Constrained Decision Transformer for Offline Safe Reinforcement Learning

Zuxin Liu*[1], Zijian Guo*[1], Yihang Yao[1], Zhepeng Cen[1], Wenhao Yu[2], Tingnan Zhang[2], Ding Zhao[1]

[1] Carnegie Mellon University
[2] Google DeepMind

November 7, 2025

Motivation

Offline safe RL aims to learn a reward-maximizing policy **within a constrained manifold** from collected data **without further interaction** with the environment.

Why safe RL?

- Many real-world tasks can hardly be formulated as a scalar reward function.
- Simply maximizing the reward may cause constraint violation and catastrophic consequences in safety-critical applications.

Why offline safe RL?

- Online interaction brings challenges in ensuring training safety.
- Correction systems or human intervention as a safety guard are expensive due to the low sample efficiency.

Offline safe RL shows advantages to satisfy the safety requirements in both training and deployment in real-world applications.

Existing Methods Limitation

Most existing offline safe RL methods estimate the cost return and reward return to solve a constrained optimization problem.

- **Unreliable Safety:** Biased cost estimation from offline data leads to policies that are either unsafe or overly conservative.
- **Poor Adaptability:** Policies are constrained by a fixed threshold, requiring full re-training to adapt to a new constraint condition.

Proposed Method: CDT

Decision Transformer (DT) learns from offline datasets to predict target-return-conditioned actions, providing potential to adapt to different constraint thresholds.

Constrained Decision Transformer (CDT) leverages the return-conditioned sequential modeling framework to achieve **zero-shot adaptation** to different constraint thresholds at deployment while maintaining safety and high reward.

Setup

A safe RL problem is specified by a Constrained Markov Decision Process (CMDP) and a constraint threshold κ.

- MDP is defined by state space $\mathcal{S}$, action space $\mathcal{A}$, transition function $\mathcal{P}$, reward function r and initial state distribution μ_0.
- CMDP augments MDP with cost c for violating the constraint.
- Given policy π and trajectory $\tau = \{s_1, a_1, r_1, c_1 ..., s_T, a_T, r_T, c_t\}$
- Reward return $R(\tau) = \sum_{t=0}^{T-1} r_t$; cost return $C(\tau) = \sum_{t=0}^{T-1} c_t$.

Objective

Safe RL aims to find the policy that maximizes the reward return while limiting the cost incurred from constraint violations to the threshold κ:

$$\max_{\pi} \mathbb{E}_{\tau \sim \pi}[R(\tau)], \quad s.t. \quad \mathbb{E}_{\tau \sim \pi}[C(\tau)] \leq \kappa$$

Decision Transformer

- Processes a trajectory τ as $\{g_1, s_1, a_1, \ldots, g_{|\tau|}, s_{|\tau|}, a_{|\tau|}\}$.
- Learns a **deterministic** policy $\pi_{\mathrm{DT}}(a_t | \mathbf{s}_{-K,t}, \mathbf{g}_{-K,t})$.
- Trained to predict the action tokens under the standard **MSE loss**:

$$\mathbb{E}_{(\mathbf{a},\mathbf{s},\mathbf{g})\sim\tau}\left[\frac{1}{K}\sum_{k=1}^{K}\left(a_k - \pi_{\mathrm{DT}}(\mathbf{s}_{-K,k}, \mathbf{g}_{-K,k})\right)^2\right].$$

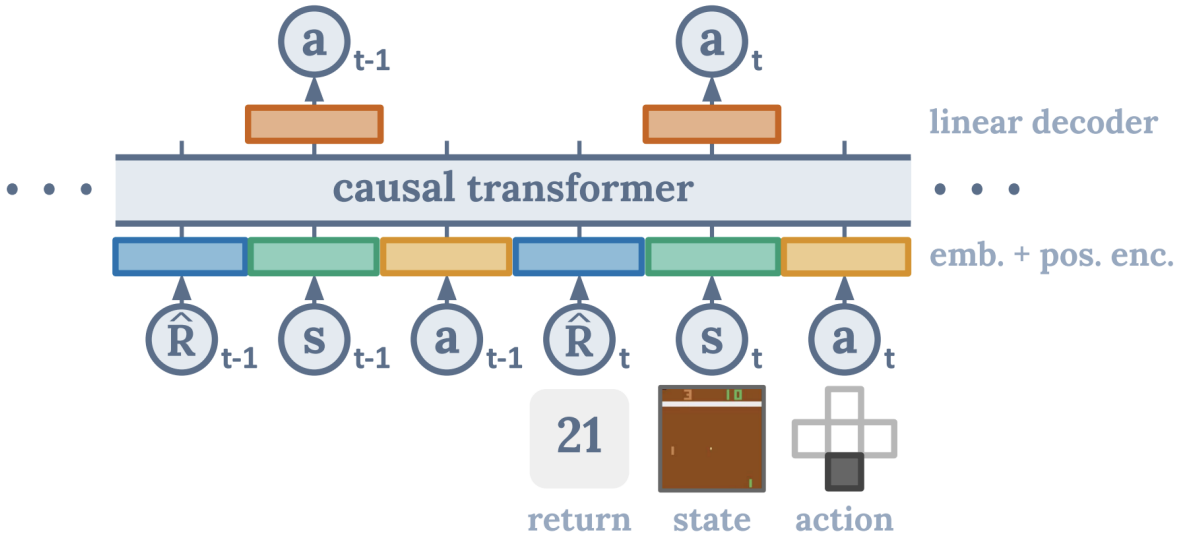


Figure: Decision Transformer architecture.

Online Decision Transformer

Online Decision Transformer (ODT) blends offline pretraining with online finetuning in a unified framework.

