Supplementary Material for: "Beyond the Known: Decision Making with Counterfactual Reasoning Decision Transformer"

A Stiching Behavior in Sequential Modeling

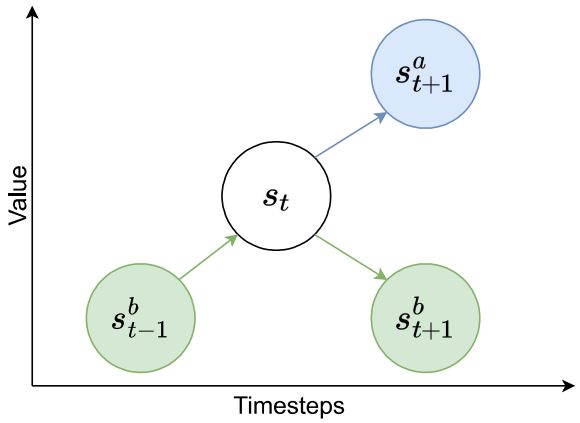

Figure 3: Given two trajectories $\left(s^a_{t-1}, s_t, s^a_{t+1}\right)$, $\left(s^b_{t-1}, s_t, s^b_{t+1}\right)$. We want our agent to be able to start from state s^b_{t-1}, however, can reach state s^a_{t+1}

Trajectory stitching is an ability that has received great attention lately in offline RL, specifically, in sequential modeling. It has been proven that traditional sequential modeling approaches, such as [2021], lack the ability to stitch suboptimal trajectories to form optimal trajectories [Kumar *et al.*, 2020; Zhuang *et al.*, 2024; Wu *et al.*, 2024]. An example of this is given in the toy environment provided in Fig. 1(a) and Fig. 3. Let's consider the scenario of two trajectories $\left(s^a_{t-1}, s_t, s^a_{t+1}\right)$, $\left(s^b_{t-1}, s_t, s^b_{t+1}\right)$ where $\left(s^a_{t-1}, s_t, s^a_{t+1}\right)$ is sampled from the set of blue trajectories (good trajectories that lead to the goal) and $\left(s^b_{t-1}, s_t, s^b_{t+1}\right)$ is sampled from the set of green trajectories (bad trajectories that do not lead to the goal). We anticipate that a sequence model trained on these trajectories will likely follow the subsequent states in a way that aligns with the provided trajectories. This means that the agent starting from the bottom-left potentially follows the green trajectories to the top-right of the maze and will not reach the goal. We want, however, to stitch these trajectories together, meaning that we want our agent to be able to start from s^b_{t-1} but end up being in s^a_{t+1}.

The explanation for why traditional DT does not have stitching ability arises from the agent's training conditions. Specifically, when using traditional DT, the prediction of the next state-action pair is conditioned on an initial target return (g_0). If g_0 is set to 0, the ball will smoothly follow the green trajectory, as this is the more common data and the returns-to-go at the crossroad (the point where the green and blue trajectories intersect) are still equal to 0. On the other hand, if conditioned on a return of 1, the ball is likely to take a random action because $g_0 = 1$ represents an out-of-distribution (OOD) returns-to-go from the bottom-left corner of the maze. In both cases, the ball fails to reach the goal. Previous works, such as [2024] and [2024] address this problem by modifying the training condition of the DT agent. Our approach, on the other hand, addresses this problem by generating better trajectories based on the idea of potential outcome, thus, guiding the agent to reach the goal.

B Potential Outcome Framework and Assumptions for Causal Identification

We build upon the potential outcomes framework [Neyman, 1923; Rubin, 1978] and its extension to time-varying treatments and outcomes [Robins and Hernan, 2008]. In order to identify the counterfactual outcome distribution over time, the following three standard assumptions for the data-generating process are required:

Assumption A.1 (Consistency): If $\bar{A}_t = \bar{a}_t$ is a fixed sequence of treatments for a particular patient, then $Y_{t+1}[\bar{a}_t] = Y_{t+1}$. This implies that the potential outcome under the treatment sequence $\bar{a}_t$ corresponds to the observed (factual) outcome for the patient, conditional on $\bar{A}_t = \bar{a}_t$.

Mapping to offline RL, consistency means that for any given action a_t the observed next state s_{t+1} and returns-to-go g_{t+1} reflect the true outcome of the action. In the context of offline RL this assumption holds given that observational data is collected from behaviour policies π_β that were trained in the same environment; therefore, the data reflects the actual dynamics of the environment.

Assumption A.2 (Sequential Overlap): For every history, there is always a non-zero probability of receiving or not receiving any treatment over time:

$$0 < P(A_t = a_t | \bar{H}_t = \bar{h}_t) < 1, \quad \text{if} \quad P(\bar{H}_t = \bar{h}_t) > 0,$$

where $\bar{h}_t$ is a particular historical experience.

For offline RL, sequential overlap guarantees that for any observed history h_t, every action a_t has a non-zero probability of being chosen. This assumption is met if D_{env} provides adequate coverage of the state-action space. If the behaviour policy π_β used to collect the data explores a wide range of actions under different histories, we can reasonably assume that the sequential overlap condition hold.

Assumption A.3 (Sequential Ignorability): This states that the current treatment is independent of the potential outcome, given the observed history:

$$A_t \perp Y_{t+1}[a_t] \mid \bar{H}_t, \forall a_t.$$

This means there are no unmeasured confounders that simultaneously influence both the treatment and the outcome.

Sequential ignorability implies that the observed history h_t includes all relevant information that influences both the agent's actions and the potential future outcomes. Since we only perform counterfactual reasoning on observed data in D_{env}, we rely on the assumption that the dataset sufficiently captures the relevant factors affecting the treatments and the resulting outcomes.

