UNCERTAINTY-AWARE CONSTRAINT INFERENCE IN INVERSE CONSTRAINED REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Aiming for safe control, Inverse Constrained Reinforcement Learning (ICRL) considers inferring the constraints respected by expert agents from their demonstrations and learning imitation policies that adhere to these constraints. While previous ICRL works often neglected underlying uncertainties during training, we contend that modeling these uncertainties is crucial for facilitating robust constraint inference. This insight leads to the development of an Uncertainty-aware Inverse Constrained Reinforcement Learning (UAICRL) algorithm. Specifically, 1) aleatoric uncertainty arises from the inherent stochasticity of environment dynamics, leading to constraint-violating behaviors in imitation policies. To address this, UAICRL constructs risk-sensitive constraints by incorporating distributional Bellman updates into the cumulative costs model. 2) Epistemic uncertainty, resulting from the model's limited knowledge of Out-of-Distribution (OoD) samples, affects the accuracy of step-wise cost predictions. To tackle this issue, UAICRL develops an information-theoretic quantification of the epistemic uncertainty and mitigates its impact through flow-based generative data augmentation. Empirical results demonstrate that UAICRL consistently outperforms other baselines in continuous and discrete environments with stochastic dynamics.

1 INTRODUCTION

Reinforcement Learning (RL) is an effective technique for solving sequential decision-making problems, typically focusing on maximizing cumulative rewards. However, recent studies (Liu et al., 2021; Satija et al., 2020; Yang et al., 2023) have argued that, in order to achieve safe control, the optimal policy must adhere to the underlying constraints in the environment. For instance, in an open-road environment, an autonomous driving policy must comply with traffic rules and social norms. Explicitly specifying these constraints is challenging. A more practical alternative is to infer the constraints from expert demonstrations by analyzing the behavioral patterns of expert agents.

To achieve this goal, Inverse Constrained Reinforcement Learning (ICRL) (Malik et al., 2021) extends the Maximum Entropy Inverse Reinforcement Learning (MEntIRL) framework (Ziebart et al., 2008) to infer constraints (rather than rewards) from expert demonstrations. ICRL alternates between Constrained Reinforcement Learning (CRL) and Inverse Constraint Inference (ICI) until the imitation policy can reproduce the expert demonstrations. During the process, traditional ICRL algorithms (Scobee & Sastry, 2020; Malik et al., 2021; Gaurav et al., 2023; Liu & Zhu, 2022) often assume deterministic training environments, without considering the influence of underlying uncertainties. Specifically, in the CRL phase, stochastic transition functions introduce *aleatoric uncertainty*, thereby influencing the policy update in CRL and may result in constraint-violating behaviors. In pursuit of safety control, human experts often exhibit risk-averse behaviors, so an imitating agent cannot accurately replicate expert demonstrations unless it formulates a risk-sensitive policy. In the ICI phase, due to the finite size of the training data, *epistemic uncertainty* arises when game contexts lie outside the data distribution, leading to inaccurate cost predictions.

To achieve uncertainty-aware constraint inference, (Liu et al., 2023; Papadimitriou et al., 2023) proposed modeling the posterior distribution of step-wise constraints using variational inference and Monte Carlo sampling. (McPherson et al., 2021; Baert et al., 2023) incorporated maximum causal entropy likelihood into ICRL. Although causal entropy and constraint distribution are sensitive to aleatoric and epistemic uncertainties respectively, none of the previous methods can handle both uncertainties. Additionally, many of them (Papadimitriou et al., 2023; McPherson et al., 2021) are restrictive to discrete spaces, which limits their robustness and scalability in real-world applications.

In this paper, we introduce the Uncertainty-aware Inverse Constrained Reinforcement Learning (UAICRL), a novel ICRL framework that models both the aleatoric and epistemic uncertainties for achieving robust constraint inference. Figure 1 provides an overview of UAICRL. Specifically,

1) We design a risk-sensitive constraint by modeling the distribution of cumulative costs and distorting this distribution to represent risk measures. The predicted distribution is sensitive to aleatoric uncertainty and we empirically justify these findings in our experiments. To enable efficient risk-sensitive control in continuous spaces, we propose the Distributional Lagrange Policy Optimization (DLPO), which incorporates distributional estimation and Lagrange mechanics into the classic Proximal Policy Optimization (PPO) (Schulman et al., 2017) for constrained policy optimization.

2) We introduce a mutual-information-driven metric (Gabrié et al., 2018) to quantify epistemic uncertainty in constraint inference and propose an information-theoretic ICI objective to minimize the impact of epistemic uncertainty when updating the constraint function. A key technique to achieve this objective involves augmenting the training data using the proposed Flow-based Trajectory Generation (FTG) algorithm. FTG effectively generates a diverse set of trajectories based on the dataset and task-dependent rewards, thus reducing the influence of OoD state-action pairs in constraint prediction.

Empirical evaluations show that UAICRL consistently achieves higher feasible rewards and lower constraint violation rates in stochastic environments with both discrete and continuous state spaces, outperforming other ICRL baselines. For a comprehensive evaluation, we conduct in-depth studies to individually assess how effectively UAICRL handles the aleatoric and epistemic uncertainty.

2 Problem Formulation

In this section, we introduce the Inverse Constrained Reinforcement Learning (ICRL) algorithm that iteratively performs constrained policy updates and inverse constraint inference. Then we define the major challenges induced by the aleatoric and epistemic uncertainties in the process of ICRL.

Constrained Reinforcement Learning (CRL). To solve a CRL problem, the agent optimizes the control policy under a Constrained Markov Decision Processes (CMDPs) $\mathcal{M}^c$, which can be defined by a tuple $(\mathcal{S}, \mathcal{A}, p_\mathcal{T}, p_\mathcal{R}, p_\mathcal{C}, \epsilon, \mu_0, \gamma, T)$ where: 1) $\mathcal{S}$ and $\mathcal{A}$ denote the space of states and actions. 2) $p_\mathcal{T}(s'|s,a)$ and $p_\mathcal{R}(r|s,a)$ define the transition and reward probabilities. 3) $p_\mathcal{C}(c|s,a)$ and ϵ denote the probability of cost and the associated bound. 4) μ_0 defines the initial state distribution. 5) $\gamma \in (0,1)$ is the discount factor and T defines the planning horizon ($T = \infty$ in principle). The goal of the CRL policy π is to maximize expected discounted rewards under the constraint:

$$\arg\max_\pi \mathbb{E}_{\pi,p_\mathcal{T},p_\mathcal{R},\mu_0}\Big[\sum_{t=0}^{T}\gamma^t r(s_t,a_t)\Big] \text{ s.t. } \mathbb{E}_{\pi,p_\mathcal{T},p_\mathcal{C},\mu_0}\Big[\sum_{t=0}^{T}\gamma^t c(s_t,a_t)\Big] \le \epsilon \tag{1}$$

Note that traditional CRL problems often assume the constraint signals are directly observable, but in real-world problems, such constraint signals are not readily available, and we must infer these constraints from the environment by solving the following inverse problem.

Inverse Constraint Inference (ICI). ICRL algorithms (Malik et al., 2021; Gaurav et al., 2023; Liu et al., 2023; Papadimitriou et al., 2023) typically assume that *reward signals are observable and the goal is to infer the constraints* from expert demonstrations. Inspired by (Malik et al., 2021), we define Φ as a Bernoulli feasibility variable that takes two values $\{\phi^+, \phi^-\}$ such that $p(\phi^+|s,a;\omega)$ (ω denotes model parameters) quantifies to what extent performing action a in the state s is feasible while $p(\phi^-|s,a;\omega)$ denotes the probability this movement is infeasible. For clarity, in the rest of this paper, we denote ϕ^+ by ϕ. Accordingly, the step-wise cost is then defined as $c_\omega(s,a) = 1 - p(\phi|s,a;\omega)$. Under these definitions, the likelihood of generating the expert dataset $\mathcal{D}_e$ can be represented as:

$$p(\mathcal{D}_e|\Phi) = \frac{1}{(\mathbb{Z}_{\mathcal{M}^{c_\omega}})^N}\prod_{n=1}^{N}\exp\Big[r(\tau^{(n)})\Big]\,\mathbb{1}^{\mathcal{M}^{c_\omega}}(\tau^{(n)}) \tag{2}$$

where 1) N denotes the number of trajectories in the expert dataset, 2) the normalizing term $\mathbb{Z}_{\mathcal{M}^{c_\omega}} = \int \exp[r(\tau)]\,\mathbb{1}^{\mathcal{M}^{c_\omega}}(\tau)\mathrm{d}\tau$, and 3) the identifier $\mathbb{1}^{\mathcal{M}^{c_\omega}}(\tau^{(n)})$ can be defined as $p(\phi|\tau^{(n)};\omega) = \prod_{t=1}^{T} p(\phi|s_t^{(n)}, a_t^{(n)};\omega)$. By substituting them to Equation (2), we can update ω by the gradient of the log-likelihood function (Malik et al., 2021):

$$\nabla_\omega \log[p(\mathcal{D}_e|\Phi)] = \sum_{n=1}^{N}\Big[\nabla_\omega \sum_{t=0}^{T}\log[p(\phi|s_t^{(n)}, a_t^{(n)};\omega)]\Big] - N\mathbb{E}_{\hat{\tau}\sim\pi_{\mathcal{M}^{c_\omega}}}\Big[\nabla_\omega \sum_{t=0}^{T}\log[p(\phi|\hat{s}_t,\hat{a}_t;\omega)]\Big] \tag{3}$$

where $\hat{\tau}$ is sampled from the current imitation policy $\pi_{\mathcal{M}^{c_\omega}}$. Intuitively, our goal is to differentiate the trajectories generated by expert policies and imitation policies that may violate the constraints. When the imitation policy under the learned constraint model c_ω matches the expert policy, this gradient goes to 0, and the update stops.

Modeling Uncertainty in ICRL. An ICRL agent solves the CRL and ICI problems iteratively until the imitation policy can reproduce expert demonstrations. However, existing ICRL algorithms typically assume a deterministic setting without considering uncertainties, but we argue that the aleatoric and epistemic uncertainties can emerge and significantly affect the training and the convergence of ICRL. To address them, we propose the 1) *Distributional Lagrange Policy Optimization (DLPO)* algorithm with a risk-sensitive constraint by modeling the distribution of the cumulative costs (Section 3), and 2) *Data-augmented Constraint Inference* with flow-based trajectory generation for constraint inference from limited demonstrations (Section 4). Figure 1 shows the flowchart of UAICRL framework.

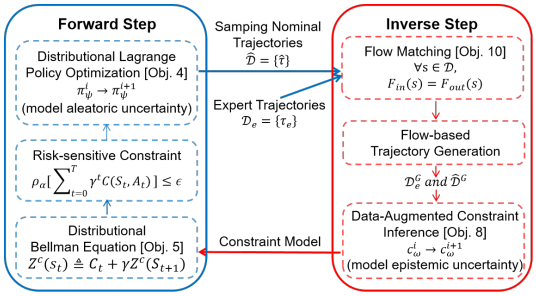

Figure 1: The flowchart of UAICRL.

3 Policy Optimization with Risk-Sensitive Constraint

In order to address the aleatoric uncertainty, risk-sensitive RL commonly incorporates the risk measure into the objective of optimization, for example, by constructing $\arg\max_\pi \rho_\alpha^\pi[\sum_{t=0}^{T} \gamma^t O(S_t, A_t)]$ where O refers to the objective, $\alpha \in [0, 1]$ defines the confidence level and ρ denotes the risk measure. This approach imposes significant costs on the policies that may lead to hazardous consequences, even if the likelihood of such events is low (e.g., a pedestrian being hit by a car on the highway). While safety is important, being too conservative can harm performance and hinder reward accumulation (Mihatsch & Neuneier, 2002; Choi et al., 2021).