- Learn a **stochastic** policy $\pi_\theta(\mathbf{a}_t|\mathbf{s}_{-K,t}, \mathbf{g}_{-K,t}) = \mathcal{N}\left(\mu_\theta(\mathbf{s}_{-K,t}, \mathbf{g}_{-K,t}), \Sigma_\theta(\mathbf{s}_{-K,t}, \mathbf{g}_{-K,t})\right), \forall t.$
- Trained to predict the action distributions under **negative log-likelihood loss**:

$$J(\theta) = \frac{1}{K}\mathbb{E}_{(\mathbf{a},\mathbf{s},\mathbf{g})\sim\tau}\left[-\sum_{k=1}^{K}\log\pi_\theta(a_k|\mathbf{s}_{-K,k}, \mathbf{g}_{-K,k})\right],$$

- Impose policy entropy to encourage exploration

$$H_\theta^{\mathcal{T}}[\mathbf{a}|\mathbf{s}, \mathbf{g}] = \frac{1}{K}\mathbb{E}_{(\mathbf{s},\mathbf{g})\sim\tau}\left[\sum_{k=1}^{K} H[\pi_\theta(a_k|\mathbf{s}_{-K,k}, \mathbf{g}_{-K,k})]\right].$$

- Solving the ODT as a constrained problem

$$\min_\theta J(\theta) \text{ subject to } H_\theta^{\mathcal{T}}[\mathbf{a}|\mathbf{s}, \mathbf{g}] \geq \beta.$$

Constrained Decision Transformer

Constrained Decision Transformer (CDT) is built upon DT with two main difference components:

1. Adding a target cost return token the input sequence.
2. Stochastic policy with entropy regularization.
3. Pareto frontier-oriented data augmentation by target return relabeling.

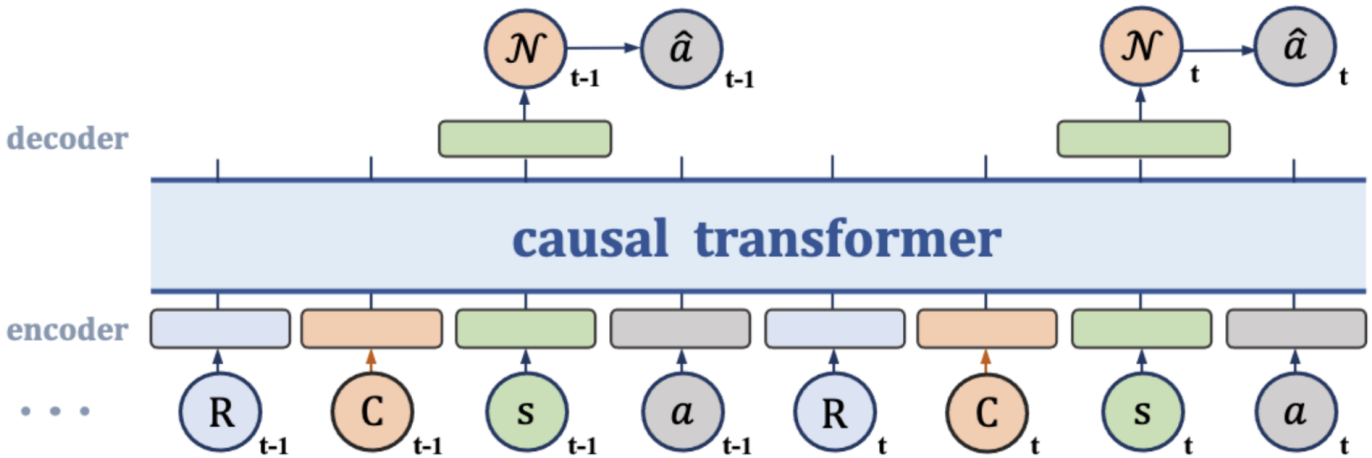
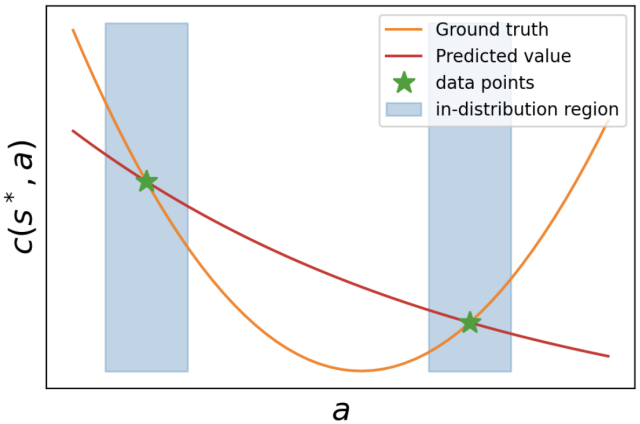
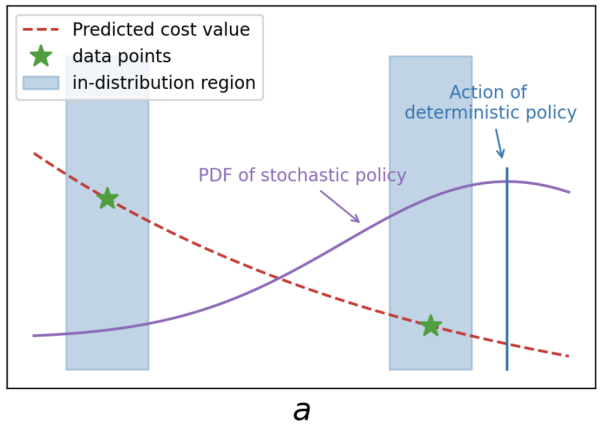


Figure: Constrained Decision Transformer architecture.