In prior works, these assumptions are applied to both environments with discrete or continuous treatments [Melnychuk *et al.*, 2022; Frauen *et al.*, 2023; Bahadori *et al.*, 2022]

C Details of DT Learning to Reason Counterfactually

Algorithm 1 Learrning to Reason Counterfactually Algorithm

Require: Offline environment dataset D_{env}.

Initialize: Treatment model $\mathcal{T}$, Outcome model $\mathcal{O}$.
for $k = 1, \ldots, K$ **do**
    Sample batch: $\tau = \left(g_t^{(i)}, s_t^{(i)}, a_t^{(i)}\right)_{t=1}^{T}, \quad i = 1, 2, \ldots, N$ from D_{env}.
    Update $\mathcal{T}$ by minimizing loss $\mathcal{L}_{\mathcal{T}(\theta)}$ with Eq. 4 or Eq. 5 using data from τ.
    Update $\mathcal{O}$ by minimizing loss $\mathcal{L}_{\mathcal{O}(\eta)}$ with Eq. 6 using data from τ.
end for

D Details of DT Counterfactual Reasoning

Algorithm 2 DT Counterfactual Reasoning Algorithm

Require: Offline environment dataset D_{env}, Treatment model $\mathcal{T}$, Outcome model $\mathcal{O}$, number of action sampled n_a, number of experiences wanted n_e, and function $U^{\alpha}(\mathbf{S}_{t+1})$ from Eq. 9.

Initialize: Counterfactual experience buffer D_{crdt}.
for $k' = 1, \ldots, K'$ **do**
    Sample batch: $\tau' = \left(g_t^{(i)}, s_t^{(i)}, a_t^{(i)}\right)_{t=1}^{T}, \quad i = 1, 2, \ldots, N'$ from D_{env}.
    for $\tau'^{(i)}$ in τ **do**
        for $t = \frac{T}{2}$ to T **do**
            Init: $h_{t-1} = (g_1, s_1, a_1, \ldots, g_{t-1}, s_{t-1}, a_{t-1})$.
            for $j = 1$ to n_a **do**
                $\hat{a}_t^{(j)} \leftarrow \mathcal{T}(h_{t-1}, s_t, g_t; \theta), \quad j = 1, 2, \ldots, n$
                Init: $\hat{h}_t^{(j)} = (g_1, s_1, a_1, \ldots, g_t, s_t, \hat{a}_t^{(j)})$.
                if Not $U^{\alpha}(\mathbf{S}_{t+1})$ **then**
                    $\hat{s}_{t+1}^{(j)}, \hat{g}_{t+1}^{(j)} \leftarrow \mathcal{O}(\hat{h}_t^{(j)})$.
                    if $\hat{g}_{t+1} < g_{t+1}$ **then**
                        $a_t, s_{t+1}, g_{t+1} = \hat{a}_t^{(j)}, \hat{s}_{t+1}^{(j)}, \hat{g}_{t+1}^{(j)}$.
                    else
                        Continue.
                    end if
                else
                    Continue
                end if
            end for
            if $t = T$ **and** $\text{len}(D_{\text{crdt}}) < n_e$ **then** $D_{\text{crdt}} \leftarrow \tau'^{(i)}$.
            end if
        end for
    end for
end for

Counterfactual Action Selection in Continuous Action Space

We aim to select n_a actions as our counterfactual actions. The selection of these actions in a continuous action space environment is inspired by the theory of maximum Gaussian random variables [Kamath, 2015]. The expectation of maximum of Gaussian random variables are bounded as:

$$0.23\sigma \cdot \sqrt{\ln(n)} \leq \mathbb{E}\left[\max(x - \mu)\right] \leq \sqrt{2}\sigma \cdot \sqrt{\ln(n)}.$$

where μ is the mean of the distribution and σ is the standard deviation. Applying this equation to our approach, wherein continuous action is assumed to follow a normal Gaussian distribution. Thus, for an action a_t, at timestep t, we can rewritten the equation into:

$$0.23\sigma \cdot \sqrt{\ln(n)} \leq \mathbb{E}\left[\max(a_t - \mu_t)\right] \leq \sqrt{2}\sigma_t \cdot \sqrt{\ln(n)},$$

or

$$0.23\sigma \cdot \sqrt{\ln(n)} + \mu_t \leq \mathbb{E}\left[\max(a_t)\right] \leq \sqrt{2}\sigma_t \cdot \sqrt{\ln(n)} + \mu_t.$$

We choose to use the upper bound of this equation as the bound for our outlier actions, thus, from this bound, we will start searching for a number of n_a outlier actions. The bound can be written as:

$$E\left[\max(a_t)\right] \leq \mu_t + \sqrt{2}\sigma_t\sqrt{\ln(n_{enc})}.$$

As $\Phi^{-1}(0.08) \approx -\sqrt{2}$. We can derive our formula to calculate each action:

$$a_t = \mu_t - \Phi^{-1}(0.08 - j \cdot \beta)\,\sigma_t\sqrt{\ln(n_{enc})}.$$

where β is the step size and $j = 0, 1, \ldots, n_a$ indicates the index of the j-th action from the total n_a sampled counterfactual actions. Φ^{-1} is the quantile function of the standard normal distribution. When $j = 0$, the value of $\Phi^{-1}(0.08 - j \cdot \beta) = \Phi^{-1}(0.08) \approx -\sqrt{2}$, thus:

$$\mu_t - \Phi^{-1}(0.08)\,\sigma_t\sqrt{\ln(n_{enc})} \approx \mu_t + \sqrt{2}\sigma_t\sqrt{\ln(n_{enc})}.$$

Here, n_{enc} denotes the number of times the model has encountered an input (h_t, s_{t+1}, g_{t+1}). In a continuous environment, recording the counting for such input is difficult. Thus, we employed a hashing function, specifically, we used the hashlib.md5() hashing function [3] in our implementation to record the input as key and the counting as the value in a dictionary. As MD5 hashing looks for an exact match of data, we expect that such hashing process will only help with saving memory and not affect the overall result of the method.

