty

First, given a policy mapping π, we define $V_c^\pi : \mathcal{S} \to \mathbb{R}$ such that $V_c^\pi(s) \coloneqq \mathbb{E}_{\pi,T}[\sum_{t=0}^{\infty}\gamma^t c(s_t,a_t) \mid s_0 = s]$. We make the following assumption on the realizability of V_c^π.

Assumption 4.1. There exists a $\beta > 0$ and a function class $\mathcal{F}$ such that $V_c^\pi \in \beta\mathcal{F}$ for all π.

With this assumption, we show that the difference between the estimated and true costs can be bounded by the integral probability metric (IPM) defined by $\mathcal{F}$ computed between the true and the learned transition models.

Lemma 4.2 (Cost Simulation Lemma and Upper Bound)**.** *Let the $\mathcal{F}$-induced IPM be defined as*

$$\begin{aligned} & d_{\mathcal{F}}(\hat{T}(s,a), T(s,a)) \\ & \coloneqq \sup_{f\in\mathcal{F}} |\mathbb{E}_{s'\sim\hat{T}(s,a)}[f(s')] - \mathbb{E}_{s'\sim T(s,a)}[f(s')]| \end{aligned} \tag{3}$$

Then, the difference between the expected policy cost computed using T and $\hat{T}$ is bounded above:

$$\textstyle\sum_{s,a}(\rho_T^\pi(s,a) - \rho_{\hat{T}}^\pi(s,a))c(s,a) \leq \gamma\beta\sum_{s,a}\rho_{\hat{T}}^\pi(s,a)d_{\mathcal{F}}(\hat{T}(s,a), T(s,a)) \tag{4}$$

We provide a proof in Appendix A. This upper bound illustrates the risk of applying MBRL without modification in safety-critical settings. Attaining $\frac{1}{1-\gamma}\sum_{s,a}\rho_{\hat{T}}^\pi(s,a)c(s,a) \leq C$ does not guarantee that π will be feasible in the real MDP (i.e., $\frac{1}{1-\gamma}\sum_{s,a}\rho_T^\pi(s,a)c(s,a) \leq C$) because the vanilla model-based optimization does not account for the model error's impact on the policy cost estimation, $\beta d_{\mathcal{F}}(\hat{T}(s,a), T(s,a))$.

To enable model-based safe RL that can transfer feasibility from the model to the real world, for a fixed learned transition model $\hat{T}$, we seek a cost penalty function $u_{\hat{T}} : \mathcal{S}\times\mathcal{A} \to \mathbb{R}$ such that $d_{\mathcal{F}}(\hat{T}(s,a), T(s,a)) \leq u_{\hat{T}}(s,a), \forall s,a$. If such a function exists, then we can solve the following LP:

$$\begin{aligned} \max_{\rho(s,a)\geq 0} \quad & \frac{1}{1-\gamma}\sum_{s,a}\rho(s,a)r(s,a) \\ \text{s.t.} \quad & \frac{1}{1-\gamma}\sum_{s,a}\rho(s,a)(c(s,a) + \gamma\beta u_{\hat{T}}(s,a)) \leq C \\ & \sum_{a}\rho(s,a) = (1-\gamma)\mu_0(s) + \gamma\sum_{s',a'}\hat{T}(s \mid s',a')\rho(s',a'), \forall s \end{aligned} \tag{5}$$

We can guarantee that the solution policy π of Eq (5) is feasible for T—in particular, note that

$$\begin{aligned} & \frac{1}{1-\gamma}\sum_{s,a}\rho_T^\pi(s,a)c(s,a) \\ & \leq \frac{1}{1-\gamma}\sum_{s,a}\rho_{\hat{T}}^\pi(s,a)(c(s,a) + \gamma\beta u(s,a)) \leq C. \end{aligned}$$

However, this result is not useful if we cannot compute $d_{\mathcal{F}}(\hat{T}(s,a), T(s,a))$. A suitable function class for analysis is $\mathcal{F} = \{f : \|f\|_\infty \leq 1\}$, which typically can be satisfied with Assumption 4.1 since the per-step cost is bounded. Then, for the tabular-MDP setting (i.e., finite state and action space), we can in fact obtain a strong probabilistic guarantee on feasibility.