Risk-Sensitive Constraint for Policy Optimization. In this work, we present a novel approach that disentangles the unified objective into two components: rewards and costs, where rewards indicate the desired outcome, while costs account for negative consequences. For example, in autonomous driving, rewards might correspond to how quickly a car reaches its destination, while costs capture the need to adhere to traffic rules and safety constraints during driving. To drive a risk-sensitive policy, the risk incurred by aleatoric uncertainty can be incorporated separately from reward optimization. We formulate the trade-off between rewards and costs as a constrained optimization problem:

$$\arg\max_\pi \mathbb{E}_{\pi, p_\mathcal{T}, p_\mathcal{R}, \mu_0}\Big[\sum_{t=0}^{T}\Big(\gamma^t r(s_t, a_t) + \beta\gamma^t \mathcal{H}[\pi(a_t|s_t)]\Big)\Big] \text{ s.t. } \rho_\alpha\Big[\sum_{t=0}^{T}\gamma^t C(S_t, A_t)\Big] \le \epsilon \quad (4)$$

where $\mathcal{H}[\pi(a_t|s_t)]$ refers to the causal entropy (Ziebart et al., 2010) and $C(\cdot)$ denotes the random variable of state-action cost [1]. To specify the risk measure, we define the trajectory-generating probability as $p^\pi(\tau) = \mu_0(s_0)\prod_{t=0}^{T-1}\pi(a_t|s_t)p_\mathcal{T}(s_{t+1}|s_t, a_t)$, where aleatoric uncertainty accumulates in sequential decision-making due to the inherent stochasticity in the initial state distribution $\mu_0(s_0)$, policy π, and transition function $p_\mathcal{T}(s_{t+1}|s_t, a_t)$. We define the corresponding risk envelope $\mathcal{U}_\alpha^\pi = \{\zeta_\alpha : \Gamma \to [0, \frac{1}{\alpha}] | \sum_{\tau\in\Gamma}\zeta(\tau)p^\pi(\tau) = 1\}$ to be a compact, convex, and bounded set, based on which the risk measure can be induced by the distorted probability distribution $p_\zeta^\pi = \zeta \cdot p^\pi$. For example, the Conditional Value-at-Risk (CVaR) can be defined as $\rho_\alpha^\pi[\sum_{t=0}^{T}\gamma^t C_t] = \sup_{\zeta_\alpha\in\mathcal{U}_\alpha^\pi}\mathbb{E}_{\tau\sim p^\pi}[\zeta_\alpha(\tau)\sum_{t=0}^{T}\gamma^t C_t]$.

Distributional Estimator for Cumulative Cost. The difficulty of constructing the distorted probability-based risk measure lies in the unknown distribution of the cumulative costs. To estimate the distribution, we define the variable of discounted cumulative costs as $Z^c(s_t) = \sum_{\iota=0}^{T-t}\gamma^\iota C_\iota | S_0 = s_t$. Based on (Luo et al., 2022), we use N supporting quantiles parameterized with the rational-quadratic function to represent Z_θ^c, where θ denotes model parameters. Then we can update Z_θ^c through quantile regression (Bellemare et al., 2017).

[1] Throughout this paper, we use uppercase letters (e.g., C) to represent random variables and lowercase letters (e.g., c) to represent instances of variables. For brevity, we use C_t as a shorthand of $C(S_t, A_t)$

During training, when the agent performs an action $a \sim \pi(\cdot|s)$ at state s, the agent receives a cost $c \sim p_{\mathcal{C}}(\cdot|s,a)$ and moves to a future state $s' \sim p_{\mathcal{T}}(\cdot|s,a)$. This stochastic process can be captured by the distributional Bellman equation (state-form) (Gerstenberg et al., 2023):

$$Z^c_\theta(s) = \int_{a\in\mathcal{A}} \pi(a|s) \int_{s'\in\mathcal{S}} p_{\mathcal{T}}(s'|s,a) \int_{c\in\mathcal{C}} p_{\mathcal{C}}(c|s,a)(b_{c,\gamma})_{\#} Z^c_\theta(s') \, \mathrm{d}a \, \mathrm{d}s' \, \mathrm{d}c \tag{5}$$

where the pushfoward operation $(b_{c,\gamma})_{\#}$ involves shifting and scaling the distribution by c and γ. For brevity, we denote the distributional Bellman equation as $Z^c(s) \overset{\Delta}{:=} C(s,A) + \gamma Z^c(S')$. We show that the predicted distribution is sensitive to the aleatoric uncertainty.

Proposition 1. *(Liu et al., 2022). The key components for representing the aleatoric uncertainty can be captured by the distributional Bellman equation under the measure of entropy.*

The detailed proof is in Appendix A.1. Leveraging the aforementioned policy optimization objective (4) and distributional estimator (5), we design the Distributional Lagrange Policy Optimization (DLPO) algorithm to learn the policy under risk-sensitive constraint. The implementation details are shown in Algorithm 2 in Appendix B.1.

4 Constraint Inference with Flow-based Data Augmentation

Epistemic uncertainty arises due to the limited training data and the model's lack of knowledge about Out-of-Distribution (OoD) data. A common measure of epistemic uncertainty is the mutual information $I(\omega; y|\boldsymbol{x}, \mathcal{D})$ (Smith & Gal, 2018; van Amersfoort et al., 2020), which quantifies the amount of information gained by the model ω when it observes the true label y for a given input $\boldsymbol{x}$. The greater the uncertainty of the model regarding the data, the more additional information it can obtain once the true label y is observed.

Intuitively, epistemic uncertainty arises when the constraint model ω is required to predict the cost of a trajectory $\bar{\tau}$ that locates OoD of training data (i.e., predict $c(\bar{\tau}) = 1 - p(\phi|\bar{\tau};\omega)$), which is generated by exploratory behaviors during policy updates. To reduce the effect of epistemic uncertainty, we need to minimize the mutual information $I(\omega; \Phi|\bar{\tau}, \mathcal{D})$, which can be represented as follows:

Proposition 2. *Let $\mathcal{D}$ denote the training dataset consisting of expert trajectories $\{\tau_e\}$ and imitation trajectories $\{\hat{\tau}\}$. Let $\bar{\tau}$ denote an OoD trajectory. Let ω denote the constraint model parameters and $q(\omega)$ denote the dropout distribution. $I(\omega; \Phi|\bar{\tau}, \mathcal{D})$ can be empirically represented by:*

$$\mathcal{H}[p(\Phi|\bar{\tau}, \mathcal{D})] - \frac{1}{M}\sum_m \mathcal{H}[p(\Phi|\bar{\tau};\omega_m)] \text{ where } \omega_m \sim q(\omega) \tag{6}$$

The proof is in Appendix A.2. Specifically, 1) $\mathcal{H}[p(\Phi|\bar{\tau}, \mathcal{D})] \in [0, \infty)$ measures the amount of information required to describe the feasibility Φ of an exploratory trajectory $\bar{\tau}$ based on the given training dataset $\mathcal{D}$. 2) $\mathcal{H}[p(\Phi|\bar{\tau}, \omega_m)]$ defines the entropy of constraint model parameterized by ω_m. To reduce the epistemic uncertainty, we integrate them into our ICI objective as follows.

4.1 Data-Augmented Inverse Constraint Inference

To reduce the impact of epistemic uncertainty, we can augment the conventional ICI objective (Obj. 3) with the mutual information term $I(\omega; \Phi|\bar{\tau}, \mathcal{D})$. Since $I(\omega; \Phi|\bar{\tau}, \mathcal{D})$ is intractable, based on Proposition 2, we instead maximize the following objective with its empirical representation:

$$\frac{1}{M}\sum_m \mathbb{E}_{\mathcal{D}_e}\Big[\sum_{t=0}^{T} \log[p(\phi|s^e_t, a^e_t;\omega_m)]\Big] - \mathbb{E}_{\hat{\mathcal{D}}}\Big[\sum_{t=0}^{T} \log[p(\phi|\hat{s}_t, \hat{a}_t;\omega_m)]\Big] - \alpha\Big(\mathcal{H}[p(\Phi|\bar{\tau}, \mathcal{D})] - \mathcal{H}[p(\Phi|\bar{\tau};\omega_m)]\Big) \tag{7}$$

where α controls the trade-off between the log-likelihood and mutual information, and we use dropout layers (Srivastava et al., 2014) for $q(\omega)$ by following (Smith & Gal, 2018). Besides, the conditional entropy $\mathcal{H}[p(\Phi|\bar{\tau}, \mathcal{D})]$ is independent of ω, which reaches zero (its minimum) when $p(\Phi, \bar{\tau}, \mathcal{D}) = p(\mathcal{D})$, indicating that the dataset has already recorded the trajectory and its feasibility (i.e., $(\Phi, \bar{\tau}) \in \mathcal{D}$). To achieve the goal, we expand the dataset by generating trajectories $\{(\Phi^G, \tau^G)\}$, obtaining the augmented expert and nominal dataset $\mathcal{D}^G_e$ and $\hat{\mathcal{D}}^G$. By substituting them to objective (7), we get the following data-augmented constraint inference objective:

$$\frac{1}{M}\sum_m \mathbb{E}_{\mathcal{D}^G_e}\Big[\sum_{t=0}^{T} \log[p(\phi|s^e_t, a^e_t;\omega_m)]\Big] - \mathbb{E}_{\hat{\mathcal{D}}^G}\Big[\sum_{t=0}^{T} \log[p(\phi|\hat{s}_t, \hat{a}_t;\omega_m)]\Big] + \alpha\mathcal{H}[p(\Phi|\bar{\tau};\omega_m)] \tag{8}$$

We propose an approach that can generate a diverse set of expert and nominal trajectories as follows.

4.2 FLOW-BASED TRAJECTORY GENERATION

We propose a Flow-based Trajectory Generation (FTG) algorithm to perform conditional generation for maximizing $p(\Phi, \bar{\tau}|\mathcal{D})$. Unlike the traditional trajectory generation with policy rollout, FTG learns the transition densities from the trajectory dataset $\mathcal{D}$. Moreover, the generation is guided by the feasibility $\Phi = \{\phi, \phi^-\}$ of trajectories, rather than the reward-maximizing policy.

To learn FTG, we define the non-negative trajectory flow function $F : \mathcal{T} \to \mathcal{R}^+$. Intuitively, $F(\tau)$ quantifies the mass of particles (Bengio et al., 2021) passing through τ, and denser particles indicate a higher probability of generating the trajectory. Under this setting, the state flow $F(s)$ is the integral of trajectory flows passing through this state: $F(s_t) = \int_{\tau \ni s_t} F(\tau) \mathrm{d}\tau$. The transition probabilities can be defined as $p_F(s_{t+1}|s_t) = \frac{F(s_t \to s_{t+1})}{F(s_t)}$ where $F(s_t \to s_{t+1}) = \int_{\tau=(\dots, s_t \to s_{t+1}, \dots)} F(\tau)$ embeds flow pass through action a_t. The trajectory generation probability $p_F(\tau) = \prod_{t=0}^{T-1} p_F\left(s_{t+1}|s_t\right)$. In order to generate $\{\Phi^G, \tau^G\}$, the generated trajectories τ must correspond to a specific ϕ. This is achieved by a careful design of the reward function. The reward functions for generating the expert ($\Phi = \phi$) and nominal trajectories ($\Phi = \phi^-$) can be set as:

$$R_e(s_t) = \begin{cases} 1, & t = T \wedge s_t \in \tau_e \\ 0, & \text{otherwise} \end{cases} \quad \text{and} \quad \hat{R}(s_t) = \begin{cases} 1, & t = T \wedge s_t \in \hat{\tau} \\ 0, & \text{otherwise} \end{cases} \tag{9}$$

To generate expert trajectories, we assign a positive reward (+1) to the terminated state of the recorded expert trajectories $\tau_e \in \mathcal{D}_e$ and 0s to others. The same approach is applied for generating nominal trajectories. We train two flow functions F_e and $\hat{F}$ by utilizing $R_e(s_t)$ and $\hat{R}(s_t)$ respectively.

