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 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.

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 [1, 2] 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) [3, 4]. 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

Deep RL Workshop, NeurIPS 2021.

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) [5] 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, tasks, and model fidelities. We observe that this crucial scale hyperparameter need not be manually set and frozen throughout training. Instead, 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 the proportional-integral (PI) controller [6], a simple variant of the 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 [7] of cross entropy method (CEM) [8] coupled with state-of-art dynamics models [9, 10] 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 [2, 11] for surveys on this topic. A popular class of approaches incorporates Lagrangian constraint regularization into the policy updates in policy-gradient algorithms [12, 13, 14, 15, 16, 17, 18]. These methods build on model-free deep RL algorithms [19, 20], 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 [21, 22] or safe regions [23, 24, 25, 26], and come with safety guarantees that are tied to these assumptions. In comparison, our work targets the more general setting, involving no such prior knowledge. 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.

Our core idea of using uncertainty estimates as penalty terms to avoid unsafe regions has been explored in several prior works [27, 21, 28]. 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 [29], 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) [30] $\mathcal{M} = (\mathcal{S}, \mathcal{A}, T, r, c, \gamma, \mu_0)$. 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 $\rho^{\pi}_{T}(s, a) := (1 - \gamma) \sum_{t=0}^{\infty} \gamma^t \mathrm{Pr}^{\pi}(s_t = s, a_t = a)$, where $\mathrm{Pr}^{\pi}(s_t = s, a_t = a)$ 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:

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

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 [31]. Instead, a common approach is to consider the dual of Eq (1) [30]:

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

where $\rho(s) = \sum_a \rho(s, a)$. 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 $\mathcal{D} := \{(s, a, r, c, s')\}$. 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 [21, 11] 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 [32, 29] 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 $V_c^\pi(s) := \mathbb{E}_{\pi,T}[\sum_{t=0}^\infty \gamma^t c(s_t, a_t) \mid s_0 = s]$. 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*

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

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

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

We provide a proof in Appendix A. This upper bound illustrates the risk of applying MBRL without modification in safety-critical settings. Attaining $\frac{1}{1-\gamma}\sum_{s,a}\rho_{\hat{T}}^\pi(s,a)c(s,a) \le C$ does not guarantee that π will be feasible in the real MDP (i.e., $\frac{1}{1-\gamma}\sum_{s,a}\rho_T^\pi(s,a)c(s,a) \le C$) 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 $d_{\mathcal{F}}(\hat{T}(s,a), T(s,a)) \le u_{\hat{T}}(s,a), \forall s, a$. If such a function exists, then we can solve the following LP:

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

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

$$\frac{1}{1-\gamma}\sum_{s,a}\rho_T^\pi(s,a)c(s,a) \le \frac{1}{1-\gamma}\sum_{s,a}\rho_{\hat{T}}^\pi(s,a)(c(s,a) + \gamma\beta u(s,a)) \le C$$

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 $\mathcal{F} = \{f : \|f\|_\infty \le 1\}$, 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 $\mathcal{F} = \{f : \|f\|_\infty \le 1\}$ and that Assumption 4.1 holds. Define $u(s,a) := \sqrt{\frac{|\mathcal{S}|}{8n(s,a)}\ln\frac{4|\mathcal{S}||\mathcal{A}|}{\delta}}$, 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.