Compute the Maximum Variance between Predictions

In this section, we present our method that was used to compute the maximum variance of the predictions in $\mathbf{S}_{t+1}$ using the function $\text{Var}(\mathbf{S}_k)$. $\mathbf{S}_{t+1} = \begin{bmatrix} s_{t+1}^{(1)} & s_{t+1}^{(2)} & \cdots & s_{t+1}^{(m)} \end{bmatrix}$ is the matrix of state predictions at timestep $t+1$, output from m forward passes of the Outcome model $\mathcal{O}$ that was trained with dropout regularization layers.

Thus, $\mathbf{S}_{t+1} \in \mathbb{R}^{m \times d}$, where m is the number of predictions and d is the dimension of each prediction. Each row of $\mathbf{S}_{t+1}$, denoted as $\mathbf{s}_{t+1}^{(i)} = \begin{bmatrix} s_{t+1}^{(i,1)} & s_{t+1}^{(i,2)} & \cdots & s_{t+1}^{(i,d)} \end{bmatrix}$, represents a predicted state at timestep $t+1$, where $i = 1, 2, \ldots, m$. Each $\mathbf{s}_{t+1}^{(i)}$ is a d-dimensional vector representing the state in the predicted space.

The variance for each dimension of the predicted states is computed using the covariance matrix of $\mathbf{S}_{t+1}$. The covariance matrix $\Sigma_k \in \mathbb{R}^{d \times d}$ is defined as:

$$\Sigma_k = \frac{1}{m-1}\sum_{i=1}^{m}\left(\mathbf{s}_{t+1}^{(i)} - \bar{\mathbf{s}}_{t+1}\right)\left(\mathbf{s}_{t+1}^{(i)} - \bar{\mathbf{s}}_{t+1}\right)^T,$$

where $\bar{\mathbf{s}}_{t+1}$ is the mean of the predicted states, $\bar{\mathbf{s}}_{t+1} = \frac{1}{m}\sum_{i=1}^{m}\mathbf{s}_{t+1}^{(i)}$.

The variance for each dimension j' (for $j' = 1, 2, \ldots, d$) is then extracted from the diagonal elements of Σ_k, denoted as:

$$\text{Var}(\mathbf{s}_{t+1}^{(j')}) = \Sigma_{k,j'j'}.$$

This allows us to get the maximum variance across all dimensions:

$$\text{Var}(\mathbf{S}_{k=t+1}) = \max\left(\Sigma_{k,11}, \Sigma_{k,22}, \ldots, \Sigma_{k,dd}\right).$$

In environments with image observation space such as Atari games, calculating the covariance matrix from raw observations is computationally expensive. Thus, we use the encoded observations, from the Outcome model, to form the prediction matrix instead. Thus $\mathbf{S}_{t+1} = \begin{bmatrix} \phi(s)_{t+1}^{(1)} & \phi(s)_{t+1}^{(2)} & \cdots & \phi(s)_{t+1}^{(m)} \end{bmatrix}$, where $\phi(s)$ denotes the encoding.

Choosing the Uncertainty Threshold

Our strategy to determine the uncertainty threshold α for each testing environment and dataset is inspired by the process used in [Kidambi *et al.*, 2020]. Specifically, we compute the accumulated maximum variance $\sum_{k=t_0}^{t+1}\max\left(\text{Var}(\mathbf{S}_k)\right)$ over several batch data (we use 1000 samples in this paper) sampled from the static dataset D_{env}. Then, we compute the mean μ_d, the standard deviation σ_d over all the accumulated maximum variance that we have collected. The uncertainty threshold is then $\alpha = \mu_d + \sigma_d \cdot \varsigma$. We tune the value of ς in steps of 0.5. The final uncertainty threshold α for each environment is presented in Appendix. F.

[3] https://docs.python.org/3/library/hashlib.html

E Details of DT Optimize Decision-Making with Counterfactual Experience

Algorithm 3 Optimize Decision-Making with Counterfactual Data Algorithm

Require: Offline environment dataset D_{env}, Counterfactual experience buffer D_{crdt}.

Initialize: $\mathcal{M}$ agent.

for $k = 1, \ldots, K$ **do**

  Sample batch: $\tau = \left(g_t^{(i)}, s_t^{(i)}, a_t^{(i)}\right)_{t=1}^{T}, \quad i = 1, 2, \ldots, N$ from D_{env}.

  Calculate loss $\mathcal{L}_{\mathcal{M}(\delta)}^{\text{env}}$ with Eq. 10 or Eq. 11 using data from τ.

  Sample batch: $\tau' = \left(g_t^{(i)}, s_t^{(i)}, a_t^{(i)}\right)_{t=1}^{T}, \quad i = 1, 2, \ldots, N'$ from D_{crdt}.

  Calculate loss $\mathcal{L}_{\mathcal{M}(\delta)}^{\text{crdt}}$ with Eq. 10 or Eq. 11 using data from τ'.

