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*

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

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

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

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

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

Then, by Assumption 4.1, we have that

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

Putting everything together, we have that

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

□

Theorem A.2 (Tabular Case High-Probability Feasibility Guarantee)**.** *Assume $\mathcal{F} = \{f : \|f\|_\infty \le 1\}$ 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).*

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

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

By the lemma above, this is equivalent to having $u(s,a) \ge d_{\mathcal{F}}(\hat{T}(s,a),T(s,a)), \forall s,a$. Since, we assume $\mathcal{F} = \{f : \|f\|_\infty \le 1\}$, 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 $i = 1, \dots, I$ do
    Sample $N$ action sequences $A^1 := \{a_t^1\}_{t=1}^H, \dots, A^N := \{a_t^N\}_{t=1}^H \sim \mathcal{N}(\mu_{i-1}, \Sigma_{i-1})$
    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 $\{a_1^*, ..., a_H^*\} := \mu_I$
```

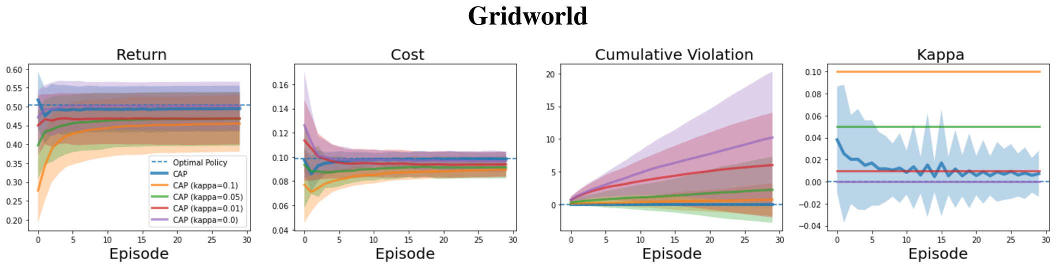

Figure 4: **Tabular gridworld results with standard deviations.**

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	0.494 ± 0.061	0.098 ± 0.006	0.0 ± 0.0
CAP	0.1	0.454 ± 0.074	0.089 ± 0.0055	0.7 ± 2.48
CAP	0.05	0.468 ± 0.071	0.091 ± 0.0093	2.22 ± 5.05
CAP	0.01	0.468 ± 0.069	0.0938 ± 0.0104	6.0 ± 8.01
CAP	0.0	0.50 ± 0.064	0.0965 ± 0.0079	10.23 ± 10.08

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

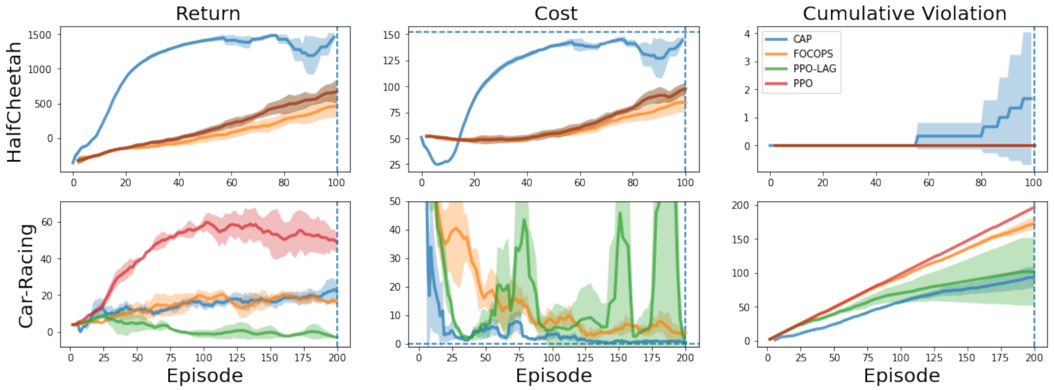

Figure 5: **Step 100k/200k CAP and model-free baselines results on HalfCheetah (top) and Car-Racing (bottom).**

		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 $u(s, a) = \max_{i=1}^{N} \left\| \Sigma_\theta^i(s, a) \right\|_F$ 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.