Theorem 4.3 (Tabular Case High-Probability Feasibility Guarantee)**.** *Assume $\mathcal{F} = \{f : \|f\|_\infty \leq 1\}$ and that Assumption 4.1 holds. Define $u(s,a) \coloneqq \sqrt{\frac{|\mathcal{S}|}{8n(s,a)}\ln\frac{4|\mathcal{S}||\mathcal{A}|}{\delta}}$,*

Algorithm 1: Safe MBRL with Conservative and Adaptive Penalty (CAP)

1: **Inputs:** Transition model $\hat{T}_\theta$, experience buffer $\mathcal{D}$, cost limit C, initial κ value, κ learning rate α
2: Initialize $\mathcal{D}$ with random policy
3: **for** Episode $= 1, 2, \ldots$ **do**
4: # Conservative penalty
5: Train $\hat{T}_\theta$ using $\mathcal{D}$
6: Optimize π using Eq (5) (LP) or Eq (7) (CCEM)
7: Collect trajectory $\tau := \{(s_t, a_t, r_t, c_t, s_{t+1})\}$ and store to buffer $\mathcal{D} = \mathcal{D} \cup \{\tau\}$
8: # Adaptive penalty
9: Compute $J_c(\pi_t) = \sum_{t=0} \gamma^t c_t$
10: Update $\kappa \leftarrow \kappa + \alpha(J_c(\pi_t) - C)$
11: **end for**

where $n(s,a)$ is the count of (s,a) in $\mathcal{D}$ and $\delta \in (0,1]$. Then, with probability $1-\delta$, a policy that is feasible for Eq (5) is also feasible for Eq (2).

Furthermore, we can extend this result to guarantee that all intermediate solutions during training are safe.

Corollary 4.4 (High-Probability Zero-Training-Violations Guarantee)**.** *Assume the same set of assumptions as Theorem 4.3 and that the training lasts for K episodes. Then, for any $\delta \in (0,1]$, define $u(s,a) := \sqrt{\frac{|\mathcal{S}|}{8n(s,a)} \ln \frac{4K|\mathcal{S}||\mathcal{A}|}{\delta}}$. Then, with probability $1-\delta$, all intermediate solutions to Eq (5) are feasible for Eq (2).*

Proofs are given in Appendix A. At a high level, Theorem 4.3 follows from observing that $d_{\mathcal{F}}$ is the total variation distance for the chosen $\mathcal{F}$ and applying concentration bound on the estimation error of $\hat{T}$. Then, Corollary 4.4 can be shown by a union-bound argument.

Together, these results suggest that a principled way of incorporating a conservative penalty function into the 3-step basic model-based safe RL framework described at the end of Sec. 3 is to replace the original constrained MDP objective (i.e., Eq (2)) with its conservative variant (i.e., Eq (5)).

4.2 Adaptive Cost Penalty

The upper bound derived in the previous section can be overly conservative in practice. Thus, we derive an adaptive penalty function based on environment feedback to make it more practical. First, we observe that the conservative penalty modification described above is not yet enough for a practical algorithm, because the proposed penalty function as in the theorem or the corollary is too conservative, to the extent that Eq (5) might admit no solutions. In practice, it is often estimated as $u(s,a) := \kappa/\sqrt{n(s,a)}$, where $\kappa \in \mathbb{R}$ is some scaling parameter.

We observe that setting κ to a fixed value throughout training can lead to poor performance. Different scales may be appropriate for use with different environments, tasks, and stages of training. If it is set too low, then the cost penalty may not be large enough to ensure safety. On the other hand, if it is set too large, then the model may be overly conservative, discouraging exploration and leading to training instability.

To avoid these issues, we propose to adaptively update κ during training. Observe that the *effect* of a particular κ value on a policy's true cost in the environment can be measured from executing this policy in the real environment. Thus, we can in fact view the co-evolution of the policy and the learned transition model as a control plant, for which the policy cost is the control output; then, κ can be viewed as its control input. Now, to set κ, we employ a PI controller, a simple variant of the widely used PID controller (Åström and Hägglund 2006) from classical control, to incrementally update κ based on the current gap between the policy's true cost and the cost threshold. More precisely, we propose the following PI control update rule:

$$\kappa_{t+1} = \kappa_t + \alpha(J_c(\pi_t) - C) \tag{6}$$

where α is the learning rate.