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 $\kappa$ value, $\kappa$ learning rate $\alpha$
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 $\pi$ 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
```

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. A key observation we make is that the *effect* of a particular κ value has on a policy's true cost in the environment can be measured from executing this policy in the real environment. Leveraging this insight, 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 [6] from classic 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).

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:

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

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) [7, 33] 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, $\{\hat{T}_\theta^i = \mathcal{N}(\mu_\theta^i, \Sigma_\theta^i)\}_{i=1}^N$ [9] and set $u(s, a) = \max_{i=1}^N \left\|\Sigma_\theta^i(s, a)\right\|_{\text{F}}$ to be the maximum Frobenius norm of the ensemble standard deviation outputs, as done for offline RL in [29]. Our visual-based implementation builds on top of PlaNet [10], 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 [34]. See the Appendix for details.

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.

5.1.1 Environment, Methods, Training Details

We consider a 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

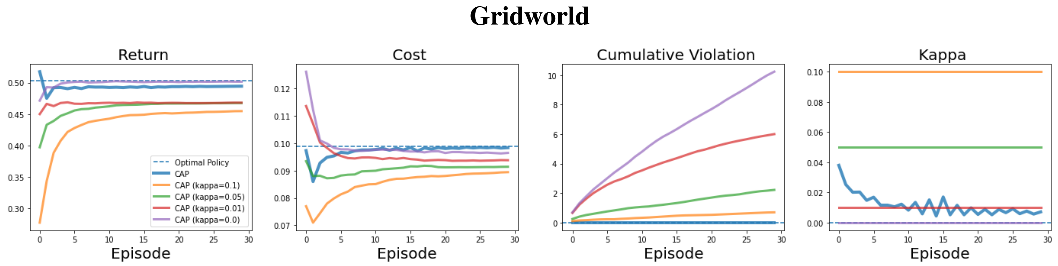

Figure 1: **Tabular gridworld results.** CAP achieves near-optimal policy with zero constraint violations during training, while all ablations either converge to sub-optimal solutions or incur a high number of training violations.

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.

5.1.2 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 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 κs 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.

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.

5.2.1 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. We use **HalfCheetah** [35] as our state-based environment; here, the cost function is the agent's horizontal velocity, and we set the cost constraint to be the 50% of the average velocity of an unconstrained expert PPO agent, 152. For the image-based environment, we use **Car-Racing** [36]. The tracks are randomized 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.

5.2.2 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 [13]. Additionally, we include **FOCOPS** [28], 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 [19] 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.

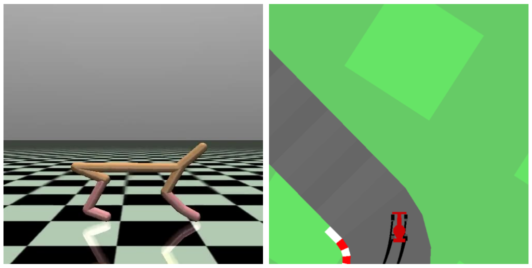

Figure 2: **High Dimensional Environments.**

5.2.3 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. For all methods, we report their hyperparameters as well as implementation details in the Appendix. In Appendix, we also supply videos of trained CAP agents in Car-Racing.

Method		**HalfCheetah**		
	Steps	Return (↑)	Cost (Limit 152) (↓)	Violation (↓)
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
Random	NA	-29.3	52.7	NA
CAP (Ours)	100K	1456.3	144.3	1.7
Method		**Car-Racing**		
	Steps	Return (↑)	Cost (Limit 0) (↓)	Violation (↓)
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
Random	NA	3.9	159.3	NA
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.

5.2.4 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×

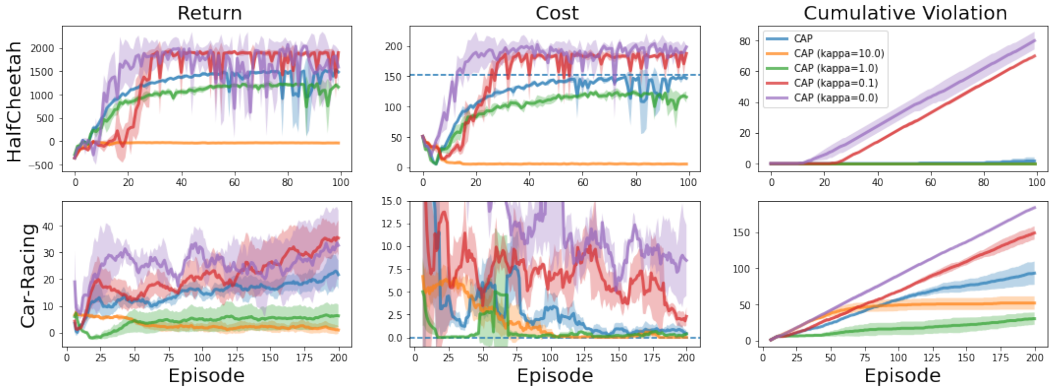

Figure 3: **CAP Ablations on HalfCheetah (top) and Car-Racing (bottom).** The adaptive κ achieves better balance than all fixed κ values and incurs much fewer violations during training.

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 (i.e., 100 episodes), 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.

5.2.5 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. These results once again show that CAP is preferrable to its fixed-κ variants, providing a positive answer to Q2.

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 environments through appropriate choices of optimizer and transition models. In future work, we aim to implement CAP using model-based policy optimization [37, 38] methods, which have shown to attain better performance in practice. Additionally, we believe that using a full PID controller will further improve safety of CAP, especially on problems where incremental update is not aggressive enough for safety. Overall, we believe that CAP opens many future directions in making MBRL practically useful for safe RL.

References

[1] Martin Pecka and Tomas Svoboda. Safe exploration techniques for reinforcement learning–an overview. In *International Workshop on Modelling and Simulation for Autonomous Systems*, pages 357–375. Springer, 2014.

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

[3] Richard S Sutton. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Machine learning proceedings 1990*, pages 216–224. Elsevier, 1990.