  Update $\mathcal{M}$ by minimizing loss $\mathcal{L}_{\mathcal{M}(\delta)} = \mathcal{L}_{\mathcal{M}(\delta)}^{\text{env}} + \mathcal{L}_{\mathcal{M}(\delta)}^{\text{crdt}}$.

end for

F Additional Experiment Details

Details of Baselines

In our paper, we have compare CRDT against a number of baselines including including conventional RL and sequential modeling techniques. Conventional methods include Behavior Cloning (BC) [Pomerleau, 1988], model-free offline methods, such as Conservative Q-Learning (CQL) [Kumar *et al.*, 2020] and Implicit Q-Learning, (IQL) [Kostrikov *et al.*, 2021b] and model-based offline methods, such as MOPO [Yu *et al.*, 2020] and MOReL [Kidambi *et al.*, 2020]. Sequential modeling baselines include simple backbone DT [Chen *et al.*, 2021], Elastic Decision Transformer (EDT) [Wu *et al.*, 2024] and state-of-the-art Reinformer (REINF) [Zhuang *et al.*, 2024]. In this section, we will clarify which results we have get from the original paper, and which results we have reproduced and where the source code is from. Given limited computational resources, our focus is on reproducing the result of sequential modeling approaches, which are our direct comparing baselines.

- The results of BC in Table. 1 comes from the REINF paper [Zhuang *et al.*, 2024], whereas the results of BC in Table. 2 is from the original DT paper [Chen *et al.*, 2021].
- The results of model-free offline RL methods, CQL and IQL, are in Table. 1, also comes from the REINF paper [Zhuang *et al.*, 2024]. While the results of model-based offline RL methods, MOPO and MOReL, are obtained straight from their original papers [Yu *et al.*, 2020; Kidambi *et al.*, 2020].
- The results of EDT and REINF, in Table. 1, are reproduced using the source codes provided by the authors (MIT licence)[4] for all the Locomotion tasks and Ant tasks, using the hyperparameters that were provided in the associated papers. For REINF, we also ran the source code on the Ant environments for a comprehensive comparison, the hyperparameters that were used are the default hyperparameters that come with the code. For Maze tasks in Fig.2, we reproduce the results of REINF using the hyperparameters provided in the paper.
- All the results of DT are reproduced using the source code provided by the authors (MIT licence)[5].

Details of Dataset and Environments

We compare our CRDT algorithm against baselines on several datasets. These include those with continuous action space environments and those that come with discrete action space environments. This is to provide a comprehensive test for the Counterfactual Action Selection and the Counterfactual Action Filtering mechanism. In this section, we provide an overview of the testing environment.

Continuous action space environments include Locomotion, Ant, and Maze2d tasks from the D4RL benchmark [Fu *et al.*, 2020]. The environments within Locomotion include hopper, halfcheetah and walker. For each of the Locomotion environments and the Ant environments, we have 3 types of dataset medium-replay (med-rep), medium (med), and medium-expert (med-exp). The environments within Maze2d environments include large and umaze; each of these environments has its corresponding dataset maze2d-large and maze2d-umaze. We obtain the datasets for Locomotion and Maze tasks using the code associated with the Reinformer paper [Zhuang *et al.*, 2024], while, the dataset for Ant tasks are collected using the code associated with the Elastic Decision Transformer paper [Wu *et al.*, 2024]. We evaluate our algorithms using gym environments from gym package ver. 0.18.3 [Brockman, 2016].

Discrete action space environments include Breakout, Qbert, Pong and Seaquest. The data for these environments were collected using the code provided in the original Decision Transformer paper [Chen *et al.*, 2021]. We also use the evaluation code provided in this paper to evaluate our algorithm. Specifically, we use ale-py package ver. 0.8.1 [Bellemare *et al.*, 2013] for evaluation.

Performance of CRDT on Continuos Action Space Environment (Detailed)

See Table 5 for a detailed performance comparison on continuous action space environments, including Locomotion and Ant. The analysis is in Sect. 4.

Performance of CRDT on Limited Subset of D_{env} (Detailed)

See Fig. F.1 for a detailed performance comparison on limited subset of D_{env}. The analysis is in Sect. 4.

Atari Raw Scores

We present the raw score of the experiments on Atari games in Table. 6. These results correspond to the normalized results presented in Table. 2. For the purpose of normalization, we used the data in Table. 7. This is similar to the process of normalization that have been used in [2021].

Performance of CRDT on Modified Evaluating Environments

We further evaluate CRDT's generalizability by testing its performance in modified environments, where the dynamics differ from those in the D_{env} dataset generated by the behaviour policy π_β. Out of the four environments tested, in Table. 8, CRDT

[4] https://github.com/kristery/Elastic-DT, https://github.com/Dragon-Zhuang/Reinformer

[5] https://github.com/kzl/decision-transformer

Table 5: Performance comparison on Locomotion and Ant tasks. We rort the results over 5 seeds. For each seed, evaluation is conducted over 100 episodes. The best result is shown in **bold**, and the second-best is in *italic*. ♠ denotes the best Sequence Modeling approaches.