This update rule is intuitive. Consider the direction of the κ update when $J_c(\pi_t) < C$. In this case, the update will be negative, which matches our intuition that the cost penalty can be applied less conservatively due to the positive margin to the cost limit C. The argument for the case $J_c(\pi_t) > C$ is analogous. In high-dimensional environments, as the full expected cost cannot be computed exactly, and we instead approximate it using a single episode (i.e., the current policy π_t rollout in the environment). To ensure κ is non-negative, we additionally perform a $\max(0, \cdot)$ operation after each PI update.

Now, the full CAP approach is described in Algorithm 1. At a high level, CAP extends upon the basic model-based safe RL framework by (1) solving the conservative LP (Line 7, Eq (5)), and (2) adapting κ using PI control (Lines 10 & 11). We set the initial value for κ using an exponential search mechanism, which we describe in the Appendix. We validate this LP formulation of CAP using a gridworld environment in our experiments.

4.3 CAP for High-Dimensional States

Note that this tabular LP variant of CAP cannot extend to environments with continuous state and action spaces, representative of many high-dimensional RL problems of interest (e.g., robotics); their continuous nature precludes enumerating all state-action pairs, which is needed to express the linear program. Therefore, we propose a scalable implementation of CAP amenable to continuous control problems. First, we revert back to the policy-based formulation in Eq (1), and define the following equivalent objective:

$$\begin{aligned} \max_{\pi} \quad & \mathbb{E}\Big[\sum_{t=0} \gamma^t r(s_t, a_t)\Big] \\ \text{s.t.} \quad & \mathbb{E}\Big[\sum_{t=0} \gamma^t \cdot (c(s_t, a_t) + \kappa u_{\hat{T}}(s_t, a_t))\Big] \leq C \end{aligned} \tag{7}$$

where $u(s_t, a_t)$ is a heuristic penalty function based on statistics of the learned transition model.

To optimize Eq (7), we employ the constrained cross entropy method (CCEM) (Wen and Topcu 2020; Liu et al.

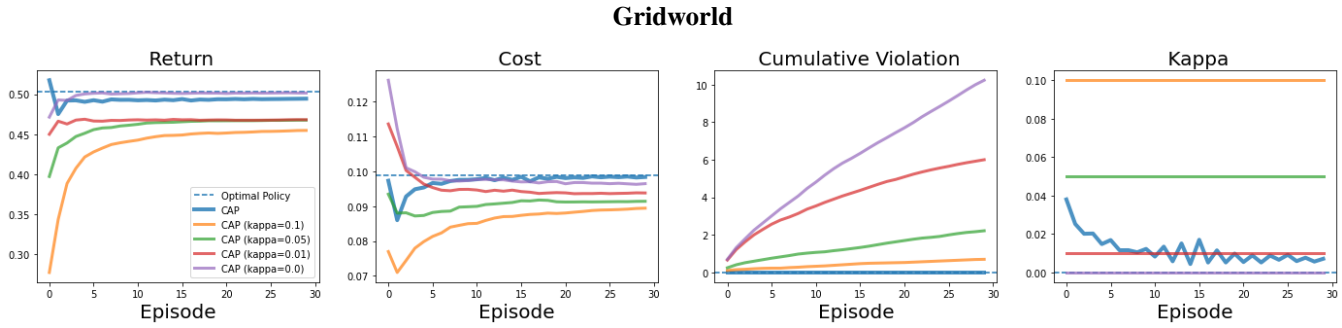


Figure 1: **Tabular gridworld results.** CAP achieves near-optimal policy with zero constraint violations during training, while all ablations either converge to sub-optimal solutions or incur a high number of training violations.

2021) as our trajectory optimizer; the procedure is summarized in Algorithm 2 in the Appendix. At a high level, CCEM first samples N action sequences (Line 4) and computes their values and costs (Line 5). Then, if there were more than E samples that satisfy the constraint, then the E samples with highest rewards are selected (Line 10); otherwise, the E samples with lowest costs are selected (Line 8). These selected *elite* samples are used to update the sampling distribution (Line 12). This process continues for I iterations, and the eventual distribution mean is selected as the optimal action sequence (Line 14).