Learning Flow Functions. A common approach to updating the flow function $F(\cdot)$ is flow matching: for a state $s_t \in \mathcal{S}$, the inflows $\int_{a:\mathcal{T}(s,a)=s_t} F(s,a)\mathrm{d}a$ must match outflows $\int_{a\in\mathcal{A}} F(s_t,a)\mathrm{d}a$. To enable flow matching in continuous environments, we sample M actions independently and uniformly from the continuous action space $\mathcal{A}$. These actions serve as an approximation of flows that can be utilized to train the flow network. The resulting loss function is approximated as (Li et al., 2023):

$$\mathcal{L}_\xi(\tau) = \sum_{s_t=s_0}^{s_T} \left\{ \log\left[\epsilon + \sum_{m=1}^{M} \exp F_\xi\Big(G(s_t,a_m),a_m\Big)\right] - \log\left[\epsilon + \lambda R(s_t) + \sum_{m=1}^{M} \exp F_\xi(s_t,a_m)\right] \right\}^2 \tag{10}$$

where: 1) ξ denotes the parameters of the flow network $F(\cdot)$. 2) $G(\cdot)$ is a retrieval neural network with (s_{t+1}, a_t) as the input and s_t as the output (i.e., to predict the parent of a state), which can be trained by MSE Loss based on given trajectories. 3) $\sum_{m=1}^{M} F_\xi\left(s_t, a_m\right)$ is an approximation of flows with M sampled actions. 4) ϵ can balance the use of small and large flows, thereby preventing numerical problems when computing the logarithm of extremely small flows.

Trajectory Generation. FTG aims to generate trajectories $\tau^G = (s_0, a_0, \dots, s_T, a_T)$ by starting from the initial state s_0 and iteratively sampling the action a_t based on the scale of the flow $F(s_t, a_t)$, thereby transitioning to the next state s_{t+1}. In this process, actions with larger flow will be sampled with higher probabilities. To label τ^G, we set Φ to ϕ if the generation is based on the expert flows (i.e., F_e, learned with R_e), and ϕ^- for nominal ones. The detailed implementation of FTG is shown in Algorithm 3 in Appendix B.1. Leveraging the modeling capability of the generative model, FTG learns intricate feature patterns of input data, generating trajectories that resemble the training data in the latent space, even if they are absent in the original state-action space. The incorporation of this data significantly mitigates the impact of epistemic uncertainty during constraint inference.

Building upon the aforementioned ideas for modeling aleatoric and epistemic uncertainties, Algorithm 1 presents the whole UAICRL algorithm.

5 RELATED WORKS

Learning constraints from demonstrations. Given the demonstrations, prior works mainly focused on inferring implicit constraints by determining the permissibility of actions under specific states. In *discrete* state-action spaces, (Chou et al., 2020b; Park et al., 2020) learned constraint sets to differentiate feasible from infeasible state-action pairs. (Scobee & Sastry, 2020) proposing inferring the constraint set under the maximum entropy principle, and (McPherson et al., 2021; Baert et al., 2023) extended it to stochastic environments using maximum causal entropy (Ziebart et al., 2010). In *continuous* domains, the goal is to infer boundaries between feasible and infeasible state-action pairs.

Algorithm 1: Uncertainty-aware Inverse Constrained Reinforcement Learning (UAICRL)

Input: Expert dataset $\mathcal{D}_e = \{\tau_e\}$, augmented data size I.
Initialize the cost model c_ω, policy π_ψ, expert flow F^e_ξ, and nominal flow $\hat{F}_{\bar{\xi}}$;
Update F^e_ξ with $\mathcal{D}_e$ by minimizing the flow loss (10);
do
  Update π_ψ based on the risk-sensitive CRL objective (4) and distributional estimator (5);
  Sample nominal trajectories $\hat{\mathcal{D}} = \{\hat{\tau}\}$ with π_ψ, and update $\hat{F}_{\bar{\xi}}$ with $\hat{\mathcal{D}}$ by the flow loss (10);
  Initialize the augmented expert and nominal datasets $\mathcal{D}^G_e = \mathcal{D}_e$ and $\hat{\mathcal{D}}^G = \hat{\mathcal{D}}$;
  for $i = 1, 2, \ldots, I$ **do**
    Generate expert and nominal trajectories τ^G_e and $\hat{\tau}^G$ with F^e_ξ and $\hat{F}_{\bar{\xi}}$;
    Add the generated trajectories to datasets: $\mathcal{D}^G_e = \mathcal{D}^G_e \cup \tau^G_e$ and $\hat{\mathcal{D}}^G = \hat{\mathcal{D}}^G \cup \hat{\tau}^G$;
  end
  Update the cost model c_ω with the ICI objective (8) based on $\mathcal{D}^G_e$ and $\hat{\mathcal{D}}^G$;
while $\hat{\mathcal{D}}$ *do not match* $\mathcal{D}_e$;

(Malik et al., 2021; Gaurav et al., 2023) used neural networks to approximate constraints. Some recent studies (Liu et al., 2023; Chou et al., 2020a; Papadimitriou et al., 2023) applied Bayesian Monte Carlo and variational inference to infer a posterior distribution of constraints in high-dimensional state space. These constraint distributions can only model the epistemic uncertainty.

Uncertainty-Aware Reinforcement Learning. Incorporating uncertainty awareness in RL algorithms is essential for efficient exploration and control, with numerous downstream applications (Hoel et al., 2023; Liu et al., 2022; Wu et al., 2023). Several existing works study defining uncertainty measures in RL environments, primarily using dropout layers (Chen et al., 2017) or ensemble models (Lütjens et al., 2019; An et al., 2021), such as Bootstrapped DQN methods (Osband et al., 2016; Da Silva et al., 2020) and their offline extensions (Kumar et al., 2019; An et al., 2021). Another approach to measuring uncertainty is through Distributional RL (Luo et al., 2022; Bellemare et al., 2017; Dabney et al., 2018; Mavrin et al., 2019), which directly models the distribution of future returns with distributional Bellman operator. Besides, Risk-sensitive RL (Mihatsch & Neuneier, 2002; Chow et al., 2015; Prashanth et al., 2022; Ni & Lai, 2022) is indeed a closely related research area, in which agents employ risk measures, such as exponential utility, variance, and Value-at-Risk, to develop risk-aware policies. In contrast to prior works that apply risk measures for rewards maximization, our work builds upon Constrained MDPs and emphasizes capturing the cost uncertainty.

6 Empirical Evaluation

Experiment Settings. We conduct empirical evaluations utilizing an ICRL benchmark (Liu et al., 2023), and extend it to include stochastic dynamics by incorporating noise into transitions. Following (Malik et al., 2021), evaluation metrics include: 1) *Constraint Violation Rate* measures the probability that a policy violates constraint in a trajectory. 2) *Feasible Cumulative Rewards* calculates the total rewards obtained by the agent before violating any constraints.

Table 1: Baseline methods for constraint inference.

Method	Continues Space	Constraint Optimization	Maximum Entropy	Aleatoric Uncertainty	Epistemic Uncertainty
GACL	✓	✗	✗	✗	✗
B2CL	✓	✓	✗	✗	✗
ICRL	✓	✓	✓	✗	✗
VICRL	✓	✓	✓	✗	✓
UAICRL-NRS	✓	✓	✓	✗	✓
UAICRL-NDA	✓	✓	✓	✓	✗
UAICRL	✓	✓	✓	✓	✓

Comparison Methods. Our experiments include the following baselines: 1) *Generative Adversarial Constraint Learning* **(GACL)** extends Generative Adversarial Imitation Learning (GAIL) (Ho & Ermon, 2016) and employs a discriminator $D(s, a)$ to differentiate validate state-action pairs from infeasible ones. The $logD(s, a)$ is directly appended to the reward function as a penalization. 2) *Binary Classifier Constraint Learning* **(BC2L)** employs a binary classifier as the constraint model without utilizing the Maximum Entropy (MEnt) framework. 3) *Inverse Constrained Reinforcement Learning* **(ICRL)** (Malik et al., 2021) follows the MEnt framework with a risk-neutral constraint. 4) *Variational ICRL* **(VICRL)** (Liu et al., 2023) captures epistemic uncertainty in constraint inference using Beta distribution based on ICRL. Additionally, we perform ablation studies where 5) **UAICRL-NRS** removes the

risk-sensitive constraint from UAICRL and 6) **UAICRL-NDA** removes the data augmentation from UAICRL. Table 1 summarizes these methods.

6.1 Discrete Environment: Stochastic Gridworld

We construct three Gridworld environments with different constraints, where the agent's objective is to navigate from a starting location to a target location while avoiding the added constraints, as shown in Figure 2. The environment exhibits a certain degree of stochasticity, where, with a specific probability (p_s = 0.01 and 0.001), the environment receives a random action instead of an agent's action. To be compatible with the environment settings, we utilize the discretized implementation of UAICRL by following (Liu et al., 2023; Malkin et al., 2022). Appendix B.2 shows the implementation details.

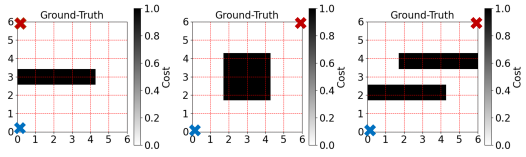

Figure 2: Three different Gridworld settings with blue, red, and black markers indicating the starting, target, and constrained locations respectively.

Table 2 shows the evaluation performance. Check Appendix D.1 for corresponding training curves. We find that in stochastic settings, UAICRL generally outperforms other methods with a relatively lower constraint violation rate and higher rewards, while GACL and ICRL can hardly work well. BC2L and VICRL achieve relatively satisfactory results in the first two settings but perform poorly in the third setting.

Table 2: Evaluation of different methods in the discrete environments with the random rate p_s = 0.01 and 0.001.