Dataset	Traditional Methods					Sequence Modeling Methods			
	BC	CQL	IQL	MOPO	MOReL	DT	EDT	REINF	**CRDT**
halfcheetah-med	42.6	*44.0*	**47.4**	42.3	42.1	42.6	42.1	42.8♠	42.82±2.3♠
halfcheetah-med-rep	36.6	*45.5*	42.4	**53.1**	40.2	36.1	37.8	38.3♠	38.03±2.5
halfcheetah-med-exp	55.2	*91.6*	86.7	63.3	53.3	90.2	82.0	91.2	**96.4±2.3♠**
hopper-med	52.9	58.5	66.3	28.0	**95.4**	67.9	59.6	*75.2*♠	67.94±1.5
hopper-med-rep	18.1	**95**.0	*94.7*	67.5	93.6	85.0	76.1	84.2	85.54±3.2♠
hopper-med-exp	52.5	105.4	91.5	23.7	*108.7*	*108.7*	92.4	107.6	**110.37±0.1♠**
walker2d-med	75.3	72.5	78.3	17.8	77.8	75.9	66.4	*77.9*	**78.95±0.9♠**
walker2d-med-rep	26.0	**77.2**	*73.9*	39.0	49.8	62.1	58.1	72.1	72.28±0.1♠
walker2d-med-exp	107.5	*108.8*	**109.6**	44.6	95.6	108.5	106.9	108.7	**109.05±0.6♠**
Locomotion	466.7	*698.5*	692.6	378.0	656.5	677.0	621.4	698.0	**701.38±1.5**
ant-med-rep	-	-	**92.0**	-	-	90.0	85.0	*91.6*♠	91.02±8.8
ant-med	-	-	*93.9*	-	-	91.5	90.7	92.7	**95.84±8.3♠**
Ant	-	-	*186*	-	-	181.5	175.7	184.3	**186.86±8.5**

Table 6: Performance comparison (raw score) on various Atari games. We report the results over 3 seeds. For each seed, evaluation is conducted over 10 episodes. The best result is shown in **bold**. ♠ indicates games in which CRDT improves the backbone DT approach.

Game	BC	DT	CRDT (Ours)
Breakout	138.9 ± 17.3	57.6 ± 1.5	**71.7 ± 18.5♠**
Qbert	**2464.1 ± 1948.2**	1118.6 ± 195.6	1155.3 ± 89.2♠
Pong	9.7 ± 7.2	**29.5 ± 1.9**	15.8 ± 3.34
Seaquest	968.6 ± 133.8	2494.0 ± 2732.6	**3190.6 ± 264.6♠**

improves DT's performance in three. In contrast, REINF shows weaker results in these environments, likely due to its architecture, which forces it to maximize returns within D_{env}—a condition that may not hold in the modified environments. CRDT excels compared to the original DT method because it generates additional counterfactual experiences, enabling it to cover a broader range of scenarios than the D_{env} dataset alone.

Performance Comparison on Random Dataset

Refer to Table 9, we compare the performance of CRDT against other sequential modelling methods on the random dataset. CRDT outperforms other methods on halfcheetah and walker2d environments. A note here is that we did not perform parameters tuning for REINF and EDT, but used the suggested parameters for med-rep dataset from their papers. Interestingly, DT performs unexpectedly well on hopper-rand, which is a noteworthy observation.

Changing Counterfactual Experience Size

We conduct this experiment to show the impact of varying the number of counterfactual experiences n_e recorded in D_{crdt}. The experiment was conducted on the 10% of the walker2d-medium-replay dataset. Our expectation is that a higher number of experiences the higher the performance. We evaluate the performance with 4000 samples (corresponding to the 10% result in Fig. 2(c)), 8000 samples, and 16000 samples; the result is presented in Fig. F.2. The figure reveals an upward trend in performance as the number of recorded samples increases, validating our expectation. With 16000 samples, CRDT achieves approximately 59 points (10 points higher than when using 4000 samples), closely approaching the performance of DT trained on the entire dataset (approximately 62 points as shown in Table. 1). However, the performance gains also diminish as the number of samples increases. While the improvement from 4000 to 8000 samples is around 6 points, the increase from 8000 to 16000 samples is only about 3 points. Moreover, generating more counterfactual experiences demands greater computational resources, underscoring the balance between performances and computational resources.

Changing Number of Search Actions

We conduct an additional experiment to assess the impact of varying the number of search actions, n_a, on the walker2d-medium-replay dataset. We specifically test 3, 5, 7, and 9 actions, with the results presented in Fig. F.3. As shown, increasing the number of actions generally improves the performance of the DT agent, aside from an outlier when $n_a = 3$. However, using $n_a = 3$ also results in a significantly higher variance in performance and produces the lowest score among the four configurations. This finding aligns with our expectation that increasing the number of actions would broaden the diversity of covered states, enabling the agent to learn more about the environment and improve its performance.