Stochastic CDT policy with entropy regularization

- Input tokens: $\mathbf{o}_t := \{\mathbf{R}_{-K:t}, \mathbf{C}_{-K:t}, \mathbf{s}_{-K:t}, \mathbf{a}_{-K:t-1}\}$,
- CDT policy: $\pi_\theta(\cdot|\mathbf{o}_t) = \mathcal{N}\left(\mu_\theta(\mathbf{o}_t), \Sigma_\theta(\mathbf{o}_t)\right)$.
- Optimization objective: $\min_\theta \mathbb{E}_{\mathbf{o}\sim\mathcal{T}}\left[-\log \pi_\theta(\mathbf{a}|\mathbf{o}) - \lambda H[\pi_\theta(\cdot|\mathbf{o})]\right]$





Why stochastic?

- Less prone to producing out-of-distribution (OOD) actions.
- Better exploration.
- Easily apply a regularizer to prevent overfitting.

PF, IPF and RF

- Pareto Frontier (PF) with $\kappa \in [0, \infty)]$:

$$\text{PF}(\kappa, \mathcal{T}) = \max_{\tau \in \mathcal{T}} R(\tau), \quad \text{s.t.} \quad C(\tau) \leq \kappa.$$

- Inverse Pareto Frontier (IPF) with $\kappa \in [0, \infty)]$:

$$\text{IPF}(\kappa, \mathcal{T}) = \max_{\tau \in \mathcal{T}} R(\tau), \quad \text{s.t.} \quad C(\tau) \geq \kappa.$$

- Reward Frontier (RF) with $\kappa \in \mathbb{C}$:

$$\text{RF}(\kappa, \mathcal{T}) = \max_{\tau \in \mathcal{T}} R(\tau), \quad \text{s.t.} \quad C(\tau) = \kappa.$$

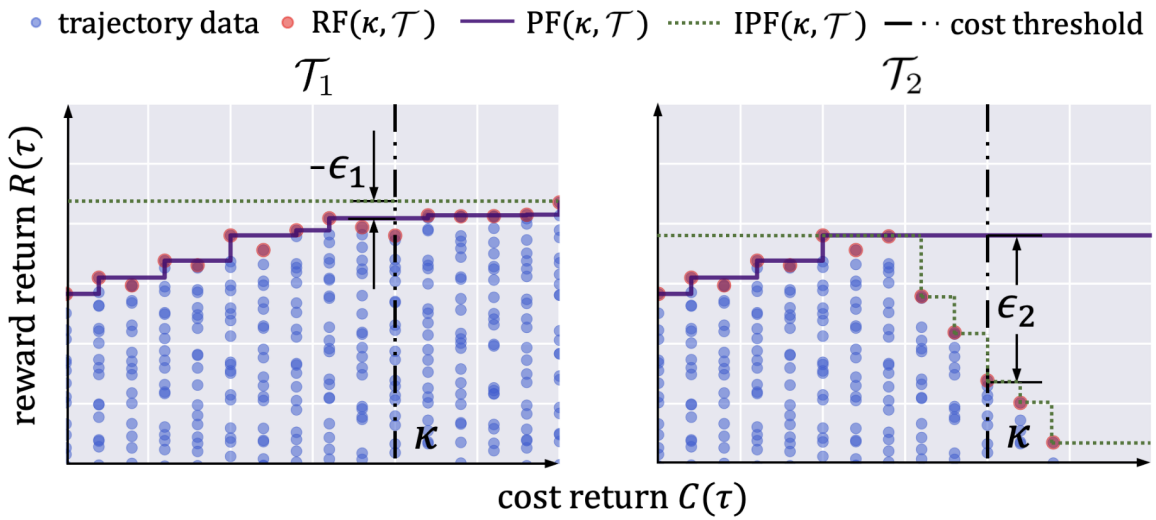
Trade-offs between safety and task performance

$\text{PF}(\kappa, \mathcal{T})$ is a non-decreasing function of κ: Finding a policy with a small cost return usually needs to sacrifice the reward.

Describe Task Difficulty

Definition: ϵ-reducible

An offline safe RL dataset $\mathcal{T}$ is ϵ-reducible w.r.t. threshold κ if:
$\mathrm{PF}(\kappa, \mathcal{T}) = \mathrm{IPF}(\kappa, \mathcal{T}) + \epsilon, \epsilon \in \mathbb{R}$