Next, we specify the choice of transition model and penalty function $u(s, a)$ for state-based and visual observation-based implementations, respectively. For the former, we model the environment transition function using an ensemble of size N, $\{\hat{T}_\theta^i = \mathcal{N}(\mu_\theta^i, \Sigma_\theta^i)\}_{i=1}^N$ (Chua et al. 2018) and set $u(s, a) = \max_{i=1}^N \left\|\Sigma_\theta^i(s, a)\right\|_{\mathrm{F}}$ to be the maximum Frobenius norm of the ensemble standard deviation outputs, as done for offline RL in Yu et al. (2020). Our visual-based implementation builds on top of PlaNet (Hafner et al. 2019), a state-of-art visual model-based transition model; here, we set $u(s, a)$ to be the ensemble disagreement of one-step hidden state prediction models (Sekar et al. 2020). See the Appendix for details. In both tabular and deep RL settings, CAP adds negligible computational overhead, making it a practical safe RL algorithm.

5 Experiments

CAP provides a general, principled, and practical framework for applying MBRL to safe RL. To support this claim, we comprehensively evaluate CAP against its ablations as well as model-free baselines in various environments. More specifically, we investigate the following questions:

(Q1) Does CAP's theoretical guarantees approximately hold in tabular environments?

(Q2) Does CAP improve reward and safety upon its ablations (i.e., fixed κ values)?

(Q3) Is CAP more sample and *cost* efficient than state-of-art model-free baselines?

(Q4) Can CAP learn safe policies even with high-dimensional inputs, such as images?

We investigate Q1-2 using a gridworld environment, and Q2-4 on two high-dimensional deep RL benchmarks. Our code is included in the supplementary materials.

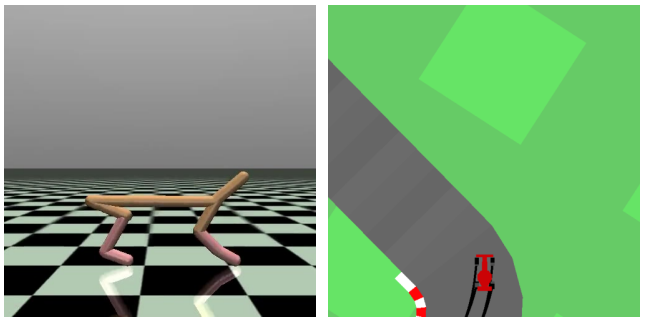
5.1 Gridworld

We begin by validating our theoretical findings in tabular gridworld, where we can solve the constrained optimization problem (Eq (5)) exactly using standard LP algorithms.

Environment, Methods, Training Details We consider an 8×8 gridworld with stochastic transitions; the reward and the cost functions are randomly generated Bernoulli distributions drawn according to a Beta prior. In addition to CAP, we compare against CAP ablations with fixed κ values of $0, 0.01, 0.05$, and 0.1; $\kappa = 0$ corresponds to the basic MBRL approach without penalty. We also include the oracle LP solution computed using the true environment dynamics. For each method (except the oracle), the training procedure lasts 30 iterations, in which each iterate includes (1) collecting 500 samples using the current LP solution, (2) updating $\hat{T}$, and (3) solving the new conservative LP objective. See Appendix for more environment and training details.

Metrics & Results In Figure 1, we illustrate the training curves for the return, cost, the cumulative number of intermediate policy solutions that violate the cost threshold. The first two metrics are standard, and the violation metric measures how safely a method explores. We additionally illustrate the training evolution of κ. These curves are the averages taken over the 100 gridworld simulations; we defer standard deviation error bar to the Appendix for better visualizations except for the kappa curve.

As expected, CAP ($\kappa = 0$), due to its asymptotically consistent nature, converges to the oracle as training continues; however, this comes at the cost of the highest number of training violations, precisely due to the lack of an uncertainty-aware penalty function. In sharp contrast, CAP is very close to the oracle in both reward and cost, and does so without incurring a single violation in all 100 trials, as indicated by its flat horizontal line at 0 in the violation plot. These results validate our key theoretical claims that when the cost penalty is applied properly, the resulting policy is guaranteed to be safe (Theorem 4.3); furthermore, it applies



HalfCheetah **Car-Racing**

Figure 2: **High Dimensional Environments.**

to all intermediate policies during training (Corollary 4.4), answering Q1 above.