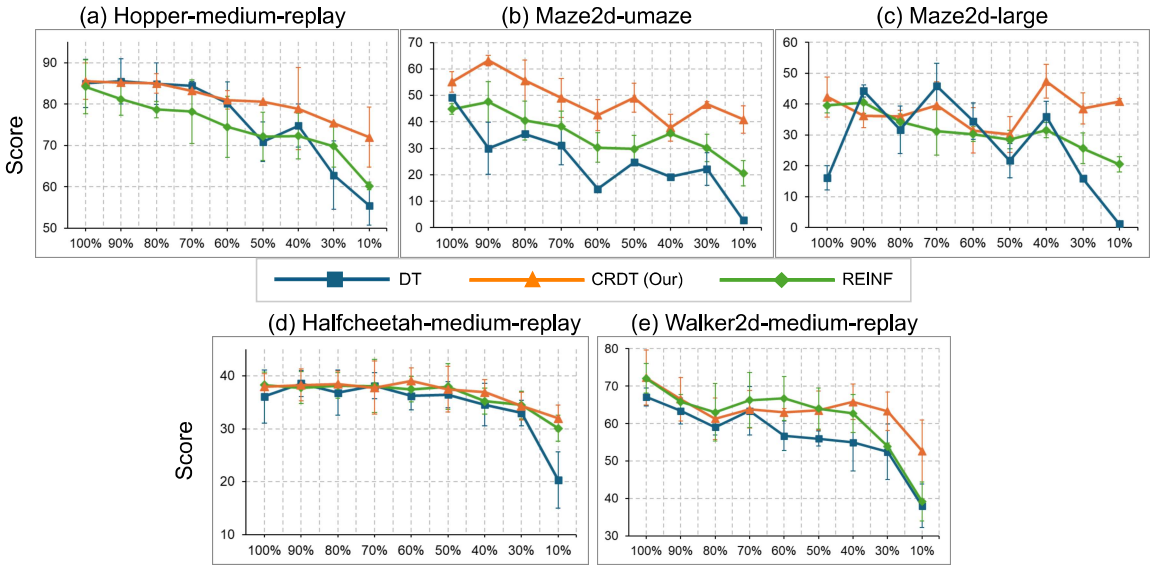

Figure F.1: Performance comparison on limited subset of D_{env} dataset. We report the results over 5 seeds. For each seed, evaluation is conducted over 100 episodes. The X-axis represents the percentage of the dataset used in the experiment.

Table 7: Atari Baseline Scores.

Game	Random	Gamer
Breakout	2	30
Qbert	164	13455
Pong	-21	15
Seaquest	68	42055

CRDT (REINF) and CRDT (EDT)

In Table 10, we present the results of using CRDT with REINF [Zhuang *et al.*, 2024] and EDT [Wu *et al.*, 2024] as the backbone algorithms. A note here is that we only replace decision-making agent $\mathcal{M}$ with the new backbone and not model $\mathcal{T}$ and $\mathcal{O}$.

CRDT (REINF): Although CRDT with REINF shows slight improvements over CRDT with the original DT on the Locomotion and Ant tasks, its performance is significantly lower on the Maze2d tasks. We attribute this decline in performance to the increased difficulty of the Maze2d tasks. Additionally, the underlying REINF algorithm likely requires parameter tuning, especially when integrating new counterfactual experiences. This tuning was not conducted in our study, which may have led to the observed decrease in performance. Here, we used the original parameters provided in the REINF paper for the backbone algorithm. Overall, CRDT with the Reinformer backbone still improve the results of REINIF, as presented in Table 1, albeit only marginally.

CRDT (EDT): Similarly, the result of using EDT as the decision-making also indicates an improve in performance over Locomotion tasks when comparing to the EDT's results provided in Table 1. We saw a noticeable improvement on walker2d-med-rep task of approximately 20%. The result on Ant tasks indicates a marginally improvement. The result overall performance, however, is still not as good as when using CRDT (DT) or CRDT (REINF).

Visualizing the Distribution of Counterfactual Actions and Original Actions

We refer to Fig. F.4 and F.5, where we illustrate the frequency distribution of action values across dimensions in the walker2d-med-rep and halfcheetah-med-exp respectively, between the counterfactual and the original actions. We compute the value over the whole original dataset provided by D4RL, while for the counterfactual samples, we compute the value over 4000 samples. One can see that in Fig. F.4, the distribution across the last 5 dimensions differs, while in Fig. F.5, the differences are in all 6 dimensions. We hypothesize that these significant distribution differences may have contributed to the greater improvement in

Table 8: Performance comparison on modified evaluating environments. We report the results over 5 seeds. For each seed, evaluation is conducted over 100 episodes. The best result is shown in **bold**.

Dataset	Modification	DT	REINF	CRDT (Ours)
hopper-med-rep	head	326.5	348.0	**359.54 ± 47.5**
hopper-med-rep	thigh	**2930.5**	2841.6	2879.4 ± 421.5
halfcheetah-med-rep	head	582.1	371.6	**617.2 ± 32.3**
halfcheetah-med-rep	thigh	1966.6	1345.2	**2070.8 ± 264.6**

Table 9: Performance comparison between DT, REINF, EDT, CRDT on random D4RL dataset. We report the results over 3 seeds. For each seed, evaluation is conducted over 100 episodes.

Dataset	Sequence Modeling Methods			
	DT	EDT	REINF	CRDT
halfcheetah-rand	2.01±2.27	0.82±2.58	-	**2.21±2.28**
hopper-rand	10.5±0.27	3.97±0.39	9.98±0.30	9.59±0.44
walker2d-rand	1.20±0.10	0.77±0.35	0.71±0.17	**2.60±0.42**

the walker2d-med-rep and halfcheetah-med-exp environments, as demonstrated in Table 1.

Details of Hyperparameters

In this paper, we have introduced a number of new parameters. This is divided into those that were used in discrete action space environments and those that were used in continuous action space environments. Apart from these parameters, we also have the parameters of the backbone DT algorithm. The same hyperparameters were used for the Treatment model $\mathcal{T}$, the Outcome model $\mathcal{O}$ and the agent $\mathcal{M}$.

Continuous Action Space Environments

We follow the hyperparameters proposed in the original paper by [2021], apart from those being specified. These parameters are applied to all of the 3 models and are provided in Table. 11.

Additional parameters that we have introduced in this paper include the number of search actions n_a, the step size β when searching for the action, the uncertainty threshold α, and the number of experiences n_e. For simplicity, we opt for using a step size $\beta = 0.01$ for all environments. The parameter α is determined through the process outlined in Appendix D. These parameters are provided in Table. 12.

Discrete Action Space Environments

For discrete action space environments (Atari), we follow the hyperparameters proposed in the original paper by [2021] and apply it to all 3 models. The selected parameters are provided in Table 13.