[4] Richard S Sutton. Planning by incremental dynamic programming. In *Machine Learning Proceedings 1991*, pages 353–357. Elsevier, 1991.

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

[6] Karl J Åström and Tore Hägglund. Pid control. *IEEE Control Systems Magazine*, 1066, 2006.

[7] Min Wen and Ufuk Topcu. Constrained cross-entropy method for safe reinforcement learning. *IEEE Transactions on Automatic Control*, 2020.

[8] Pieter-Tjerk De Boer, Dirk P Kroese, Shie Mannor, and Reuven Y Rubinstein. A tutorial on the cross-entropy method. *Annals of operations research*, 134(1):19–67, 2005.

[9] Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models, 2018.

[10] Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels, 2019.

[11] 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, 2021.

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

[13] Alex Ray, Joshua Achiam, and Dario Amodei. Benchmarking safe exploration in deep reinforcement learning. 2019.

[14] Chen Tessler, Daniel J Mankowitz, and Shie Mannor. Reward constrained policy optimization. *arXiv preprint arXiv:1805.11074*, 2018.

[15] 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.

[16] 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.

[17] Yiming Zhang, Quan Vuong, and Keith W Ross. First order constrained optimization in policy space. *arXiv preprint arXiv:2002.06506*, 2020.

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

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

[20] John Schulman, Sergey Levine, Philipp Moritz, Michael I. Jordan, and Pieter Abbeel. Trust region policy optimization, 2017.

[21] Felix Berkenkamp, Matteo Turchetta, Angela P. Schoellig, and Andreas Krause. Safe model-based reinforcement learning with stability guarantees, 2017.

[22] Torsten Koller, Felix Berkenkamp, Matteo Turchetta, Joschka Boedecker, and Andreas Krause. Learning-based model predictive control for safe exploration and reinforcement learning, 2019.

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

[24] Shuo Li and Osbert Bastani. Robust model predictive shielding for safe reinforcement learning with stochastic dynamics. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7166–7172. IEEE, 2020.

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

[26] Anayo K Akametalu, Jaime F Fisac, Jeremy H Gillula, Shahab Kaynama, Melanie N Zeilinger, and Claire J Tomlin. Reachability-based safe learning with gaussian processes. In *53rd IEEE Conference on Decision and Control*, pages 1424–1431. IEEE, 2014.

[27] Gregory Kahn, Adam Villaflor, Vitchyr Pong, Pieter Abbeel, and Sergey Levine. Uncertainty-aware reinforcement learning for collision avoidance, 2017.

[28] Jesse Zhang, Brian Cheung, Chelsea Finn, Sergey Levine, and Dinesh Jayaraman. Cautious adaptation for reinforcement learning in safety-critical settings, 2020.

[29] Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. Mopo: Model-based offline policy optimization. *arXiv preprint arXiv:2005.13239*, 2020.

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

[31] Dimitri P Bertsekas et al. *Dynamic programming and optimal control: Vol. 1*.

[32] Yuping Luo, Huazhe Xu, Yuanzhi Li, Yuandong Tian, Trevor Darrell, and Tengyu Ma. Algorithmic framework for model-based deep reinforcement learning with theoretical guarantees, 2021.

[33] Zuxin Liu, Hongyi Zhou, Baiming Chen, Sicheng Zhong, Martial Hebert, and Ding Zhao. Constrained model-based reinforcement learning with robust cross-entropy method, 2021.

[34] Ramanan Sekar, Oleh Rybkin, Kostas Daniilidis, Pieter Abbeel, Danijar Hafner, and Deepak Pathak. Planning to explore via self-supervised world models, 2020.

[35] 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, 2012.

[36] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.

[37] Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization. *arXiv preprint arXiv:1906.08253*, 2019.

[38] Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination, 2020.

[39] Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2021.

A Proofs

In this section, we provide proofs for the theoretical results appeared in Section 4. We will restate each of the results and then append their corresponding proof.

Lemma A.1 (Cost Simulation Lemma and Upper Bound)**.** *Let the $\mathcal{F}$-induced IPM be defined as*

$$d_{\mathcal{F}}(\hat{T}(s,a), T(s,a)) \coloneqq \sup_{f \in \mathcal{F}} |\mathbb{E}_{s' \sim \hat{T}(s,a)}[f(s')] - \mathbb{E}_{s' \sim T(s,a)}[f(s')]| \tag{8}$$

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

$$\sum_{s,a} (\rho^{\pi}_{T}(s,a) - \rho^{\pi}_{\hat{T}}(s,a)) c(s,a) \leq \gamma\beta \sum_{s,a} \rho^{\pi}_{\hat{T}}(s,a) d_{\mathcal{F}}(\hat{T}(s,a), T(s,a)) \tag{9}$$

Proof. Using the telescoping lemma [29, 32], we have that

$$\begin{aligned}
&\frac{1}{1-\gamma} \sum_{s,a} (\rho^{\pi}_{T}(s,a) - \rho^{\pi}_{\hat{T}}(s,a)) c(s,a) \\