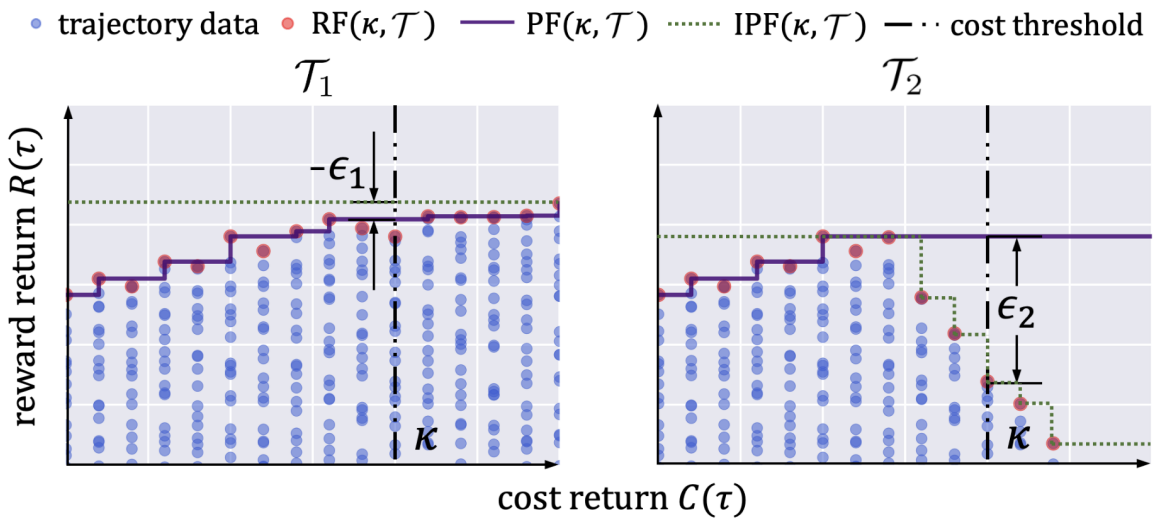


Hypothesis: Suppose problem $(\kappa, \mathcal{T})$ is ϵ-reducible, then

- the smaller ϵ, the more difficult to find the optimal solution;
- the larger ϵ, the more likely to reduce the problem to standard offline RL.

Choose Target Cost-Reward

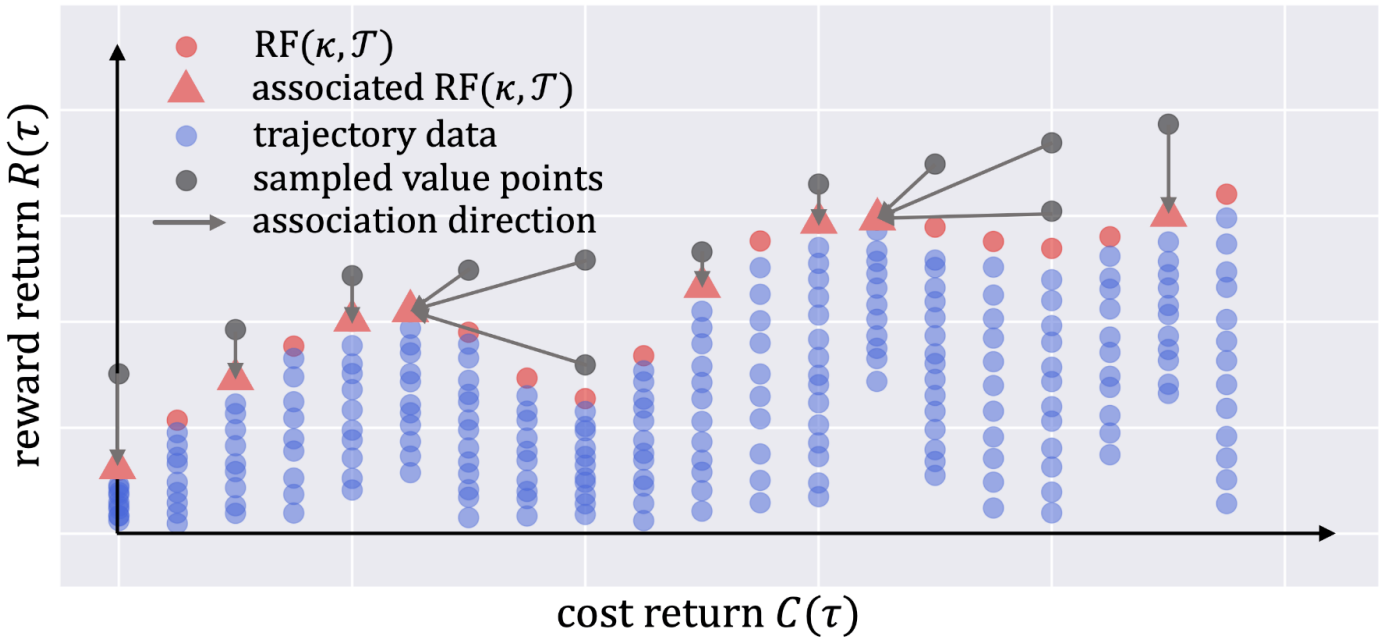
- **In offline RL,** setting a large enough target reward for the agent to maximize the reward.
- **In offline safe RL,** valid target cost-reward return pairs are restricted under the RF points.



How can we resolve the potential conflict between desired returns and ensure the target cost is of higher priority than the target reward?

Data Augmentation by Return Relabeling

1. An infeasible target return pair (ρ, κ), s.t. $\rho > \text{RF}(\rho, \kappa)$.
2. Associate (ρ, κ) with the safe trajectory that of the maximum reward return $\tau^* = \{\mathbf{R}^*, \mathbf{C}^*, \mathbf{s}^*, \mathbf{a}^*\} = \arg\max_{\tau \sim \mathcal{T}} R(\tau), \quad \text{s.t.} \quad C(\tau) \leq \kappa.$
3. Append the new trajectory $\hat{\tau} = \{\mathbf{R}^* + \rho - R(\tau^*), \mathbf{C}^* + \kappa - C(\tau^*), \mathbf{s}^*, \mathbf{a}^*\}$ to the dataset: $\mathcal{T} \leftarrow \mathcal{T} \cup \{\hat{\tau}\}$.