On the other hand, CAP ablations with fixed κ values, though constraint-satisfying at the end, incur higher number of violations and achieve sub-optimal solutions, validated by their lower returns and conservative costs. Interestingly, while all these variants on *average* satisfy the constraint from Episode 2 and on (i.e., their average costs are below the threshold of 0.1 in the cost plot), their average numbers of violations uniformly increase throughout training. This suggests that fixed κ values are not *robust* to random gridworld simulations, as the same value may be too modest for some random draws and hence incur violations, and too aggressive for some other draws and lead to suboptimal solutions. Indeed, we observe greater variance in the performance of fixed κ ablations than CAP (see Appendix).

In contrast, CAP automatically finds suitable sequences of κ for each simulation, evidenced by the large variance the κ sequences exhibit over the simulations. Its zero-violation and lower variance in all metrics suggest that the adaptive penalty mechanism has the additional benefit of being *distributionally robust* to the randomness in the environment distribution. Finally, the overall downward oscillating trend indeed reflects CAP's effectiveness at using feedback to optimize reward and cost simultaneously. Together, these ablations answer Q2 affirmatively. In the Appendix, we provide additional result analysis for this tabular experiment.

5.2 High-Dimensional Environments

Next, we evaluate CAP's generality and effectiveness in high-dimensional environments. We begin by summarizing our experimental setup; details are in the Appendix.

Environments We consider two deep RL environments, spanning different input modalities, constraint types, and associated cost types. We describe these environments here; see Figure 2 for illustrations:

- A velocity-constrained version of Mujoco HalfCheetah (Todorov, Erez, and Tassa 2012), representative of robot tasks in which we want to avoid robots damaging themselves from over-exertion. The state space is 17-dimensional and the action space is 6-dimensional (controlling the robot joints). To ensure a meaningful cost constraint, we constrain the average velocity to be below 50% of the average velocity of an unconstrained expert PPO agent (152) (Zhang, Vuong, and Ross 2020).
- A 2D image-based racing task Car-Racing (Brockman et al. 2016), with randomized tracks in every episode. The state space is a $64 \times 64 \times 3$ top-down view of the car; the action space is continuous and 3-dimensional (steering, acceleration, and braking). A per-step cost of 1 is incurred if any wheels skid from excessive force; the cost limit is 0, indicating that the car should never skid. This task is representative of visual environments with complex dynamics.

Method	HalfCheetah			
	Steps	Return (↑)	Cost (Limit 152) (↓)	Violation (↓)
Random	NA	-29.3	52.7	NA
PPO	1M	2791.3	296.9	378.0
	100K	670.2	97.6	0
PPO-Lag	1M	1436.8	150.7	108.0
	100K	670.2	97.6	0
FOCOPS	1M	1591.4	160.2	202.8
	100K	456.0	84.6	0
CAP (Ours)	100K	1456.3	144.3	1.7
Method	**Car-Racing**			
	Steps	Return (↑)	Cost (Limit 0) (↓)	Violation (↓)
Random	NA	3.9	159.3	NA
PPO	1M	32.7	52.0	975.0
	200K	48.8	224.8	196.0
PPO-Lag	1M	-3.2	0.0	101.3
	200K	-3.2	0.3	101.3
FOCOPS	1M	23.4	0.8	581.0
	200K	16.2	3.9	172.0
CAP	200K	21.7	0.4	93.3

Table 1: **Baseline comparison results.** CAP is substantially more sample-efficient with respect to both return and cost. In addition, it is much safer during training, as demonstrated by the significantly fewer violations.