		$p_s = 0.01$			$p_s = 0.001$		
		Gridworld Setting 1	Gridworld Setting 2	Gridworld Setting 3	Gridworld Setting 1	Gridworld Setting 2	Gridworld Setting 3
Feasible Rewards	BC2L	0.451	**0.716**	0.125	0.647	0.602	0.192
	GACL	0.032	0.109	0.000	0.011	0.070	0.000
	ICRL	0.244	0.532	0.033	0.356	0.368	0.089
	VICRL	0.537	0.310	0.051	0.778	0.610	0.070
	UAICRL	**0.650**	0.683	**0.359**	**0.797**	**0.739**	**0.401**
Constraint Violation Rate	BC2L	33.0%	19.4%	58.3%	29.4%	27.2%	51.7%
	GACL	43.4%	29.0%	77.9%	67.0%	10.8%	84.0%
	ICRL	53.3%	33.4%	63.3%	35.8%	26.8%	73.3%
	VICRL	35.0%	32.7%	44.6%	18.9%	27.9%	53.2%
	UAICRL	**13.1%**	**9.4%**	**34.2%**	**9.4%**	**7.2%**	**38.3%**

6.2 Virtual Environment: Stochastic MuJoCo

We utilize five MuJoCo environments in the benchmark (Liu et al., 2023) and additionally incorporate Gaussian noise into the transition function as $p_{\mathcal{T}}(s_{t+1}|s_t, a_t) = f(s_t, a_t) + \mathcal{N}(\mu, \sigma)$. Given our primary focus on a stochastic environment where absolute constraint satisfaction is challenging, we opt for a small but positive ϵ bound. Each experiment is repeated with four random seeds, over which the mean ± standard deviation (std) results are reported. More details are shown in Appendix C.2.

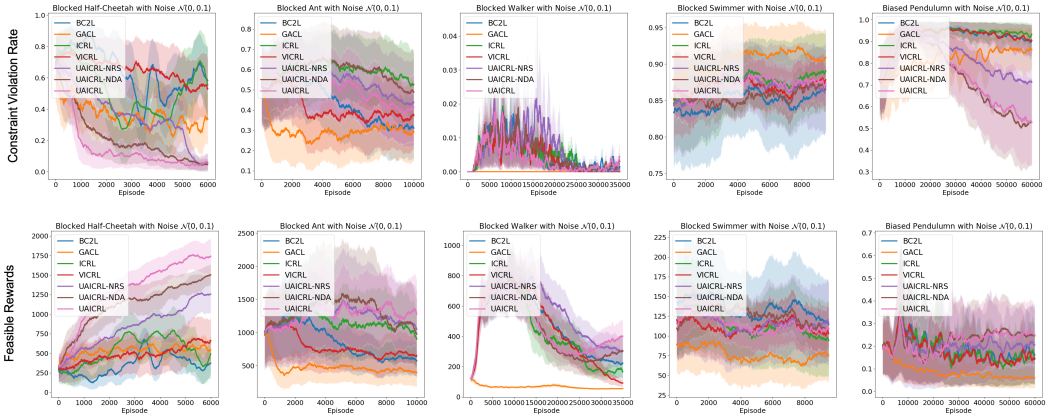

Figure 3: The constraint violation rate (top) and feasible rewards (bottom) in MuJoCo environments during training. From left to right, the environments are Blocked Half-cheetah, Blocked Ant, Blocked Walker, Blocked Swimmer, and Biased Pendulumn with stochasticity of $\mathcal{N}(0, 0.1)$.

Figure 3 displays the training curves in stochastic MuJoCo environments with noise σ = 0.1 (check Appendix D.2 for results with σ = 0.01 and 0.001). We observe that UAICRL-NRS and UAICRL-NDA can generally outperform the baselines in most environments, which underscores the effectiveness of employing data augmentation for modeling epistemic uncertainty and utilizing distributional estimator for capturing aleatoric uncertainty. When implementing both techniques, UAICRL demonstrates even more robust and superior performance compared to other methods. It is worth noting that,

although GACL exhibits the fewest constraint violations in the Blocked Ant environment, it struggles to pursue rewards and underperforms in other environments. Interestingly, we find that increased stochasticity does not necessarily result in poorer model performance. This is because small noise levels are difficult to detect, causing the models to be less sensitive to minor variations.

6.3 Realistic Environment: Stochastic Highway Driving

We conduct experiments on a high-dimensional realistic Highway Driving (HighD) environment (Krajewski et al., 2018; Liu et al., 2023), which defines a complex highway driving task with constraints and requires the agent to drive safely to the destination by observing human drivers' demonstrations. In this paper, we focus on the velocity constraint, which guarantees the ego car to drive at a safe velocity. Additionally, we add Gaussian noise to the agent's action a_t to introduce the control-level stochasticity. The results under noise $\mathcal{N}(0, 0.1)$ are shown in Figure 4 (check Appendix D.3 for complete results with $\mathcal{N}(0, 0.01)$ and $\mathcal{N}(0, 0.001)$). We find that although GACL violates the fewest constraints, it is too conservative to pursue rewards, which limits its practical application. Other baselines including BC2L, ICRL, and VICRL can obtain high rewards, but their constraint violation rates are still relatively high. Instead, UAICRL can achieve the highest rewards while maintaining a satisfactorily low constraint violation rate, indicating its great potential to handle uncertainties even in complex environments.

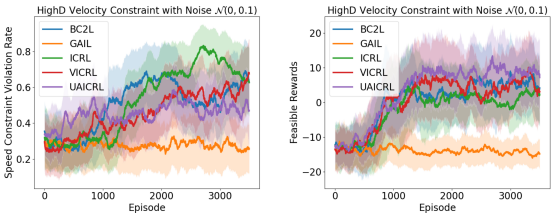

Figure 4: Model performance in the HighD environment with velocity constraint under $\mathcal{N}(0, 0.1)$.

6.4 In-depth Study on Uncertainty Management Performance

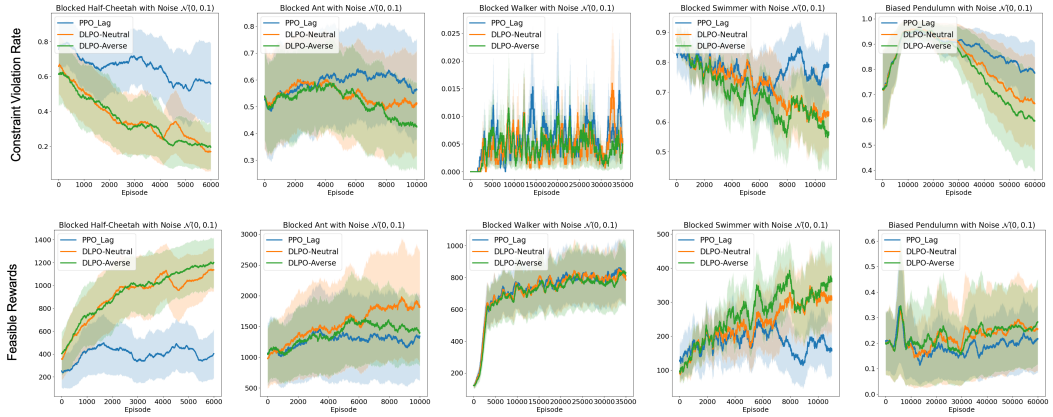

Figure 5: The constraint violation rate (top) and feasible rewards (bottom) in MuJoCo environments with stochasticity of $\mathcal{N}(0, 0.1)$ when given ground-truth constraints. Check Appendix D.2 for the complete results with $\mathcal{N}(0, 0.01)$ and $\mathcal{N}(0, 0.001)$.

In order to comprehend the critical factors contributing to uncertainty sensitivity, we study the model's capabilities of representing aleatoric and epistemic uncertainties from the following perspectives:

"Can DLPO lead to more robust control under stochastic dynamics?" To address this question, we provide each agent with the ground-truth constraints in the environment. We focus solely on whether DLPO can learn constraint-satisfying behavior while being influenced by aleatoric uncertainty. Figure 5 displays the results in MuJoCo environments. The compared methods include: 1) **DLPO-Neutral** and **DLPO-Averse**, which refer to the options of applying expectation and CVaR(0.25) as the risk measures (ρ) in Objective 4. 2) **Classic PPO-Lagrange**, following the implementation in (Liu et al., 2023). Our findings indicate that by implementing the risk-sensitive constraint, the policy effectively learns to respect the constraints while simultaneously pursuing rewards. Furthermore, in most cases, the DLPO-Averse method employing CVaR as the risk measure results in fewer constraint violations compared to the risk-neutral approach. However, CVaR sometimes leads to overly conservative behavior, hindering the agent from efficiently pursuing rewards.

Based on the same motivation, we train agents in Gridworld environments under known constraints to better understand the advantages of the distributional estimator. Figure 6 visualizes the trajectories generated by the PPO-Lagrange and DLPO algorithms in three distinct scenarios. We notice that DLPO consistently maintains an appropriate distance from the constrained locations, effectively mitigating the impact of noise. For example, at $s(4,4)$ of the first Gridworld environment, PPO-Lag chooses to move left, whereas DLPO decides to move upper-left, maintaining a larger gap to the constrained region. We use red circles to highlight the critical locations, where the agents display different actions. Figure 6 (last row) visualizes the predicted cost distribution at these locations. We find that these distributions are sensitive to the choices of actions, consequently assigning larger expectations and variances to actions with a higher risk of constraint violation (e.g., moving lower left in the red spot of the first setting).

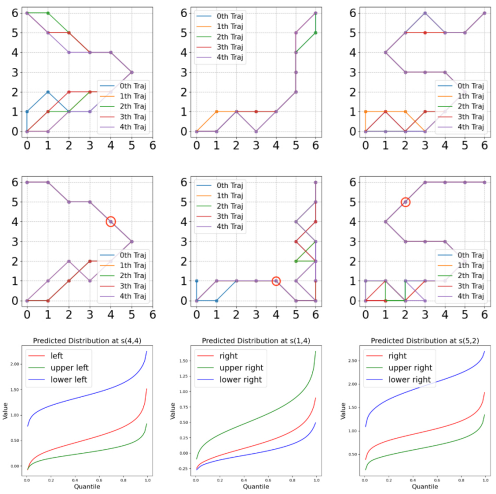

Figure 6: The trajectories generated by PPO-Lag (first row) and DLPO (second row). The bottom row illustrates the predicted constraint distributions at critical states denoted by red circles.

"Can FTG facilitate constraint inference in stochastic environments?" Despite the ablation studies in the main experiments (e.g., compare UAICRL and UAICRL-NDA), for a better understanding of the effectiveness of FTG, we visualize the inferred constraint from MEICRL (Malik et al., 2021) and our UAICRL with FTG. Figure 7 illustrates the recovered costs in Gridworld environments. It shows that although accurately inferring the exact ground-truth constraints (see Figure 2) is hard, the constraints recovered using FTG demonstrate a closer approximation compared to those without data augmentation. This finding suggests that when trained with additionally generated trajectories, the constraint model can predict step-wise costs more accurately, thereby reducing the epistemic uncertainty caused by OoD samples.

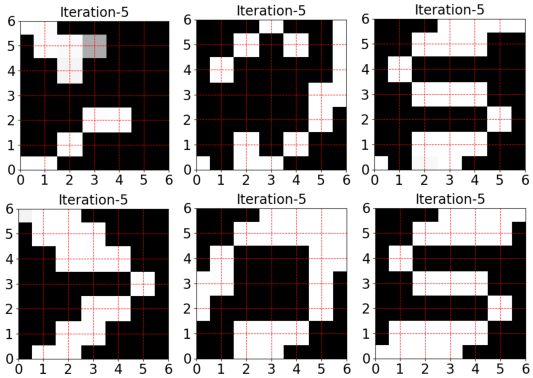

Figure 7: The constraint maps recovered by MEICRL and UAICRL in three stochastic Gridworlds.

6.5 Limitations

Theoretical Results. While our study indeed encompasses theoretical findings, it still lacks a rigorous theoretical analysis. Existing theoretical results on IRL (Metelli et al., 2021; Lindner et al., 2022; Metelli et al., 2023) depended on the construction of the feasible set of rewards, but defining the exact feasible set for the constraint is challenging due to its various types (e.g., hard, soft or probabilistic constraints) and optimization techniques (e.g., Lagrange methods). A rigorous theoretical study is beyond the scope of our current work. Nevertheless, we affirm the importance of such understanding and suggest extending the theoretical outcomes from IRL to ICRL as a future work.