Data Augmentation by Return Relabeling

Algorithm 1 Data Augmentation via Relabeling

Input: dataset $\mathcal{T}$, samples N, reward sample max r_{max}
Output: augmented trajectory dataset $\mathcal{T}$

$c_{min} \leftarrow \min_{\tau \sim \mathcal{T}} C(\tau)$, $c_{max} \leftarrow \max_{\tau \sim \mathcal{T}} C(\tau)$
for $i = 1, ..., N$ **do**
  ▷ *sample a cost return*
  $\kappa_i \sim \text{Uniform}(c_{min}, c_{max})$
  ▷ *sample a reward return above the RF value*
  $\rho_i \sim \text{Uniform}(\text{RF}(\kappa_i, \mathcal{T}), r_{max})$
  ▷ *find the closest and safe Pareto trajectory*
  $\tau_i^* \leftarrow \arg\max_{\tau \sim \mathcal{T}} R(\tau), s.t. \quad C(\tau) \leq \kappa_i$
  ▷ *relabel the reward and cost return*
  $\hat{\tau}_i \leftarrow \{\mathbf{R}_i^* + \rho_i - R(\tau_i^*), \mathbf{C}_i^* + \kappa_i - C(\tau_i^*), \mathbf{s}_i^*, \mathbf{a}_i^*\}$
  ▷ *append the trajectory to the dataset*
  $\mathcal{T} \leftarrow \mathcal{T} \cup \{\hat{\tau}_i\}$
end for

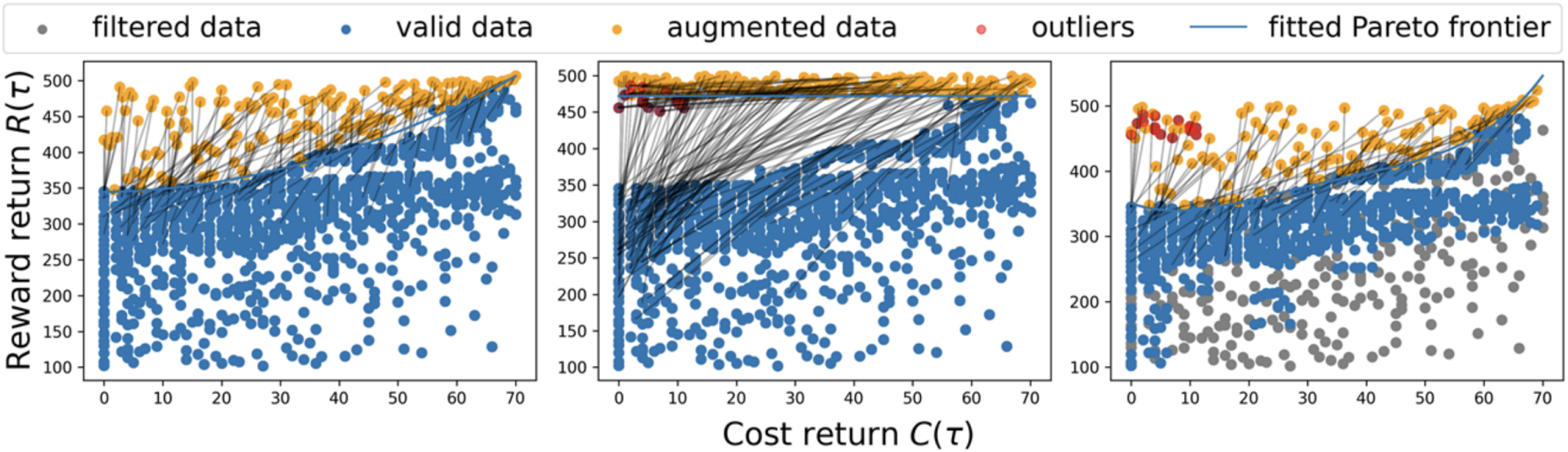
Intuition:

Relabel the associated Pareto trajectory's reward and cost returns, such that the agent can learn to imitate the behavior of the most rewarding and safe trajectory τ^* when the desired return (ρ, κ) is infeasible.

Data Augmentation by Return Relabeling

"Lucky" outliers can disrupt the data augmentation!

- **Sampling-based association :** Associate each augmented return pair (r, c) with a trajectory sampled in the neighborhood of the nearest and safe Pareto frontier data point based on a distance metric.
- **Density filter:** Employ a density filter to remove such outliers exhibiting abnormal reward and cost returns during the creation of the training dataset.



Evaluation

Algorithm 2 Returns Conditioned Evaluation for CDT

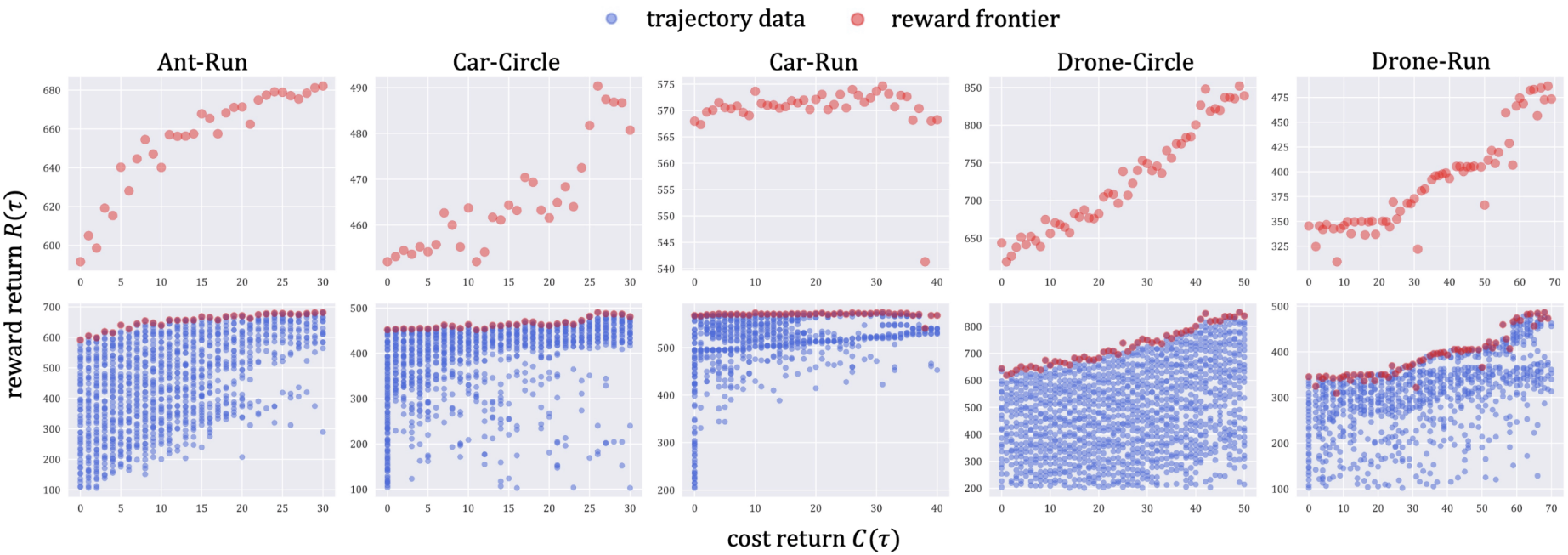
Input: trained Transformer policy π_θ, episode length T, context length K, target reward and cost R_1, C_1, env