Baselines In these high-dimensional settings, we compare against both deep model-free safe RL baselines as well as CAP ablations. To this end, we include PPO-Lagrangian (**PPO-LAG**), which iterates between PPO policy update and cost lagrangian parameter update to simultaneously optimize return and constraint satisfaction; despite its simplicity, PPO-LAG has been shown to be a strong safe RL baseline (Ray, Achiam, and Amodei 2019). Additionally, we include **FOCOPS** (Zhang et al. 2020), a state-of-art model-free algorithm which uses first-order projection methods to ensure that constraint satisfaction minimally deteriorates policy return. Finally, we include PPO (Schulman et al. 2017b) in order to provide comparison to an unconstrained method. Finally, as in the gridworld experiment, we consider CAP ablations with fixed κ values and separately visualize the training curves. We use $\kappa = 0, 0.1, 1, 10$ for both HalfCheetah and Car-Racing to include a wide range of magnitudes; in particular, $\kappa = 0$ corresponds to the basic model-based safe RL approach without the conservative penalty; this is the constrained CEM method introduced by (Liu et al. 2021). In Appendix E.5, we additionally compare to PETS (Chua et al. 2018), a widely used unconstrained model-based planning method.

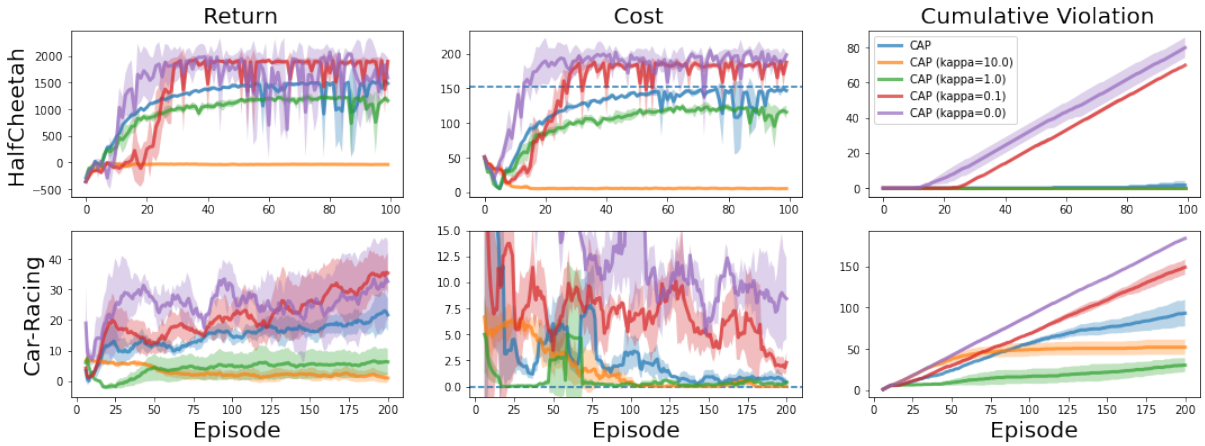


Figure 3: **CAP Ablations on HalfCheetah (top) and Car-Racing (bottom).** The adaptive κ achieves better balance than all fixed κ values and incurs much fewer violations during training.

Training Details & Evaluation Metrics For model-free algorithms, we train using 1M environment steps, and for our model-based algorithms, we train using 100K steps for HalfCheetah and 200K steps for Car-Racing. An episode in both environments is 1000 steps. We report results on Car-Racing at 200k since it is more challenging to learn the dynamics of a visual environment; both model-free and model-based methods take more steps to converge in Car-Racing. As in gridworld, we report the training curves of the return, cost, and cumulative episodic violations; they are included in the Appendix. In the main text, we report a numerical "snapshot" version of these curves at the end of training (average over last 10 episodes); for model-free baselines, we also report these metrics after 100K/200K steps to have a head-to-head comparison against CAP. We include all hyperparameters and implementation details in the Appendix.

Baseline Comparisons Results The results are shown in Table 1. While the most competitive algorithm FOCOPS matches CAP's return and cost with 1 million environmental steps in both environments, CAP requires about 5-10$\times$ fewer steps, demonstrating its sample efficiency. Furthermore, the relative performance of CAP at 100K/200K steps is significantly better than all model-free algorithms, which have not learned a good policy by that point. This has direct implication for safety. On the Car-Racing environment, because model-free methods learn much slower, they also spend more training episodes violating the constraint. On HalfCheetah, all methods achieve 0 cumulative episodic violations with 100K steps, but this is because in HalfCheetah the algorithm will not violate the speed constraint initially because it has not learned the running behavior yet.