Limited Risk Measure. For fairness and simplicity, our study predominantly employs the widely used CVaR method. Although the performance of other metrics like VaR and Entropic Value-at-Risk (EVaR) has not been studied, they could be easily integrated into our framework for future exploration.

7 Conclusion

This paper introduces UAICRL, a novel ICRL framework that considers both aleatoric and epistemic uncertainties towards uncertainty awareness in constraint inference. Empirical results demonstrate that UAICRL outperforms several other ICRL methods by leveraging its uncertainty-aware capability. A direction of future work is to extend our model to offline settings without access to the environment.

REFERENCES

Gaon An, Seungyong Moon, Jang-Hyun Kim, and Hyun Oh Song. Uncertainty-based offline reinforcement learning with diversified q-ensemble. In *Neural Information Processing Systems (Neurips)*, 2021.

Mattijs Baert, Pietro Mazzaglia, Sam Leroux, and Pieter Simoens. Maximum causal entropy inverse constrained reinforcement learning. *arXiv preprint arXiv:2305.02857*, 2023.

Marc G Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. In *International Conference on Machine Learning (ICML)*, pp. 449–458, 2017.

Yoshua Bengio, Salem Lahlou, Tristan Deleu, Edward J. Hu, Mo Tiwari, and Emmanuel Bengio. Gflownet foundations. In *Neural Information Processing Systems (Neurips)*, 2021.

Lu Chen, Xiang Zhou, Cheng Chang, Runzhe Yang, and Kai Yu. Agent-aware dropout dqn for safe and efficient on-line dialogue policy learning. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 2454–2464, 2017.

Jinyoung Choi, Christopher R. Dance, Jung-Eun Kim, Seulbin Hwang, and Kyungsik Park. Risk-conditioned distributional soft actor-critic for risk-sensitive navigation. In *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 8337–8344, 2021.

Glen Chou, Dmitry Berenson, and Necmiye Ozay. Uncertainty-aware constraint learning for adaptive safe motion planning from demonstrations. In *Conference on Robot Learning (CoRL)*, pp. 1612–1639, 2020a.

Glen Chou, Dmitry Berenson, and Necmiye Ozay. Learning constraints from demonstrations. In *Algorithmic Foundations of Robotics XIII: Proceedings of the 13th Workshop on the Algorithmic Foundations of Robotics 13*, pp. 228–245, 2020b.

Yinlam Chow, Aviv Tamar, Shie Mannor, and Marco Pavone. Risk-sensitive and robust decision-making: a cvar optimization approach. In *Neural Information Processing Systems (Neurips)*, 2015.

Felipe Leno Da Silva, Pablo Hernandez-Leal, Bilal Kartal, and Matthew E. Taylor. Uncertainty-aware action advising for deep reinforcement learning agents. In *AAAI Conference on Artificial Intelligence*, pp. 5792–5799, 2020.

Will Dabney, Mark Rowland, Marc G. Bellemare, and Rémi Munos. Distributional reinforcement learning with quantile regression. In *AAAI Conference on Artificial Intelligence*, pp. 2892–2901, 2018.

Marylou Gabrié, Andre Manoel, Clément Luneau, Nicolas Macris, Florent Krzakala, Lenka Zdeborová, et al. Entropy and mutual information in models of deep neural networks. In *Neural Information Processing Systems (Neurips)*, 2018.

Ashish Gaurav, Kasra Rezaee, Guiliang Liu, and Pascal Poupart. Learning soft constraints from constrained expert demonstrations. In *International Conference on Learning Representations (ICLR)*, 2023.

Julian Gerstenberg, Ralph Neininger, and Denis Spiegel. On solutions of the distributional bellman equation. *Electronic Research Archive*, 31(8):4459–4483, 2023.

Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. In *Neural Information Processing Systems (Neurips)*, pp. 4565–4573, 2016.

Carl-Johan Hoel, Krister Wolff, and Leo Laine. Ensemble quantile networks: Uncertainty-aware reinforcement learning with applications in autonomous driving. *IEEE Transactions on Intelligent Transportation Systems*, 2023.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

Robert Krajewski, Julian Bock, Laurent Kloeker, and Lutz Eckstein. The highd dataset: A drone dataset of naturalistic vehicle trajectories on german highways for validation of highly automated driving systems. In *International Conference on Intelligent Transportation Systems (ITSC)*, pp. 2118–2125, 2018.

Aviral Kumar, Justin Fu, Matthew Soh, George Tucker, and Sergey Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. In *Neural Information Processing Systems (Neurips)*, 2019.

Yinchuan Li, Shuang Luo, Haozhi Wang, and Jianye Hao. Cflownets: Continuous control with generative flow networks. In *International Conference on Learning Representations (ICLR)*, 2023.

David Lindner, Andreas Krause, and Giorgia Ramponi. Active exploration for inverse reinforcement learning. In *NeurIPS*, 2022.

Guiliang Liu, Yudong Luo, Oliver Schulte, and Pascal Poupart. Uncertainty-aware reinforcement learning for risk-sensitive player evaluation in sports game. In *Neural Information Processing Systems (Neurips)*, 2022.

Guiliang Liu, Yudong Luo, Ashish Gaurav, Kasra Rezaee, and Pascal Poupart. Benchmarking constraint inference in inverse reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2023.

Shicheng Liu and Minghui Zhu. Distributed inverse constrained reinforcement learning for multi-agent systems. In *Neural Information Processing Systems (Neurips)*, 2022.

Yongshuai Liu, Avishai Halev, and Xin Liu. Policy learning with constraints in model-free reinforcement learning: A survey. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 4508–4515, 2021.

Yudong Luo, Guiliang Liu, Haonan Duan, Oliver Schulte, and Pascal Poupart. Distributional reinforcement learning with monotonic splines. In *International Conference on Learning Representations (ICLR)*, 2022.

Björn Lütjens, Michael Everett, and Jonathan P How. Safe reinforcement learning with model uncertainty estimates. In *International Conference on Robotics and Automation (ICRA)*, pp. 8662–8668, 2019.

Shehryar Malik, Usman Anwar, Alireza Aghasi, and Ali Ahmed. Inverse constrained reinforcement learning. In *International Conference on Machine Learning (ICML)*, pp. 7390–7399, 2021.

Nikolay Malkin, Moksh Jain, Emmanuel Bengio, Chen Sun, and Yoshua Bengio. Trajectory balance: Improved credit assignment in gflownets. In *Neural Information Processing Systems (Neurips)*, 2022.

Borislav Mavrin, Hengshuai Yao, Linglong Kong, Kaiwen Wu, and Yaoliang Yu. Distributional reinforcement learning for efficient exploration. In *International Conference on Machine Learning (ICML)*, pp. 4424–4434, 2019.

David Livingston McPherson, Kaylene C. Stocking, and S. Shankar Sastry. Maximum likelihood constraint inference from stochastic demonstrations. In *IEEE Conference on Control Technology and Applications, (CCTA)*, pp. 1208–1213, 2021.

Alberto Maria Metelli, Giorgia Ramponi, Alessandro Concetti, and Marcello Restelli. Provably efficient learning of transferable rewards. In *International Conference on Machine Learning, ICML*, volume 139, pp. 7665–7676, 2021.

Alberto Maria Metelli, Filippo Lazzati, and Marcello Restelli. Towards theoretical understanding of inverse reinforcement learning. In *International Conference on Machine Learning, ICML*, volume 202, pp. 24555–24591, 2023.

Oliver Mihatsch and Ralph Neuneier. Risk-sensitive reinforcement learning. *Machine Learning*, 49 (2-3):267–290, 2002.

Xinyi Ni and Lifeng Lai. Evar optimization for risk-sensitive reinforcement learning. *IEEE Transactions on Information Theory*, 2022.

Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. Deep exploration via bootstrapped dqn. In *Neural Information Processing Systems (Neurips)*, 2016.

Dimitris Papadimitriou, Usman Anwar, and Daniel S Brown. Bayesian methods for constraint inference in reinforcement learning. *Transactions on Machine Learning Research*, 2023.

Daehyung Park, Michael Noseworthy, Rohan Paul, Subhro Roy, and Nicholas Roy. Inferring task goals and constraints using bayesian nonparametric inverse reinforcement learning. In *Conference on Robot Learning (CoRL)*, pp. 1005–1014, 2020.

LA Prashanth, Michael C Fu, et al. Risk-sensitive reinforcement learning via policy gradient search. *Foundations and Trends® in Machine Learning*, 15(5):537–693, 2022.

Harsh Satija, Philip Amortila, and Joelle Pineau. Constrained markov decision processes via backward value functions. In *International Conference on Machine Learning (ICML)*, pp. 8502–8511, 2020.

John Schulman, Philipp Moritz, Sergey Levine, Michael I. Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. In *International Conference on Learning Representations (ICLR)*, 2016.

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

Dexter R. R. Scobee and S. Shankar Sastry. Maximum likelihood constraint inference for inverse reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2020.

Lewis Smith and Yarin Gal. Understanding measures of uncertainty for adversarial example detection. In *The Conference on Uncertainty in Artificial Intelligence (UAI)*, pp. 560–569, 2018.

Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

Joost van Amersfoort, Lewis Smith, Yee Whye Teh, and Yarin Gal. Uncertainty estimation using a single deep deterministic neural network. In *International Conference on Machine Learning (ICML)*, pp. 9690–9700, 2020.

Jingda Wu, Zhiyu Huang, and Chen Lv. Uncertainty-aware model-based reinforcement learning: methodology and application in autonomous driving. *IEEE Transactions on Intelligent Vehicles*, 8 (1):194–203, 2023.

Qisong Yang, Thiago D Simão, Simon H Tindemans, and Matthijs TJ Spaan. Safety-constrained reinforcement learning with a distributional safety critic. *Machine Learning*, 112(3):859–887, 2023.

Brian D. Ziebart, Andrew L. Maas, J. Andrew Bagnell, and Anind K. Dey. Maximum entropy inverse reinforcement learning. In *AAAI Conference on Artificial Intelligence*, pp. 1433–1438, 2008.

Brian D. Ziebart, J. Andrew Bagnell, and Anind K. Dey. Modeling interaction via the principle of maximum causal entropy. In *International Conference on Machine Learning (ICML)*, pp. 1255–1262, 2010.

A PROOF

A.1 PROOF OF PROPOSITION 1

Let $\boldsymbol{Z}, \bar{\boldsymbol{C}}^\pi$ denote the vector-valued random variables of size $|\mathcal{S}|$, where $Z(s) = \sum_{t=0}^{\infty} \gamma^t C_t | S_0 = s$ and $\bar{C}^\pi(s) = \int_{a\in\mathcal{A}} \pi(a|s) C(s,a) \mathrm{d}a$. Let $\boldsymbol{P}^\pi$ denote the state transition probability matrix under policy π, where $P^\pi_{s,s'} = \int_{a\in\mathcal{A}} P\left(s'|s,a\right) \mathrm{d}\pi\left(a|s\right)$. Assuming the Bellman consistency holds by $\boldsymbol{Z} \overset{\Delta}{:=} \bar{\boldsymbol{C}}^\pi + \gamma \boldsymbol{P}^\pi \boldsymbol{Z}$, we show that the uncertainty of distributions $\boldsymbol{Z}$ under an entropy measure can be given by:

$$\mathcal{H}(\boldsymbol{Z}) = \mathcal{H}[\bar{\boldsymbol{C}}^\pi] - |\mathcal{S}| \log(1-\gamma) + \log\left|\det\left(\mathbf{d}^\pi\right)\right| \tag{11}$$

where $\mathbf{d}^\pi = (1-\gamma)\left(I - \gamma \boldsymbol{P}^\pi\right)^{-1} \in [0,1]^{|S|\times|S|}$ is the induced matrix for distributions over states by following policy π and transition $P_{\mathcal{T}}$.