We introduce three key parameters: the outlier action threshold γ, the number of experiences n_e, and the uncertainty threshold α. As in continuous action space environments, the uncertainty threshold α is determined using the method described in Appendix D. The action threshold γ is tuned over the range $[0.1, 0.3]$ with a step size of 0.05, and a value of 0.25 is selected for all four evaluation environments. For n_e, a value of 500 transitions is chosen, constrained by available computational resources. The selected parameters are summarized in Table 14.

G Relation to Causal Inference and Counterfactual Reasoning

Although CRDT is inspired by causal inference and counterfactual reasoning, the method did not explicitly establish a formal causal structure learning process, such as constructing a causal graph or a Structural Causal Model (SCM) [Pearl and Mackenzie, 2018]. The method is more closely related to the potential outcome framework [Neyman, 1923; Rubin, 1978] and its extension to time-varying treatments and outcomes [Robins and Hernan, 2008], which did not explicitly require a causal graph [Pearl and Mackenzie, 2018]. The proposed counterfactual reasoning process in CRDT also differs from "Pearl-style counterfactual reasoning", which requires the inference of the posterior distribution of exogenous noise variable and intervention on the parental variables. In CRDT, we assume that the noise is implicit in the dynamic model. The method, however, leverages several concepts from these frameworks.

Specifically, our method estimates the outcomes of different treatments using an Outcome Network, which aligns with prior work in adapting machine learning methods for causal effect inference [Shalit *et al.*, 2017; Jacob, 2021; Melnychuk *et al.*, 2022], where neural networks were used to estimate treatment effects by modeling counterfactual outcomes. While the potential outcome framework does not strictly require a causal graph and the choice of the underlying ML algorithm is very flexible [Jacob, 2021], in CRDT, we purposefully chose Transformers architecture for both the Treatment and Outcome networks due to their ability to capture long-term dependencies through attention mechanisms. The attention scores within the Transformer architecture underpinning these networks can serve as a simple causal masking mechanism [Pitis *et al.*, 2020; Seitzer *et al.*, 2021; Pitis *et al.*, 2022]. Furthermore, our framework assumes the three key causal assumptions, namely Consistency, Sequential

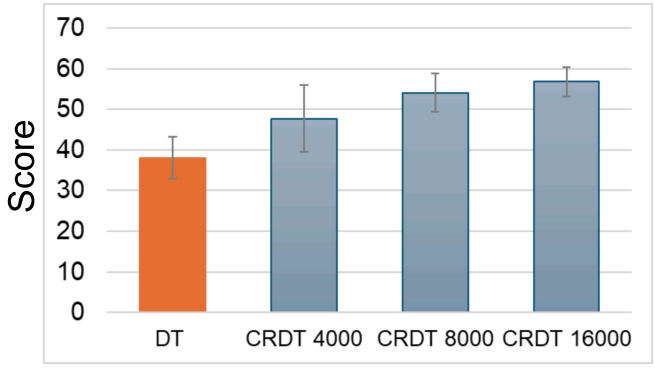

Figure F.2: The impact of varying the number of counterfactual experiences n_e in D_{CRDT} on the performance. The agent is trained using the 10% walker2d-medium-replay dataset. The terms CRDT 4000, 8000, and 16000 refer to configurations of CRDT with n_e set to 4000, 8000, and 16000 samples, respectively. We report the results over 5 seeds. For each seed, evaluation is conducted over 100 episodes.

Overlap, and Sequential Ignorability, as detailed in Appendix B. These connections demonstrate how causal inference concepts underpin our framework, even if they are not formalized in the traditional sense.

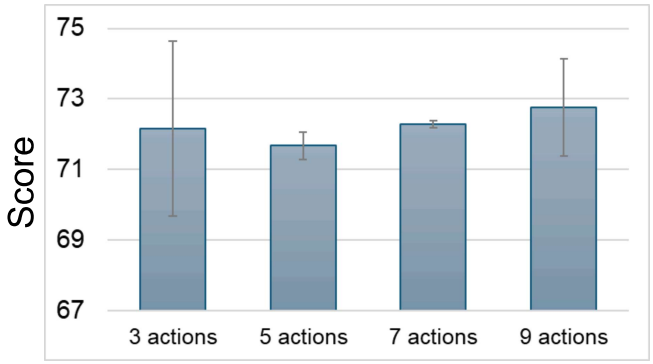

Figure F.3: The impact of varying the number of search action n_a. The agent is trained with walker2d-medium-replay dataset. We report the results over 5 seeds. For each seed, evaluation is conducted over 100 episodes.

Table 10: Performance comparison between CRDT (DT) versus CRDT (REINF) versus CRDT (EDT) on Locomotion, Ant, and Maze tasks. We report the results over 5 seeds. For each seed, evaluation is conducted over 100 episodes.

Dataset	Sequence Modeling Methods		
	CRDT (DT)	CRDT (REINF)	CRDT (EDT)
halfcheetah-med	42.8±2.32	43.0±1.51	43.1±0.36
halfcheetah-med-rep	38.0±2.54	36.8±2.01	36.0±2.21
halfcheetah-med-exp	96.4±2.32	94.4±1.74	72.3±9.19
hopper-med	67.9±1.56	74.2±6.37	54.4±7.56
hopper-med-rep	85.5±3.24	85.2±2.29	70.2±8.71
hopper-med-exp	110.3±0.14	110.3±0.63	108.7±2.92
walker2d-med	78.9±0.91	79.2±2.73	65.4±1.51
walker2d-med-rep	72.2±0.11	70.0±2.29	72.6±21.7
walker2d-med-exp	109.05±0.63	108.7±0.46	107.2±0.22
Total Locomotion	701.38±1.53	701.88±2.22	630.2±6.29
ant-med-rep	91.0±8.84	92.1±0.55	87.2±3.57
ant-med	95.84±8.32	95.2±1.13	90.2±4.60
Total Ant	186.8±8.58	187.3±0.84	177.4±4.08
maze2d-umaze	55.2±9.20	41.3±4.39	-
maze2d-large	42.3±3.74	47.7±13.6	-
Total Maze2d	97.5 ±6.47	89 ±8.99	-

Table 11: DT's Parameters for Continuous Action Space Environments.