=&\gamma \sum_{s,a} \rho^{\pi}_{\hat{T}}(s,a) \left[\mathbb{E}_{s' \sim T(s,a)} V^{\pi}_{T}(s') - \mathbb{E}_{s' \sim \hat{T}(s,a)} V^{\pi}_{\hat{T}}(s')\right]
\end{aligned}$$

Then, by Assumption 4.1, we have that

$$\begin{aligned}
&\gamma \sum_{s,a} \rho^{\pi}_{\hat{T}}(s,a) \left[\mathbb{E}_{s' \sim T(s,a)} V^{\pi}_{T}(s') - \mathbb{E}_{s' \sim \hat{T}(s,a)} V^{\pi}_{\hat{T}}(s')\right] \\
\leq&\gamma \sum_{s,a} \rho^{\pi}_{\hat{T}}(s,a) \sup_{f \in \beta\mathcal{F}} \left|\mathbb{E}_{s' \sim \hat{T}(s,a)}[f(s')] - \mathbb{E}_{s' \sim T(s,a)}[f(s')]\right| \\
\leq&\gamma \sum_{s,a} \rho^{\pi}_{\hat{T}}(s,a) \beta d_{\mathcal{F}}(\hat{T}(s,a), T(s,a))
\end{aligned}$$

Putting everything together, we have that

$$\begin{aligned}
&\sum_{s,a} (\rho^{\pi}_{T}(s,a) - \rho^{\pi}_{\hat{T}}(s,a)) c(s,a) \\
\leq&\gamma\beta \sum_{s,a} \rho^{\pi}_{\hat{T}}(s,a) d_{\mathcal{F}}(\hat{T}(s,a), T(s,a))
\end{aligned}$$

□

Theorem A.2 (Tabular Case High-Probability Feasibility Guarantee)**.** *Assume $\mathcal{F} = \{f : \|f\|_{\infty} \leq 1\}$ and that Assumption 4.1 holds. Define $u(s,a) \coloneqq \sqrt{\frac{|\mathcal{S}|}{8n(s,a)} \ln \frac{4|\mathcal{S}||\mathcal{A}|}{\delta}}$, 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)*.*

Proof. In order for a policy that is feasible for Eq (5) is also feasible for Eq (2), we need to have

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

By the lemma above, this is equivalent to having $u(s,a) \geq d_{\mathcal{F}}(\hat{T}(s,a), T(s,a)), \forall s, a$. Since, we assume $\mathcal{F} = \{f : \|f\|_{\infty} \leq 1\}$, this implies

$$\begin{aligned}
&d_{\mathcal{F}}(\hat{T}(s,a), T(s,a)) \\
=&d_{\text{TV}}(\hat{T}(s,a), T(s,a)) \\
=&\frac{1}{2} \left\|\hat{T}(s,a), T(s,a)\right\|_1
\end{aligned}$$

where the last step follows because $\hat{T}(s,a)$ and $T(s,a)$ are multinomial distributions, which are countable. Then, we need

$$u(s,a) \geq \frac{1}{2} \max_{s,a} \left\| \hat{T}(s,a), T(s,a) \right\|_1 \tag{10}$$

By Hoeffding's inequality and the l_1 concentration bound for multinomial distribution, we have that, for any $\delta > 0$, we can set $u(s,a) := \sqrt{\frac{|\mathcal{S}|}{8n(s,a)} \ln \frac{4|\mathcal{S}||\mathcal{A}|}{\delta}}$, then Eq (10) will hold with probability $1-\delta$, completing the proof. □

Corollary A.3 (High-Probability Zero-Training-Violations Guarantee)**.** *Assume the same set of assumptions as Theorem A.2 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).

Proof. Since we want all K intermediate solutions to be feasible with probability $1-\delta$, the fault tolerance for any individual intermediate solution is δ/K; this follows from an union bound argument. Therefore, we can adjust the concentration bound from Hoeffding's inequality by a factor of K and obtain that by setting $u(s,a) := \sqrt{\frac{|\mathcal{S}|}{8n(s,a)} \ln \frac{4K|\mathcal{S}||\mathcal{A}|}{\delta}}$, with probability $1-\delta$, we can guarantee all intermediate solutions to Eq (5) are feasible for Eq (2). □

B CAP with Linear Programming

This implementation of CAP is described in detail in the main text. Here, we describe the exponential search mechanism we use to initialize κ for the very first training episode. Starting with a high value for κ (e.g., 10), we use it to construct a new constrained optimization problem of form Eq (5) and attempt to solve it. If the problem is infeasible, then we halve the value of κ and repeat the process. We stop at the first value of κ for which the problem is feasible, and this value is taken as the initialized κ value.

C CAP with Constrained Cross Entropy Method

In Algorithm 2, we provide the pseudocode for CAP implemented using constrained cross entropy method. Here, we reiterate the algorithm description from the main text for completeness. 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).

D Gridworld Experimental Detail

The gridworld environment is of size 8×8. The action space consists of the four directional primitives: Up, Down, Left, Right. For each action, there is a 20% chance that slippage occurs and the agent moves in a random direction, introducing stochastic transitions to the environment. The reward and the cost functions are randomly generated Bernoulli distributions drawn according to a Beta(1,3) prior. Each state has uniform probability of being selected as the initial state for each episode. The discount rate is 0.99. The cost threshold is kept at 0.1 for all trials. Training lasts 30 episodes, and we use Gurobi [39] as the LP solver in our implementation.

For the gridworld experiments, we also pursue a more aggressive way of updating κ. After observing $J_c(\pi_t)$ for episode t, we set $\kappa := \frac{(J_c(\pi_t)-C)_+}{\sum_{s,a} \rho^{\pi_t}_{\hat{T}(s,a)} n(s,a)}$; this amounts to a proportional PID controller.

Algorithm 2: CAP with Constrained Cross Entropy Method