Get the initial state: $s_1 \leftarrow \text{env.reset}()$
Initialize input sequence $\mathbf{o} = [\{R_1, C_1, s_1\}]$
for $t = 1, ..., T$ **do**
    Get predicted action $a_t \sim \pi(\cdot|\mathbf{o}[-K:])[-1]$
    Execute the action: $s_{t+1}, r_t, c_t \leftarrow \text{env.step}(a_t)$
    ▷ *compute target returns for the next step*
    $R_{t+1} = R_t - r_t, C_{t+1} = C_t - c_t,$
    Append the new token $o_t = \{R_{t+1}, C_{t+1}, s_{t+1}, a_t\}$ to the sequence $\mathbf{o}$
end for

Experiment

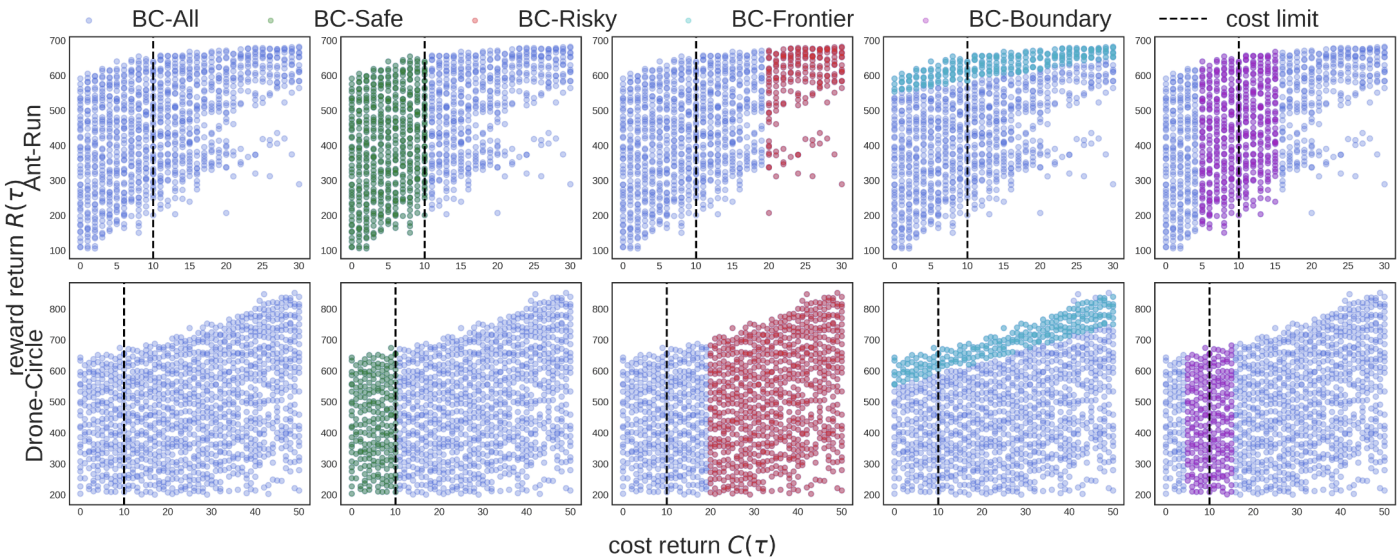
- **Tasks:** Robot locomotion continuous control tasks.
- **Offline datasets:** Collected using the CPPO safe RL approach.
- **Metrics:** Normalize reward return and cost return.

$$R_{\text{normalized}} = \frac{R_\pi - r_{\min}(\mathcal{T})}{r_{\max}(\mathcal{T}) - r_{\min}(\mathcal{T})} \times 100, \qquad C_{\text{normalized}} = \frac{C_\pi}{\kappa + \epsilon}$$



Baselines

- Offline safe RL: **CPQ**, **COptiDICE**
- Lagrangian-based adaptions: **BCQ-Lag**, **BEAR-Lag**
- Vanilla DT with cost return token: **DT-Cost**
- Behavior Cloning: **BC-Safe**, BC-all, BC-risky, BC-frontier, BC-boundary