Proof:

$$H(\boldsymbol{Z}) \overset{\text{(a)}}{=} H\left[\left(I - \gamma \boldsymbol{P}^\pi\right)^{-1} \bar{\boldsymbol{C}}^\pi\right] \tag{12}$$

$$\overset{\text{(b)}}{=} \log\left|\det\left[\left(I - \gamma \boldsymbol{P}^\pi\right)^{-1}\right]\right| + H[\bar{\boldsymbol{C}}^\pi] \tag{13}$$

$$\overset{\text{(c)}}{=} \log\left|\det\left[\frac{\mathbf{d}^\pi}{1-\gamma}\right]\right| + H[\bar{\boldsymbol{C}}^\pi] \tag{14}$$

$$\overset{\text{(d)}}{=} -|S|\log(1-\gamma) + \log\left|\det\left[\mathbf{d}^\pi\right]\right| + H[\bar{\boldsymbol{C}}^\pi] \tag{15}$$

(a) holds by following the Bellman consistency $\boldsymbol{Z} \overset{\Delta}{:=} \bar{\boldsymbol{C}}^\pi + \gamma \boldsymbol{P}^\pi \boldsymbol{Z}$. To show the invertibility of $(I - \gamma \boldsymbol{P}^\pi)$, it is sufficient to demonstrate that the matrix is full rank. Specifically, for any non-zero vector $\boldsymbol{x} \in \mathbb{R}^{|S|}$:

$$\begin{aligned}
\left\|\left(I - \gamma \boldsymbol{P}^\pi\right)\boldsymbol{x}\right\|_\infty &= \left\|\boldsymbol{x} - \gamma \boldsymbol{P}^\pi \boldsymbol{x}\right\|_\infty \\
&\geq \|\boldsymbol{x}\|_\infty - \gamma \left\|\boldsymbol{P}^\pi \boldsymbol{x}\right\|_\infty \\
&\geq \|\boldsymbol{x}\|_\infty - \gamma \|\boldsymbol{x}\|_\infty \\
&= (1-\gamma)\|\boldsymbol{x}\|_\infty \\
&\geq 0
\end{aligned}$$

(b) holds by applying the properties of differential entropy and utilizing the invertibility of $(I - \gamma \boldsymbol{P}^\pi)$.

(c) holds by defining $\mathbf{d}^\pi = (1-\gamma)\left(I - \gamma \boldsymbol{P}^\pi\right)^{-1}$. We aim to demonstrate that $\mathbf{d}^\pi \in [0,1]^{|S|\times|S|}$ represents the induced matrix for distributions over states when following policy π. Specifically, the $(s)^{th}$ row of $\mathbf{d}^\pi$ corresponds to the distribution over states induced by following policy π after starting with $s_0 = s$. This follows directly from the definition of $\mathbf{d}^\pi$.

$$\begin{aligned}
\mathbf{d}^\pi &= (1-\gamma)\sum_{t=1}^{\infty}\left(\gamma \boldsymbol{P}^\pi\right)^t \\
&= \frac{(1-\gamma)\left[1 - \left(\gamma \boldsymbol{P}^\pi\right)^\infty\right]}{1 - \left(\gamma \boldsymbol{P}^\pi\right)} \\
&= \frac{(1-\gamma)}{1 - \left(\gamma \boldsymbol{P}^\pi\right)}
\end{aligned}$$

Proposition 1 provides a decomposition of the entropy of $\boldsymbol{Z}$ into three components. 1) The first component is the entropy of the cost variables, which quantifies the uncertainty associated with the current costs. 2) The second component captures the uncertainty induced by the discount factor, which determines the degree to which the current uncertainty estimation should be influenced by the stochasticity of future rewards or transitions. 3) The third component is a log-absolute determinant of the induced distribution matrix, which measures the extent to which the transition function $P_{\mathcal{T}}$ and the policy π stretch or alter the initial state-action distribution. These components correspond to the key elements of aleatoric uncertainty.

A.2 PROOF OF PROPOSITION 2

Proof:

The mutual information term can be factorized as follows:

$$I(\omega;\Phi|\bar{\tau},\mathcal{D}) \overset{(a)}{=} \mathcal{H}[p(\Phi|\bar{\tau},\mathcal{D})] - \mathbb{E}_{p(\omega|\mathcal{D})}\Big[\mathcal{H}[p(\Phi|\bar{\tau};\omega)]\Big] \tag{16}$$

$$\overset{(b)}{=} \mathcal{H}[p(\Phi|\bar{\tau},\mathcal{D})] - \int p(\omega|\mathcal{D})\mathcal{H}[p(\Phi|\bar{\tau};\omega)]\mathrm{d}\omega \tag{17}$$

$$\overset{(c)}{\simeq} \mathcal{H}[p(\Phi|\bar{\tau},\mathcal{D})] - \int q(\omega)\mathcal{H}[p(\Phi|\bar{\tau};\omega)]\mathrm{d}\omega \tag{18}$$

$$\overset{(d)}{\simeq} \mathcal{H}[p(\Phi|\bar{\tau},\mathcal{D})] - \frac{1}{M}\sum_{m}\mathcal{H}[p(\Phi|\bar{\tau};\omega_m)] \text{ where } \omega_m \sim q(\omega) \tag{19}$$

(a) holds by following the meaning of mutual information (i.e., the amount of information gained by the model ω if receiving a label Φ for a new trajectory $\bar{\tau}$, given the dataset $\mathcal{D}$).

(c) holds by using the variational inference to approximate the intractable posterior $p(\omega|\mathcal{D})$ with a simpler approximating distribution $q(\omega)$. For neural networks, the dropout distribution is commonly used.

(d) holds by sampling from the approximating distribution (e.g., the dropout distribution) with Monte Carlo method.

B IMPLEMENTATION DETAILS

B.1 MORE ALGORITHMS

We show the Distributional Lagrange Policy Optimization (DLPO) and Flow-based Trajectory Generation (FTG) in Algorithm 2 and Algorithm 3, respectively.

Algorithm 2: Distributional Lagrange Policy Optimization (DLPO)

Input: Lagrange multiplier λ, risk measure ρ, GAE lambda λ_g, rollout rounds B, update rounds $\mathcal{K}$, policy π_ψ, reward value critic V^r and distributional cost value critic Z^c;
Initialize state s_0 from CMDP and the roll-out dataset $\mathcal{D}_{roll}$;
for $b = 1, 2, \ldots, B$ **do**
  Perform policy π_ψ and collect trajectories $\tau_b = [s_0, a_0, r_0, c_0, \ldots, s_T, a_T, r_T, c_T]$;
  Calculate reward advantages A_t^r and return R_t^r via GAE (Schulman et al., 2016) ;
  Calculate cost advantages $A_t^c = \sum_{\iota=t}^{T}(\gamma\lambda_g)^\iota\left[c_\iota + \gamma\rho(Z^c(s_{\iota+1})) - \rho(Z^c(s_\iota))\right]$;
  Add samples to the dataset $\mathcal{D}_{roll} = \mathcal{D}_{roll} \cup \{s_t, a_t, r_t, A_t^r, R_t^r, c_t, A_t^c\}$;
end
for $\kappa = 1, 2, \ldots, \mathcal{K}$ **do**
  Sample a data point $s_\kappa, a_\kappa, r_\kappa, A_\kappa^r, R_\kappa^r, c_\kappa, A_\kappa^c$ from the dataset $\mathcal{D}_{roll}$;
  Calculate the clipping loss $L^{CLIP} =$
  $\min\left[\frac{\pi_\psi(a_\kappa|s_\kappa)}{\pi_{\psi,old}(a_\kappa|s_\kappa)}(A_\kappa^r - \lambda(A_\kappa^c - \epsilon)), clip(\frac{\pi_\psi(a_\kappa|s_\kappa)}{\pi_{\psi,old}(a_\kappa|s_\kappa)}, 1-\delta, 1+\delta)(A_\kappa^r - \lambda(A_\kappa^c - \epsilon))\right]$;
  Update policy parameters ψ by minimizing the loss: $-L^{CLIP} - \beta\mathcal{H}[\pi_\psi(a_\kappa|s_\kappa)]$;
  Update the reward critic V^r by minimizing the loss: $L^{VF} = \|V^r(s_\kappa) - R_\kappa\|_2^2$;
  Update the cost distribution Z^c by distributional Bellman operator (Equation 5) ;
end
Update the Lagrange multiplier λ by minimizing the loss $L^\lambda = \lambda[\mathbb{E}_{\mathcal{D}_{roll}}(c) - \epsilon]$;

B.2 DISCRETIZED IMPLEMENTATION

We follow the Policy Iteration Lagrange algorithm (see Algorithm 2 in (Liu et al., 2023)) to solve discretized control problems. Based on it, we simply incorporate a distributional term into the value

Algorithm 3: Flow-based Trajectory Generation (FTG)

Input: Flow neural network F_ξ, retrieval neural network G_χ, trajectory dataset $\mathcal{D} = \{\tau_i\}_{i=1}^N$, empty probability buffer $\mathcal{P}$, empty generated trajectory buffer $\mathcal{D}^G$, generate rounds I;

// Flow matching

while *not converge* **do**

- Sample a minibatch $\mathcal{B}$ of trajectory data from $\mathcal{D}$;
- Update retrieval network G_χ by minimizing the loss: $MSELoss(s_t, G_\chi(s_{t+1}, a_t))$;
- Uniformly sample M actions $\{a_m\}_{m=1}^M$ from action space $\mathcal{A}$ for each state s_t in $\mathcal{B}$;
- Calculate inflows $F_{in} = \log\left[\epsilon + \sum_{m=1}^M \exp F_\xi\left(G_\chi\left(s_t, a_m\right), a_m\right)\right]$;
- Calculate outflows $F_{out} = \log\left[\epsilon + \lambda R\left(s_t\right) + \sum_{m=1}^M \exp F_\xi\left(s_t, a_m\right)\right]$;
- Update flow network F_ξ by flow matching (Eqn. 8);

end

// Trajectory generation

for $\kappa = 1, 2, \ldots, I$ **do**

- Initialize state s_0 from CMDP, set $t = 0$;
- Append s_0 to the generated trajectory τ_κ^G;
- **while** $s \neq terminal$ **do**
  - Uniformly sample K actions $\{a_i\}_{i=1}^K$ from action space $\mathcal{A}$;
  - Generate the action probability buffer $\mathcal{P}$ by calculating edge flow for each action: $\mathcal{P} = \{F_\xi(s_t, a_i)\}_{i=1}^K$;
  - Sample $a_t \sim \mathcal{P}$ and execute a_t to move to the next state s_{t+1};
  - Append s_{t+1} to the generated trajectory τ_κ^G;
- **end**
- Append generated trajectory τ_κ^G to buffer $\mathcal{D}^G$;

end

matrix and modify the flow-based data augmentation into a discretized version following (Malkin et al., 2022).