It is particularly illuminating to observe the cumulative episodic violations at the end of each method's training: we see that CAP violates the speed constraint in HalfCheetah for fewer than 2 episodes out of its 100 training episodes, while all baselines violate this constraint at much higher rates and volumes. This confirms that these model-free methods struggle to ensure safety during training regardless of the safety of their final policy, while CAP is able to minimize violations throughout training. On the more challenging image-based Car-Racing environment, CAP cannot avoid training violations entirely, but manages to significantly reduce them compared to the baselines. These comparisons provide strong evidence for Q3 and Q4.

CAP Ablations Results The training curves of CAP as well as its ablations are illustrated in Figure 3. Consistent with our findings in gridworld, setting κ to a fixed value is rarely desirable. Setting it too low often leads to solutions that fail to satisfy constraint, suggested by the high training cost and violations of CAP ($\kappa = 0.0, 0.1$) in both environments; setting it too high often precludes reward learning in the first place, evidenced by the training curves of CAP ($\kappa = 10.0$) in both environments. Furthermore, since the cost limit is 0 on Car-Racing, exploration will always violate the constraint initially. Hence, we can additionally measure the safe exploration of a method by its slope on the violation curve: the lower the slope, the fewer violations a method incurs as training goes on. There, we see that CAP has the flattest violation slope out of all variants that learn policies with non-trivial driving behavior, answering Q2 affirmatively.

6 Conclusion

We have presented CAP, a general model-based safe reinforcement learning framework. We have derived a linear programming formulation and proven that we can guarantee safety by using a conservative penalty; this penalty is then made adaptive based on environmental feedback to make it practically useful. We have validated our theoretical results in a tabular gridworld environment and demonstrated that CAP can be easily extended to high-dimensional visual environments through appropriate choices of optimizer and transition models. In future work, we aim to extend CAP to the offline and risk-sensitive settings (Yu et al. 2020; Ma, Jayaraman, and Bastani 2021). Overall, we believe that CAP opens many future directions in making MBRL practically useful for safe RL.

Acknowledgement

This work is funded in part by an Amazon Research Award, gift funding from NEC Laboratories America, NSF Award CCF-1910769, NSF Award CCF-1917852 and ARO Award W911NF-20-1-0080. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein

References

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

Akametalu, A. K.; Fisac, J. F.; Gillula, J. H.; Kaynama, S.; Zeilinger, M. N.; and Tomlin, C. J. 2014. Reachability-based safe learning with Gaussian processes. In 53rd IEEE Conference on Decision and Control, 1424–1431. IEEE.

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

Åström, K. J.; and Hägglund, T. 2006. PID control. IEEE Control Systems Magazine, 1066.

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

Bastani, O. 2021. Safe Reinforcement Learning with Nonlinear Dynamics via Model Predictive Shielding. In 2021 American Control Conference (ACC), 3488–3494. IEEE.

Berkenkamp, F.; Turchetta, M.; Schoellig, A. P.; and Krause, A. 2017. Safe Model-based Reinforcement Learning with Stability Guarantees. arXiv:1705.08551.

Brockman, G.; Cheung, V.; Pettersson, L.; Schneider, J.; Schulman, J.; Tang, J.; and Zaremba, W. 2016. OpenAI Gym. arXiv:1606.01540.

Brunke, L.; Greeff, M.; Hall, A. W.; Yuan, Z.; Zhou, S.; Panerati, J.; and Schoellig, A. P. 2021. Safe Learning in Robotics: From Learning-Based Control to Safe Reinforcement Learning. arXiv:2108.06266.

Cheng, R.; Orosz, G.; Murray, R. M.; and Burdick, J. W. 2019. 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, 3387–3395.

Chow, Y.; Nachum, O.; Faust, A.; Duenez-Guzman, E.; and Ghavamzadeh, M. 2019. Lyapunov-based safe policy optimization for continuous control. arXiv preprint arXiv:1901.10031.

Chua, K.; Calandra, R.; McAllister, R.; and Levine, S. 2018. Deep Reinforcement Learning in a Handful of Trials using Probabilistic Dynamics Models. arXiv:1805.12114.