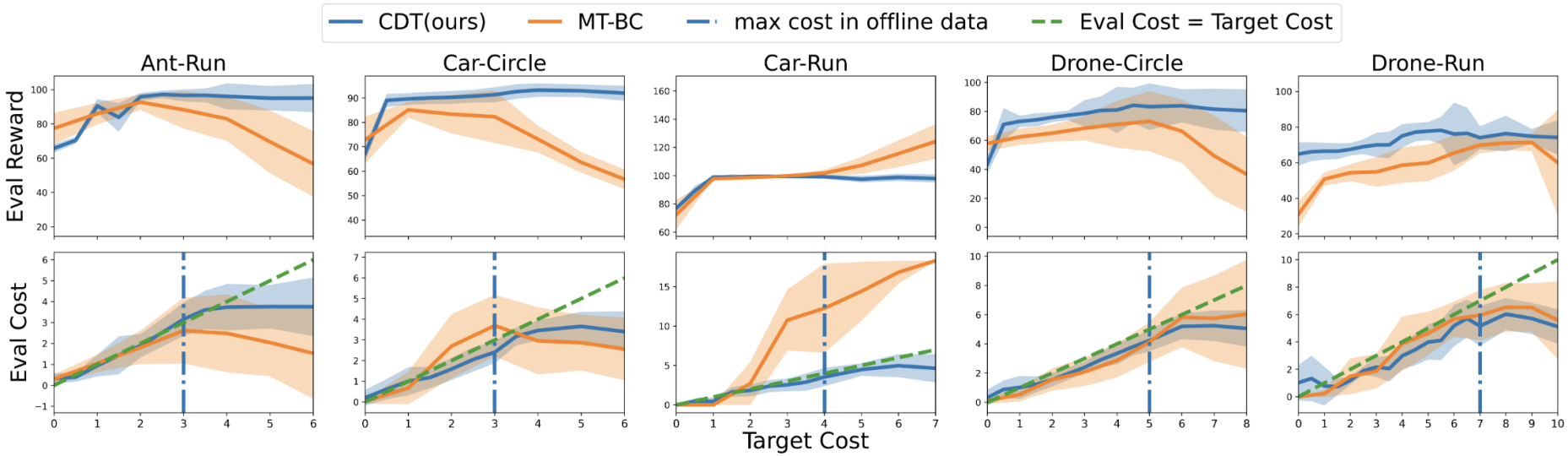
Can CDT learn safe policies from offline datasets?

Methods	Ant-Run		Car-Circle		Car-Run		Drone-Circle		Drone-Run		Average	
	reward ↑	cost ↓	reward ↑	cost ↓	reward ↑	cost ↓	reward ↑	cost ↓	reward ↑	cost ↓	reward ↑	cost ↓
CDT(ours)	**89.76**	**0.83**	**89.53**	**0.85**	**99.0**	**0.45**	**73.01**	**0.88**	**63.64**	**0.58**	**82.99**	**0.72**
BC-Safe	**80.56**	**0.64**	**78.21**	**0.74**	**97.21**	**0.01**	**66.49**	**0.56**	**32.73**	**0.0**	**71.04**	**0.39**
DT-Cost	91.69	1.32	89.08	2.14	100.67	11.83	78.09	2.38	72.3	4.43	86.37	4.42
BCQ-Lag	92.7	1.04	89.76	3.91	96.14	3.21	71.14	3.37	47.61	1.81	79.47	2.67
BEAR-Lag	91.19	1.66	15.48	2.24	**99.09**	**0.09**	72.36	1.99	**19.07**	**0.0**	59.44	1.2
CPQ	**78.52**	**0.14**	**75.99**	**0.0**	**97.72**	**0.11**	55.14	9.67	72.24	4.28	75.92	2.84
COptiDICE	**45.55**	**0.6**	52.17	6.38	**92.86**	**0.89**	36.44	5.54	26.56	1.38	50.72	2.96
CDT(w/o augment)	93.62	1.53	89.8	1.38	99.58	1.89	74.9	1.35	66.93	1.53	84.97	1.54
CDT(w/o entropy)	**87.47**	**0.64**	89.94	1.07	**98.92**	**0.44**	**73.76**	**0.97**	**62.29**	**0.6**	**82.48**	**0.74**
CDT(deterministic)	94.21	1.42	89.53	1.43	101.52	17.53	**76.4**	**1.0**	68.44	1.36	86.02	4.55

Yes, the CDT can learn safe and high-rewarding policies in difficult tasks well.

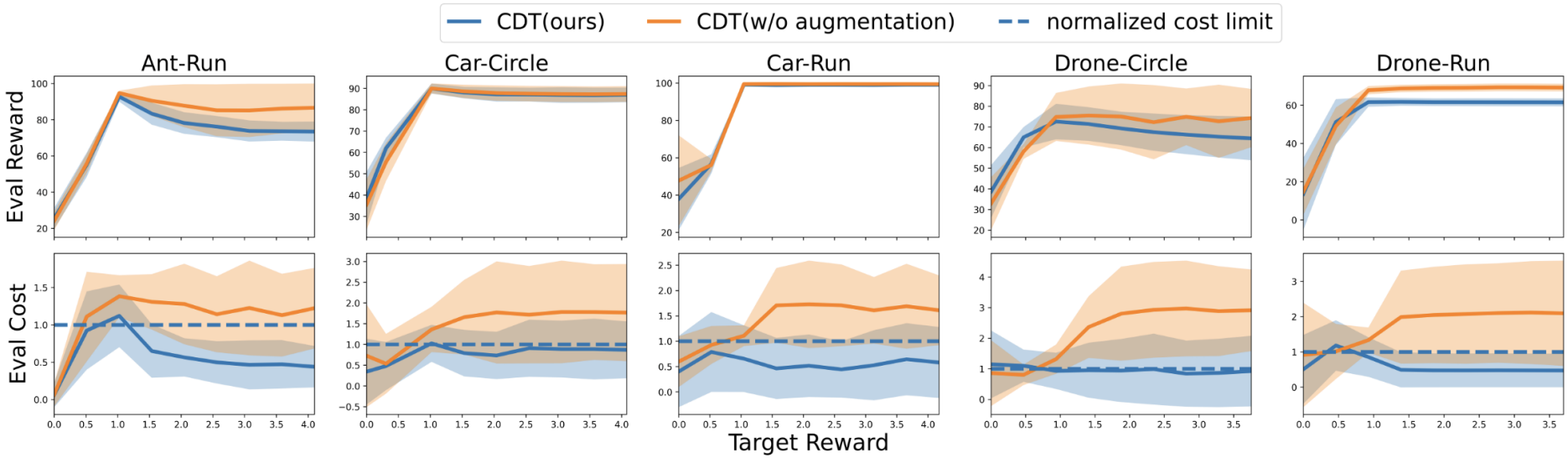
Can CDT achieve zero-shot adaptation to different constraint thresholds?