B.3 Experimental Settings

For training the ICRL models, we utilized a total of 8 NVIDIA GeForce RTX 3090 GPUs, each equipped with 24 GB of memory. The training process was conducted on a single running node, utilizing 8 CPUs per task. The random seeds in the MuJoCo and HighD environment are 123, 321, 456, and 654. With the allocated resources described above, running one seed in the virtual and realistic environments typically required a duration of 2-4 and 3-5 hours, respectively. To optimize all of our networks, we employed the Adam optimization algorithm (Kingma & Ba, 2015). The learning rate was updated using an exponential decay schedule parameterized by a decay factor in every iteration. We summarize the main hyperparameters in Teble B.1.

C Environmental Details

C.1 Gridworld

In this paper, we establish a map with dimensions of 7 × 7 units and construct three distinct settings, as illustrated in Figure 2. At each time, the agent is permitted to navigate to any of the adjacent eight grids by moving one step. Starting from the initial location, a reward of 1 is granted if the agent successfully traverses to the target location while avoiding the imposed constraints, and a reward of 0 is assigned for any other situations. Moreover, we modify the environment to simulate a certain degree of stochasticity. Specifically, there exists a random rate ($p_s = 0.01$ and 0.001) with which the environment receives a random action instead of the intended action executed by the agent.

Table B.1: List of the utilized hyperparameters in UAICRL. To ensure equitable comparisons, we maintain consistency in the parameters of the same neural networks across different models.

Parameters	HalfCheetah	Ant	Walker	Swimmer	Pendulum
General					
Expert Rollouts	10	50	50	50	50
Max Length	1000	500	500	500	100
Gamma	0.99	0.99	0.99	0.99	0.99
PPO					
Steps	2048	2048	2048	2048	2048
Reward-GAE-λ	0.95	0.9	0.9	0.9	0.9
Cost-GAE-λ	0.95	0.9	0.9	0.95	0.9
Policy Network π_θ	64, 64	64, 64	64, 64	64, 64	64, 64
Reward Network V_θ^r	64, 64	64, 64	64, 64	64, 64	64, 64
Cost Network Z_θ^c	256, 256	256, 256	256, 256	256, 256	256, 256
Learning Rate θ	3e-4	3e-5	1e-4	3e-4	1e-4
Target KL	0.01	0.02	0.02	0.01	0.02
Lagrangian					
Initial Value	1	0.05	0.1	1	0.1
Learning Rate	0.1	0.02	0.05	0.01	0.05
Constraint Function					
Network C_ω	20	40, 40	64, 64	20	20
Learning Rate ω	0.05	0.005	0.001	0.001	0.001
Backward Iterations	10	5	5	5	5
DLPO					
Quantiles	64	64	64	64	64
Risk Measure	CVaR	CVaR	CVaR	CVaR	CVaR
Risk Level	0.25	0.95	0.95	0.95	0.25
FTG					
Network F_ξ	256, 256	256, 256	256, 256	256, 256	256, 256
Learning Rate ξ	3e-4	3e-4	3e-4	3e-4	3e-4
Sample Action Size	10000	10000	10000	10000	10000
Sample Flows	50	50	50	50	50

C.2 MuJoCo

The five virtual robotics environments are built upon MuJoCo (see Figure C.1).

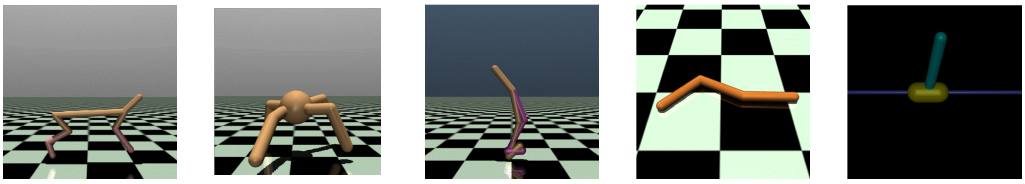

Figure C.1: MuJoCo environments. From left to right, the environments are Half-cheetah, Ant, Walker, Swimmer and Inverted Pendulum, respectively.

We modify them by incorporating predefined constraints and adjusting reward terms for ICRL. To simulate the stochasticity in the environment dynamics, we incorporate a Gaussian noise $\eta \sim \mathcal{N}(\mu, \sigma)$ into the transition function at each step such that $p\left(s_{t+1} \mid s_t, a_t\right) = f\left(s_t, a_t\right) + \eta$. In this work, we utilize $\mu = 0$ with $\sigma = 0.1, 0.01$ and 0.001 to represent different scales of noise. We provide a more comprehensive description as follows:

1) *Blocked Half-Cheetah.* In this environment, the agent controls a robot with two legs. The reward obtained by the agent is determined by the distance the robot walks between two time steps, and a penalty based on the magnitude of the input action. The game continues until a maximum time step of 1000 is reached. Due to the fact that the robot is easier to move backward than forward, we define a blocked region where the X-coordinate < -3. So the robot is restricted to move within the region where the X-coordinate ≥ -3.

2) *Blocked Ant.* In this environment, the agent controls a robot with four legs. The rewards obtained by the agent depend on the distance of the robot from the origin, and a healthy bonus for maintaining balance. The game terminates when a maximum time step of 500 is reached. Similar to the Blocked

Table C.1: The stochastic MuJoCo environments with constraints

Name	Obs. Dim.	Act. Dim.	Constraints	Stochasticity
Blocked Half-Cheetah	18	6	X-Coordinate $\geq$ -3	$\mathcal{N}(\mu, \sigma)$
Blocked Ant	113	8	X-Coordinate $\geq$ -3	$\mathcal{N}(\mu, \sigma)$
Blocked Walker	18	6	X-Coordinate $\geq$ -3	$\mathcal{N}(\mu, \sigma)$
Blocked Swimmer	10	2	X-Coordinate $\leq$ 0.5	$\mathcal{N}(\mu, \sigma)$
Biased Pendulum	4	1	X-Coordinate $\geq$ -0.015	$\mathcal{N}(\mu, \sigma)$

Half-Cheetah environment, we establish a constraint that blocks the region with X-coordinate < -3. As a result, the robot is only permitted to move within the region where the X-coordinate ≥ -3.

3) *Blocked Walker.* In this environment, the agent controls a two-legged robot and learns how to make it walk. The termination conditions for the game are either when the robot loses its balance or when it reaches the maximum time step of 500. The reward obtained by the agent is determined by the distance the robot walks between two time steps, along with a penalty based on the magnitude of the input action. Similar to the above environments, we impose a constraint that blocks the region where the X-coordinate < -3. Consequently, the robot is only allowed to move within the region where the X-coordinate ≥ -3.

4) *Blocked Swimmer.* In this environment, the agent controls a robot with two rotors connecting three segments, and learns how to move. The reward obtained by the agent is determined by the distance the robot walks between the current and previous time step, along with a penalty based on the magnitude of the input action. The game ends when the robot reaches the maximum time step of 500. In contrast to the aforementioned environments, the Swimmer robot is easier to move forward than to move backward. Therefore, we constrain the region where the X-coordinate > 0.5. Consequently, the robot is only allowed to move within the region where the X-coordinate ≤ 0.5.

5) *Biased Pendulum.* In this environment, the agent controls a pole to balance it on a cart. The game terminates either when the pole falls or when the maximum time step of 100 is reached. To increase difficulty, we assign higher rewards to the left locations, where the constraints are also imposed. Specifically, we introduce a constraint that blocks the region with X-coordinate < -0.015. At each step, a reward of 0.1 is provided if the X-coordinate ≥ 0, and a reward of 1 is given if the X-coordinate ≤ -0.01. The reward gradually decreases from 1 to 0.1 when the X-coordinate falls within $(-0.01, 0)$. Consequently, the agent is challenged to resist the temptation of higher rewards and stay within safe regions.

The MuJoCo environment settings in this work are summarized in Table C.1

C.3 Highway Driving

Figure C.2 illustrates the Highway Driving environment. In this scenario, the ego car is displayed in blue while other cars are shown in red. The ego car has limited visibility and can only observe objects within its vicinity (marked in blue). The objective is to navigate the car to reach the destination point without going off-road, colliding with other cars, or violating time limits and other constraints (e.g., velocity).

In this paper, we mainly focus on the velocity constraint, where we limit the speed of the ego car to no more than 40 m/s, to ensure the ego car can drive at a safe speed. Additionally, we add Gaussian noise to the agent's action at each time step to introduce the control-level stochasticity.

Figure C.2: The Highway Driving (HighD) environment.

D MORE EXPERIMENTAL RESULTS

D.1 MORE RESULTS IN GRIDWORLD

Figure D.1 and Figure D.2 show the training curves in Gridworlds with the random rate $p_s = 0.01$ and 0.001, respectively.

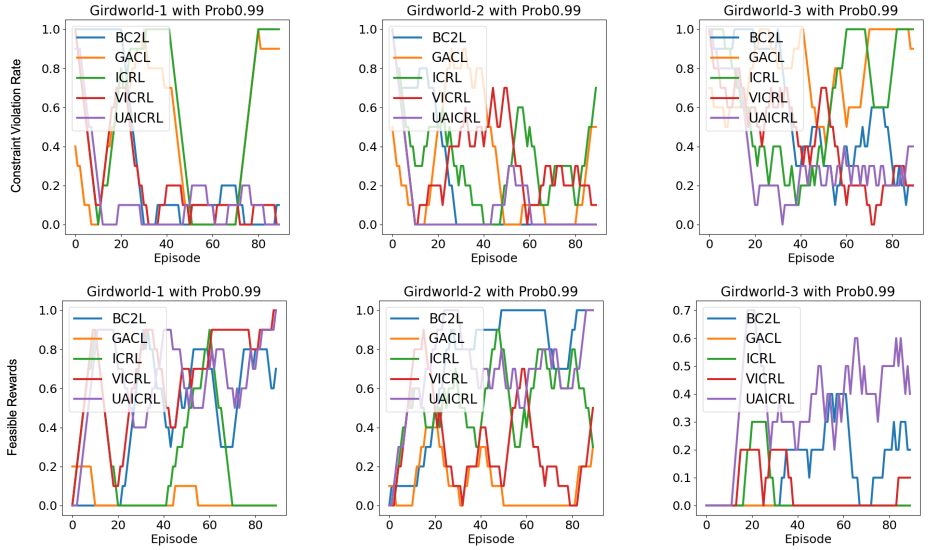

Figure D.1: The constraint violation rate (top) and feasible rewards (bottom) in three Gridworld settings with the random rate $p_s = 0.01$.

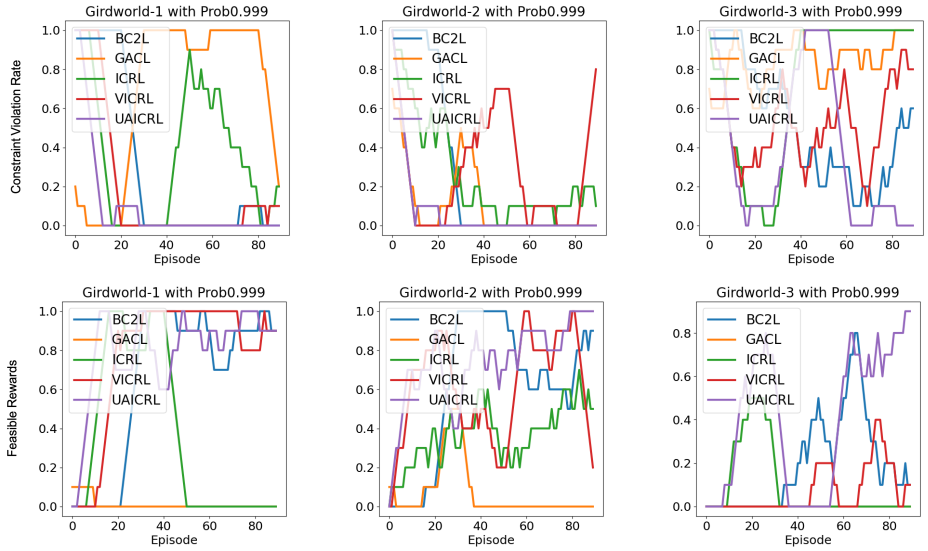

Figure D.2: The constraint violation rate (top) and feasible rewards (bottom) in three Gridworld settings with the random rate $p_s = 0.001$.