```
**Inputs:** Transition model estimate $\hat{T}_\theta$, experience buffer $\mathcal{D}$, cost limit $C$
**CCEM Hyperparameters:** Population size $N$, elite population size $E$, max iteration $I$,
planning horizon $H$, initial sampling distribution $\mathcal{N}(\mu_0, \Sigma_0)$
for $i = 1, \ldots, I$ do
    Sample $N$ action sequences $A^1 := \{a_t^1\}_{t=1}^H, \ldots, A^N := \{a_t^N\}_{t=1}^H \sim \mathcal{N}(\mu_{i-1}, \Sigma_{i-1})$
    Evaluate the action sequences using Eq (7) by simulating trajectories in $\hat{T}_\theta$
    Construct feasible set $\mathcal{X} := \{A^n | \tilde{J}_c(A^n) \leq C, n \in [N]\}$
    if $|\mathcal{X}| < E$ then
        Construct elite set $\mathcal{E} :=$ { The $E$ sequences out of all $\{A^n\}_{n=1}^N$ with lowest costs }
    else
        Construct elite set $\mathcal{E} :=$ {The $E$ sequences in $\mathcal{X}$ with highest rewards }
    end if
    Compute $\mu_i, \Sigma_i$ using Maximum Likelihood over $\mathcal{E}$
end for
**Outputs:** Optimal action sequence $\{a_1^*, ..., a_H^*\} := \mu_I$
```

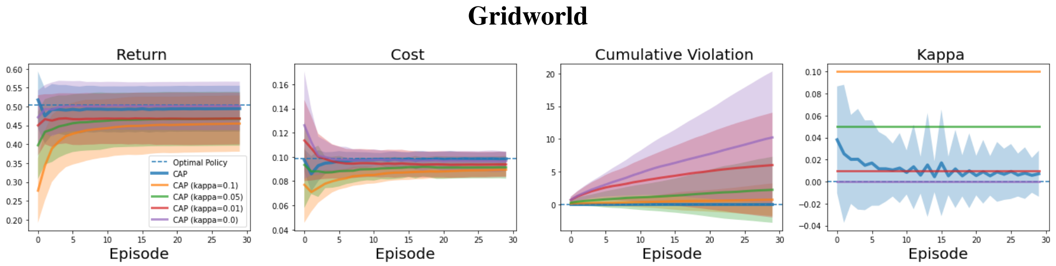

Figure 4: **Tabular gridworld results with standard deviations.**

D.1 Additional results

In Figure 4, we illustrate the full version of Figure 1 with one standard deviation error bars added in. In Table 2, we also show these results in table format. As shown, CAP ablations with fixed κ values exhibit greater variance in their performances over 100 random seeds; this supports the claim in the main text that fixed κ values are more sensitive to the randomness in the environment distribution.

E High-Dimensional Environments Experimental Detail

E.1 Environments

- **Velocity Constrained HalfCheetah:** The state space is 17-dimensional and the action space is 6-dimensional. We use the original environment reward, $v - \frac{1}{10}a^T a$, v is the forward velocity. The cost is $|v|$ [28], meaning that there is a direct trade-off between cost and reward. The cost limit is set to 152, half of the average speed of an unconstrained PPO expert agent [28].

Method	**Gridworld**			
	Kappa κ	Return	Cost (Limit 0.1)	Violations
CAP	Adaptive	0.49 ± 0.06	0.10 ± 0.01	0.00 ± 0.00
CAP	0.1	0.45 ± 0.07	0.09 ± 0.01	0.70 ± 2.48
CAP	0.05	0.47 ± 0.07	0.09 ± 0.01	2.22 ± 5.05
CAP	0.01	0.47 ± 0.07	0.09 ± 0.01	6.00 ± 8.01
CAP	0.0	0.50 ± 0.06	0.10 ± 0.01	10.23 ± 10.08

Table 2: **CAP ablations results on Gridworld.**

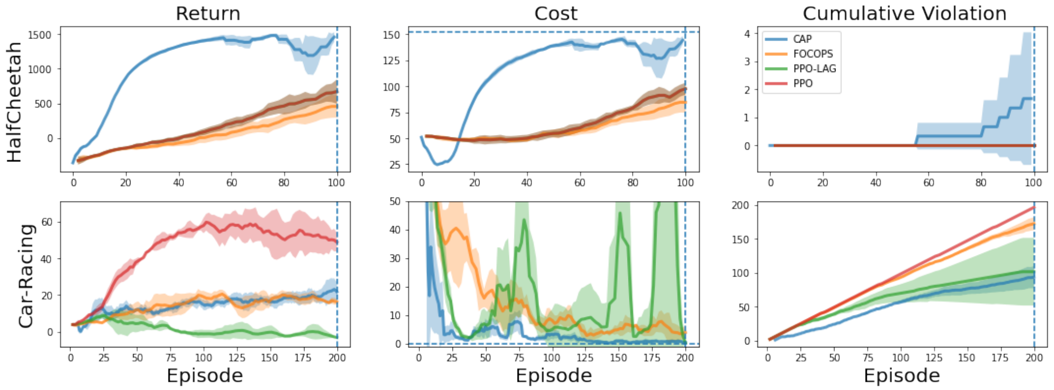

Figure 5: **Step 100k/200k CAP and model-free baselines results on HalfCheetah (top) and Car-Racing (bottom).**

		HalfCheetah			Car-Racing		
Method	Kappa κ	Return	Cost (Limit 152)	Cost Violation	Return	Cost (Limit 0)	Cost Violation
CAP	Adaptive	1456.3	144.3	1.7	21.7	0.4	93.3
CAP	10.0	-36.5	5.4	0.0	1.0	0.4	52.0