Baseline: Multi-task behavior Cloning (**MT-BC**)



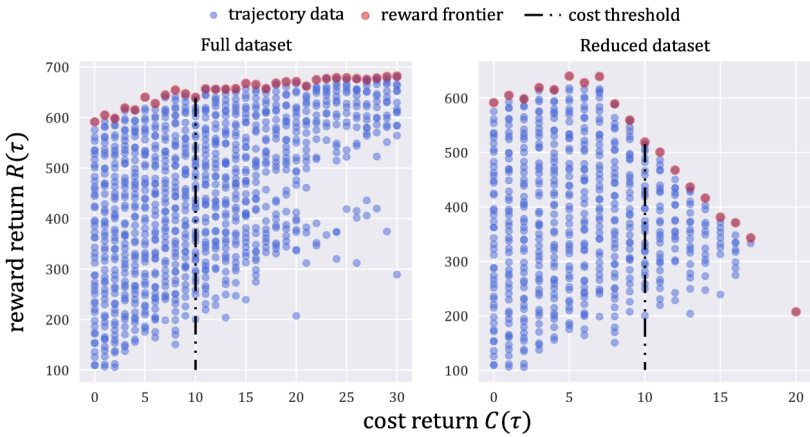
Yes, the CDT shows great interpolation and extrapolation capability.

Is CDT robust to conflict reward returns?

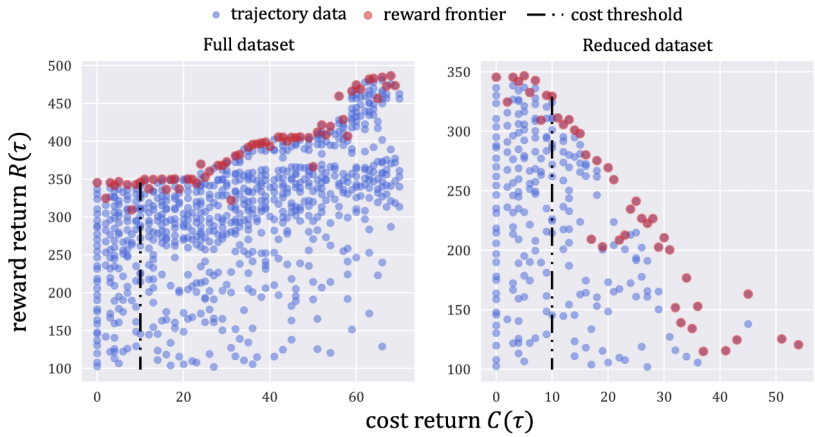


Yes, data augmentation in CDT successfully handles conflicting target returns.

Can ϵ-reducible characterize the task difficulty?



(a) Ant-Run task datasets, $\hat{\epsilon}|_{\kappa=10} = -0.040$ within full dataset, $\hat{\epsilon}|_{\kappa=10} = 0.189$ within reduced dataset.



(b) Drone-Run task datasets, $\hat{\epsilon}|_{\kappa=10} = -0.281$ within full dataset, $\hat{\epsilon}|_{\kappa=10} = 0.102$ within reduced dataset.

Methods	Full Dataset (small ϵ)						Reduced Dataset (large ϵ)					
	Ant-Run ($\hat{\epsilon} = -0.040$)		Drone-Run ($\hat{\epsilon} = -0.281$)		Average		Ant-Run ($\hat{\epsilon} = 0.189$)		Drone-Run ($\hat{\epsilon} = 0.102$)		Average	
	Reward ↑	Cost ↓	Reward ↑	Cost ↓	Reward ↑	Cost ↓	Reward ↑	Cost ↓	Reward ↑	Cost ↓	Reward ↑	Cost ↓
DT	96.13	2.47	49.09	4.69	72.61	3.58	**81.26**	**0.91**	63.88	1.03	**72.57**	**0.97**
BCQ	97.99	5.39	60.1	3.01	79.04	4.2	82.59	1.19	**42.34**	**0.51**	**62.46**	**0.85**
BEAR	91.05	1.69	**42.13**	**0.48**	66.59	1.08	**84.2**	**0.55**	**33.29**	**0.0**	**58.74**	**0.28**

Yes, larger ϵ-reducible problems are relatively easier to solve for the same task, as using standard offline RL algorithms can achieve good performance.

Contribution

- Leverage a novel multi-objective optimization (MMO) perspective and the return-conditioned sequential modeling capability of Transformer to offline safe RL.
- Propose three key techniques in CDT, enabling the first successful offline safe RL approach that can achieve zero-shot adaptation to different safety requirements after training.
- CDT significantly outperforms prior methods in both safety and reward, demonstrating robust generalization across different cost thresholds.

Recommendation

Is it worth reading? Yes.

- Address the critical challenge of offline safe RL.
- Provide compelling experimental evidence.
- Highly detailed and well-documented, featuring extensive visualizations for helping understanding.

Is it worth implementing? Maybe.

- Clinical applications are highly safety-critical, and clinical data are inherently offline collected.
- Data characteristic gaps between clinical vs. robotics applications: data quality, noisy data, action space, metrics of reward and cost.