D.2 MORE RESULTS IN MUJOCO

Table D.1 shows the evaluation performance in MuJoCo environments with the stochasticity of $\mathcal{N}(0, 0.1)$.

Table D.1: Evaluation of different methods in the virtual environments with the stochasticity of $\mathcal{N}(0, 0.1)$. We report the mean ± std results averaged over 4 random seeds.

		Blocked Half-Cheetah	Blocked Ant	Blocked Walker	Blocked Swimmer	Biased Pendulum
Feasible Rewards	BC2L	333.4±175.2	595.0±225.2	222.1±94.9	91.9±46.1	0.1617±0.1295
	GACL	577.2±135.4	387.5±205.1	55.5±5.5	64.7±30.6	0.0572±0.0439
	ICRL	343.0±224.2	965.0±432.4	166.9±49.0	91.0±42.8	0.1440±0.1184
	VICRL	615.4±299.9	641.8±297.4	91.5±25.3	105.5±29.1	0.1447±0.1139
	UAICRL	**1714.4±222.0**	**1313.5±533.2**	**399.8±106.3**	**177.3±30.7**	**0.2551±0.1347**
Constraint Violation Rate	BC2L	68.8%±19.6%	31.4%±16.3%	0.0%±0.0%	89.4%±5.3%	90.6%±7.3%
	GACL	29.1%±16.2%	**28.8%±14.4%**	0.0%±0.0%	91.5%±3.7%	86.8%±9.8%
	ICRL	67.2%±21.2%	52.4%±17.5%	0.0%±0.0%	89.5%±4.9%	93.4%±5.2%
	VICRL	56.7%±20.6%	37.0%±17.3%	0.0%±0.0%	87.6%±3.3%	90.8%±7.0%
	UAICRL	**6.6%±5.5%**	41.7%±16.7%	**0.0%±0.0%**	**86.3%±3.3%**	**52.0%±20.3%**

Figure D.3 and Figure D.4 show the additional experimental results in MuJoCo environments with constraint recovery under the stochasticity of $\mathcal{N}(0, 0.01)$ and $\mathcal{N}(0, 0.001)$, respectively.

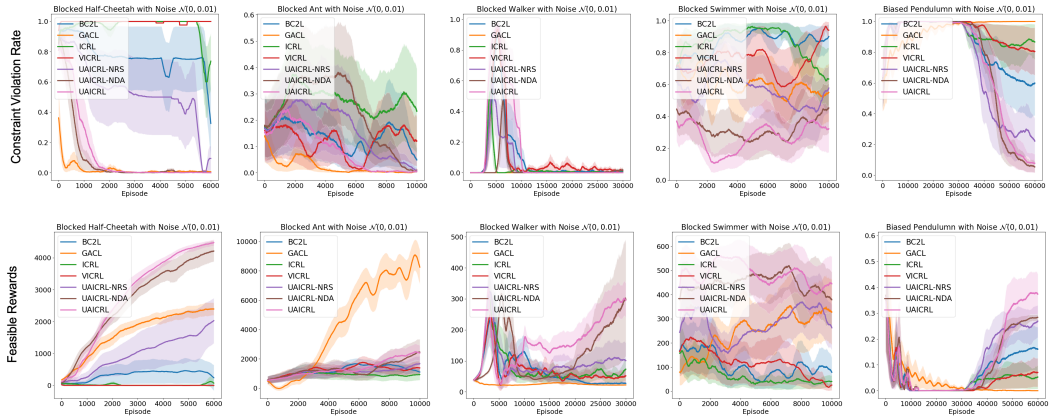

Figure D.3: The constraint violation rate (top) and feasible rewards (bottom) in MuJoCo environments with constraint recovery under the stochasticity of $\mathcal{N}(0, 0.01)$.

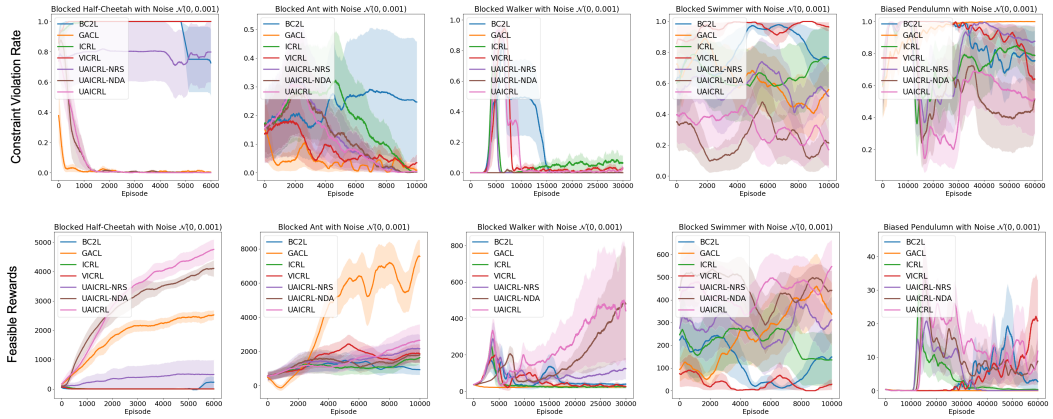

Figure D.4: The constraint violation rate (top) and feasible rewards (bottom) in MuJoCo environments with constraint recovery under the stochasticity of $\mathcal{N}(0, 0.001)$.

Figure D.5 and Figure D.6 show the additional experimental results in MuJoCo environments when given ground-truth constraints under the stochasticity of $\mathcal{N}(0, 0.01)$ and $\mathcal{N}(0, 0.001)$, respectively.

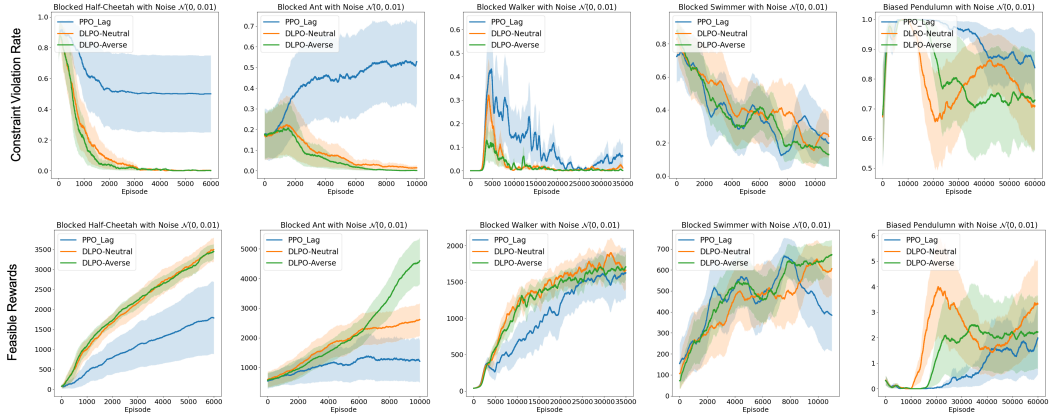

Figure D.5: The constraint violation rate (top) and feasible rewards (bottom) in MuJoCo environments with the stochasticity of $\mathcal{N}(0, 0.01)$ when given ground-truth constraints.

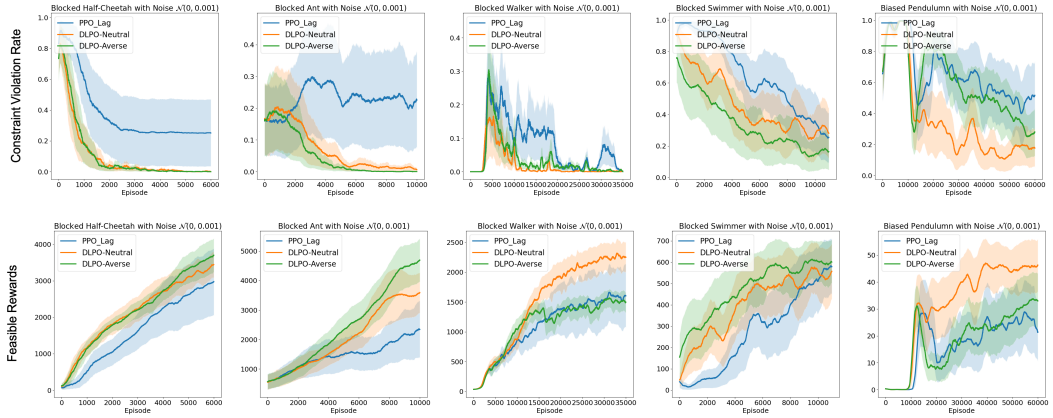

Figure D.6: The constraint violation rate (top) and feasible rewards (bottom) in MuJoCo environments with the stochasticity of $\mathcal{N}(0, 0.001)$ when given ground-truth constraints.

D.3 MORE RESULTS IN HIGHD

Figure D.7 show the additional experimental results in the HighD environment with constraint recovery under the stochasticity of $\mathcal{N}(0, 0.01)$ and $\mathcal{N}(0, 0.001)$, respectively.

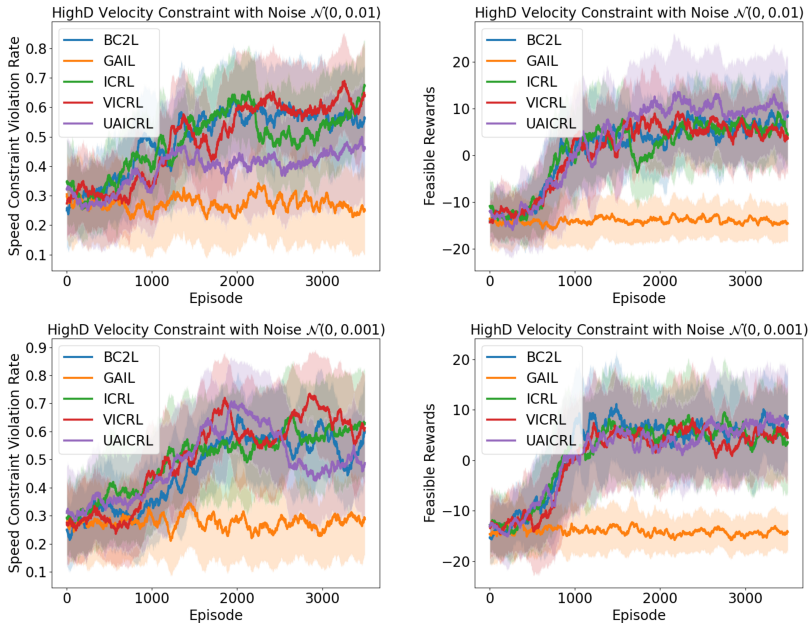

Figure D.7: Model performance in the HighD environment with velocity constraint under $\mathcal{N}(0, 0.01)$ (top) and $\mathcal{N}(0, 0.001)$ (bottom).