Dalal, G.; Dvijotham, K.; Vecerik, M.; Hester, T.; Paduraru, C.; and Tassa, Y. 2018. Safe exploration in continuous action spaces. arXiv preprint arXiv:1801.08757.

De Boer, P.-T.; Kroese, D. P.; Mannor, S.; and Rubinstein, R. Y. 2005. A tutorial on the cross-entropy method. Annals of operations research, 134(1): 19–67.

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

Gurobi Optimization, LLC. 2021. Gurobi Optimizer Reference Manual.

Hafner, D.; Lillicrap, T.; Fischer, I.; Villegas, R.; Ha, D.; Lee, H.; and Davidson, J. 2019. Learning Latent Dynamics for Planning from Pixels. arXiv:1811.04551.

Kahn, G.; Villaflor, A.; Pong, V.; Abbeel, P.; and Levine, S. 2017. Uncertainty-Aware Reinforcement Learning for Collision Avoidance. arXiv:1702.01182.

Koller, T.; Berkenkamp, F.; Turchetta, M.; Boedecker, J.; and Krause, A. 2019. Learning-based Model Predictive Control for Safe Exploration and Reinforcement Learning. arXiv:1906.12189.

Li, S.; and Bastani, O. 2020. Robust model predictive shielding for safe reinforcement learning with stochastic dynamics. In 2020 IEEE International Conference on Robotics and Automation (ICRA), 7166–7172. IEEE.

Liu, Z.; Zhou, H.; Chen, B.; Zhong, S.; Hebert, M.; and Zhao, D. 2021. Constrained Model-based Reinforcement Learning with Robust Cross-Entropy Method. arXiv:2010.07968.

Luo, Y.; Xu, H.; Li, Y.; Tian, Y.; Darrell, T.; and Ma, T. 2021. Algorithmic Framework for Model-based Deep Reinforcement Learning with Theoretical Guarantees. arXiv:1807.03858.

Ma, Y. J.; Jayaraman, D.; and Bastani, O. 2021. Conservative Offline Distributional Reinforcement Learning. arXiv:2107.06106.

Müller, A. 1997. Integral probability metrics and their generating classes of functions. Advances in Applied Probability, 29(2): 429–443.

Pecka, M.; and Svoboda, T. 2014. Safe exploration techniques for reinforcement learning–an overview. In International Workshop on Modelling and Simulation for Autonomous Systems, 357–375. Springer.

Ray, A.; Achiam, J.; and Amodei, D. 2019. Benchmarking safe exploration in deep reinforcement learning.

Schulman, J.; Levine, S.; Moritz, P.; Jordan, M. I.; and Abbeel, P. 2017a. Trust Region Policy Optimization. arXiv:1502.05477.

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017b. Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347.

Sekar, R.; Rybkin, O.; Daniilidis, K.; Abbeel, P.; Hafner, D.; and Pathak, D. 2020. Planning to Explore via Self-Supervised World Models. arXiv:2005.05960.

Sutton, R. S. 1990. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In Machine learning proceedings 1990, 216–224. Elsevier.

Sutton, R. S. 1991. Planning by incremental dynamic programming. In Machine Learning Proceedings 1991, 353–357. Elsevier.

Tessler, C.; Mankowitz, D. J.; and Mannor, S. 2018. Reward constrained policy optimization. arXiv preprint arXiv:1805.11074.

Todorov, E.; Erez, T.; and Tassa, Y. 2012. MuJoCo: A physics engine for model-based control. In 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, 5026–5033.

Wen, M.; and Topcu, U. 2020. Constrained cross-entropy method for safe reinforcement learning. IEEE Transactions on Automatic Control.

Yu, T.; Thomas, G.; Yu, L.; Ermon, S.; Zou, J.; Levine, S.; Finn, C.; and Ma, T. 2020. Mopo: Model-based offline policy optimization. arXiv preprint arXiv:2005.13239.

Zhang, J.; Cheung, B.; Finn, C.; Levine, S.; and Jayaraman, D. 2020. Cautious Adaptation For Reinforcement Learning in Safety-Critical Settings. arXiv:2008.06622.

Zhang, Y.; Vuong, Q.; and Ross, K. W. 2020. First order constrained optimization in policy space. arXiv preprint arXiv:2002.06506.