CAP	1.0	1092.9	111.5	0.0	6.2	0.4	30.3
CAP	0.1	1774.4	179.9	70.0	35.4	2.3	149.0
CAP	0.0	1588.0	198.1	80.0	26.9	9.3	184.0
CEM	N/A	2330.7	344.0	78.7	40.3	202.1	194.3

Table 3: **CAP ablations results on HalfCheetah and Car-Racing.**

- **Constrained Car-Racing:** The state space is a top down image of the car and the surrounding track. We downscale the image to 64 by 64 by 3. For model-free baselines, we also stack the last 4 frames, as common in reinforcement learning on image based environments. The action space is three dimensional, controlling steering, acceleration and braking. Each value is continuous and bounded. We use an action repeat of 2 to produce a better signal to the model [10]. We keep the original reward, which incentivizes the agent to drive through as many tiles as possible. We use a binary cost that is 1 if the car skids. Skidding is a part of the original environment; a wheel skids if it's force exceeds the friction limit, which is different on grass and road surfaces.

E.2 Uncertainty Estimators

State-based environments: We model the environment transition function using an neural ensemble of size N, where network's output neurons parameterize a Gaussian distribution $\hat{T} = \mathcal{N}(\mu(s_t, a_t), \Sigma(s_t, a_t)$ [9]. We 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 [29].

Image-based environments: We implement PlaNet [10], which models the environment transition function using a latent dynamics model with deterministic and stochastic transition states; we refer interested readers to the original paper for details. PlaNet does not provide an uncertainty estimate because it only utilizes a single transition model. To obtain an uncertainty estimate, we train a bootstrap ensemble of one-step hidden-state dynamics model as in [34]. Each one-step model in the ensemble predicts, from each deterministic state h, the next stochastic state. We formulate our uncertainty estimator as $u(h,a) = Var(\mu_i(h,a)|i = [1..K])$, the variance of ensemble predictions $\{\mu_i\}_{i=1}^{K}$. As in [34], to keep the scale of this uncertainty estimator similar to that of state-based uncertainty estimator, we multiply it by 10000.

E.3 Network Architecture

We use a neural network C to approximate the environment's true cost function. When the cost is continuous, the network's output neurons parameterize a Gaussian distribution and we construct our conservative cost function as $C(s,a) + \kappa u(s,a)$. When the cost is binary, the network outputs a logit and we construct our conservative cost function as $\mathbb{1}[C(s,a) + u(s,a) > 0]$.

To apply model free algorithms on imaged-based environments, we used a shared CNN module to encode the image input. The network consists of 5 convolutional layers followed by a ReLU non-linearity.