Dataset	Batch Size	K	Learning Rate	No. Layers	Atten. Heads
maze2d-large	64	10	0.0004	5	8
maze2d-umaze	64	20	0.0001	3	8
Others	64	20	0.0001	3	1

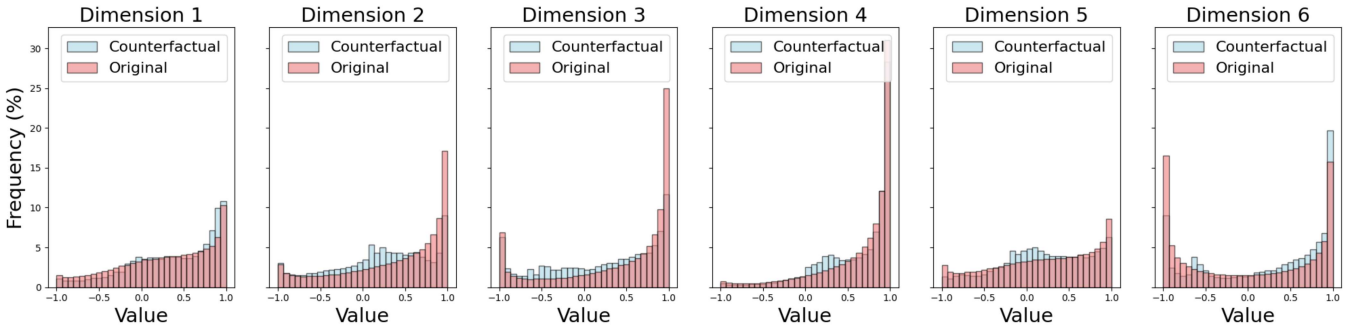

Figure F.4: Frequency distribution of action values across dimensions in the walker2d-med-rep environment. The histograms represent the percentage frequency of action values for each of the six dimensions, offering insights into the distribution patterns of actions in the dataset.

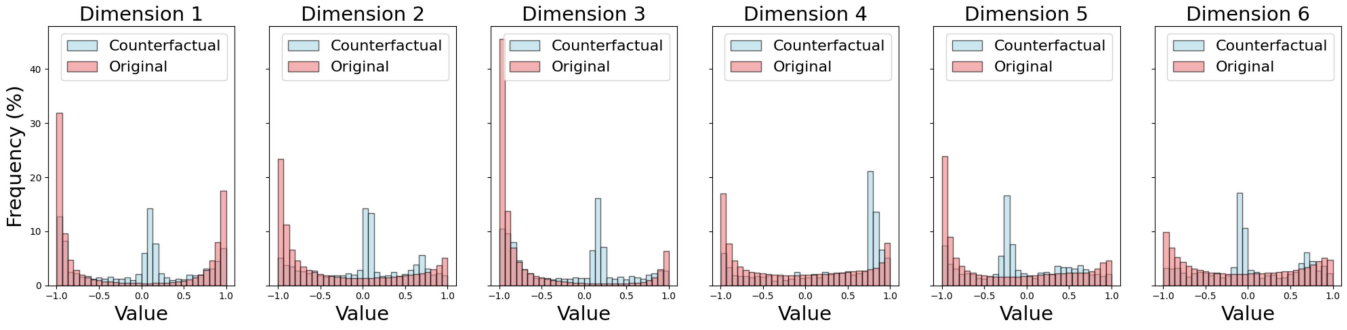

Figure F.5: Frequency distribution of action values across dimensions in the halfcheetah-med-exp environment. The histograms represent the percentage frequency of action values for each of the six dimensions, offering insights into the distribution patterns of actions in the dataset.

Table 12: New Hyperparameters for Continuous Action Space Environments.

Dataset	n_a	α	n_e
halfcheetah-med-rep	5	4.2	1000
halfcheetah-med	7	2.5	1000
halfcheetah-med-exp	5	0.3	1000
hopper-med-rep	5	0.7	1000
hopper-med	7	0.7	4000
hopper-med-exp	5	0.4	4000
walker2d-med-rep	7	1.8	4000
walker2d-med	5	1.8	1000
walker2d-med-exp	5	0.4	4000
ant-med-rep	5	0.8	4000
ant-med	5	1.5	2000
maze2d-umaze	5	0.1	2000
maze2d-large	5	0.1	2000
halfcheetah-med-rep (less_data)	5	0.1	4000
hopper-med-rep (less_data)	5	0.7	4000
walker2d-med-rep (less_data)	7	1.8	4000
maze2d-umaze (less_data)	5	0.1	4000
maze2d-large (less_data)	5	0.1	4000

Table 13: DT's Parameters for Discrete Action Space Environments.

Games	K	Learning Rate	No. Layers	Atten. Heads
Breakout, Qbert, Seaquest	30	0.0006	6	8
Pong	50	0.0006	6	8

Table 14: New Hyperparameters for Discrete Action Space Environments.

Games	α	n_e
Pong, Seaquest, Breakout	10	500
Qbert	75	500