```
4x4 conv, 8, stride 2
3x3 conv, 16, stride 2
3x3 conv, 32, stride 2
3x3 conv, 64, stride 2
3x3 conv, 128, stride 1
```

E.4 Hyperparameters

In Table 4, we include the hyperparameters we used for state-based and image-based experiments, respectively.

Hyperparameter	State-based	Image-based
Ensemble size K	5	5
Optimizer	Adam	Adam
Optimizer κ	Adam	Adam
Learning rate	0.001	0.001
Learning rate κ	0.1	0.01
Initial κ	1.0	0.1
Reward discount factor γ	0.99	0.99
Cost discount factor γ_{cost}	*0.99	*0.99
Batch size	256	50
Exploration steps	1000	5000
Experience buffer size	1000000	1000000
Uncertainty multiplier	1	100000
CEM Hyperparameters		
Planning horizon H	30	12
Max iteration I	5	10
Population size N	500	1000
Elite population size E	50	100

Table 4: **CAP hyperparameters**

* We set cost discount factor to 1.0 when the cost is binary, so total cost per episode is directly interpretable.

E.5 Additional results

In Figure 5, we illustrate the training curves of CAP and model free baselines in HalfCheetah and Car-Racing. For clarity, we focus on the first 100K/200K steps. The results are also presented in Table 1. In HalfCheetah, all model free methods have 0 cost violations in the first 100K steps, this is because they have not learnt a running gait that can violate the speed costraint. On the other hand, CAP is able to quickly a gait and keep cost below the limit, with less than two violations per 100 episodes. In CarRacing, all methods have high cost violations because the cost limit is 0. An initial random policy will violate the cost constraint and exploration will always risk violation. Even still, we see that CAP dominates FOCOPS, obtaining better episode return with lower cost and total violations. CAP has more cost violations than PPO-Lagrangian, but we see that this is because PPO-Lagrangian degrades to a trivial policy that maintains a stationary position, obtaining negative return with minimal risk of cost violations.

E.6 Compute resources

We use a single GTX 2080 Ti with 32 cores to run our experiments, each run takes about 10 hours in clock time.