Beyond the Known: Decision Making with Counterfactual Reasoning Decision Transformer

Minh Hoang Nguyen[1] , **Linh Le Pham Van**[1] , **Thommen George Karimpanal**[2] , **Sunil Gupta**[1] and **Hung Le**[1]

[1]Applied AI Initiative, Deakin University, Australia
[2]School of IT, Deakin University, Australia
{s223669184, l.le, thommen.karimpanalgeorge, sunil.gupta, thai.le}@deakin.edu.au

Abstract

Decision Transformers (DT) play a crucial role in modern reinforcement learning, leveraging offline datasets to achieve impressive results across various domains. However, DT requires high-quality, comprehensive data to perform optimally. In real-world applications, the lack of training data and the scarcity of optimal behaviours make training on offline datasets challenging, as suboptimal data can hinder performance. To address this, we propose the Counterfactual Reasoning Decision Transformer (CRDT), a novel framework inspired by counterfactual reasoning. CRDT enhances DT's ability to reason beyond known data by generating and utilizing counterfactual experiences, enabling improved decision-making in unseen scenarios. Experiments across Atari and D4RL benchmarks, including scenarios with limited data and altered dynamics, demonstrate that CRDT outperforms conventional DT approaches. Additionally, reasoning counterfactually allows the DT agent to obtain stitching abilities, combining suboptimal trajectories, without architectural modifications. These results highlight the potential of counterfactual reasoning to enhance reinforcement learning agents' performance and generalization capabilities.

1 Introduction

In the pursuit of achieving artificial general intelligence (AGI), reinforcement learning (RL) has been a widely adopted approach. Conventional RL methods have shown impressive success in training AI agents to perform tasks across various domains, such as gaming [Silver *et al.*, 2017] and robotics [van Hoof *et al.*, 2015; Le Pham Van *et al.*, 2024]. When referring to conventional RL approaches, we mean methods that train agents to discover an optimal policy that maximizes returns, either through value function estimation or policy gradient derivation [Sutton and Barto, 2018]. However, recent advances, such as Decision Transformers (DT) [Chen *et al.*, 2021], introduce a paradigm shift by leveraging supervised learning (SL) on offline RL datasets, offering a more practical and scalable alternative to the online learning traditionally required in RL. This shift highlights the growing importance of SL on offline RL, which can be efficient in environments where data collection is expensive and impractical [Chen *et al.*, 2021].

In its original form, the DT agent is trained to maximize the likelihood of actions conditioned on past experiences [Chen *et al.*, 2021]. Numerous follow-up studies have tried to improve DT, such as through online fine-tuning [Zheng *et al.*, 2022], pre-training [Xie *et al.*, 2023], or improving its stitching capabilities [Wu *et al.*, 2023; Zhuang *et al.*, 2024]. These works have shown that DT techniques can match or even outperform state-of-the-art conventional RL approaches on certain tasks. However, these improvements focus solely on maximizing the use of available data, raising the question: What if the optimal data is underrepresented in the given dataset? This scenario is illustrated in Fig. 1 of a toy navigation environment, wherein the good (blue) trajectories are underrepresented compared to the bad (green) trajectories. DT is expected to underperform in this environment because it simply maximizes the likelihood of the training data, which can be problematic when optimal data is lacking. Additionally, it lacks effective stitching capabilities—the ability to combine suboptimal trajectories. This leads us to a key question: *Can we improve DT's performance by enabling the agent to reason about what lies beyond the known?*

Our Counterfactual Reasoning Decision Transformer (CRDT) approach is inspired by the potential outcome framework, specifically, the ability to reason counterfactually [Neyman, 1923; Rubin, 1978]. The core idea behind CRDT is that by reasoning about hypothetical—imagining better outcomes that didn't happen— the agent must evaluate how alternative actions could have influenced outcomes. This process reveals causal relationships between states, actions, and rewards, enhancing its understanding and improving generalization. This mirrors how humans imagine alternative scenarios from past experiences to inform better decisions in the future.

The CRDT framework has three key steps. The first step involves training the agent to reason counterfactually. We introduce two models: the Treatment model $\mathcal{T}$ and the Outcome model $\mathcal{O}$. The model $\mathcal{T}$ is trained to estimate the conditional distribution of actions given the historical experiences, i.e., the probability of selecting actions based on past trajectories. This differs from the original DT, which directly predicts the action itself rather than modeling the underly-

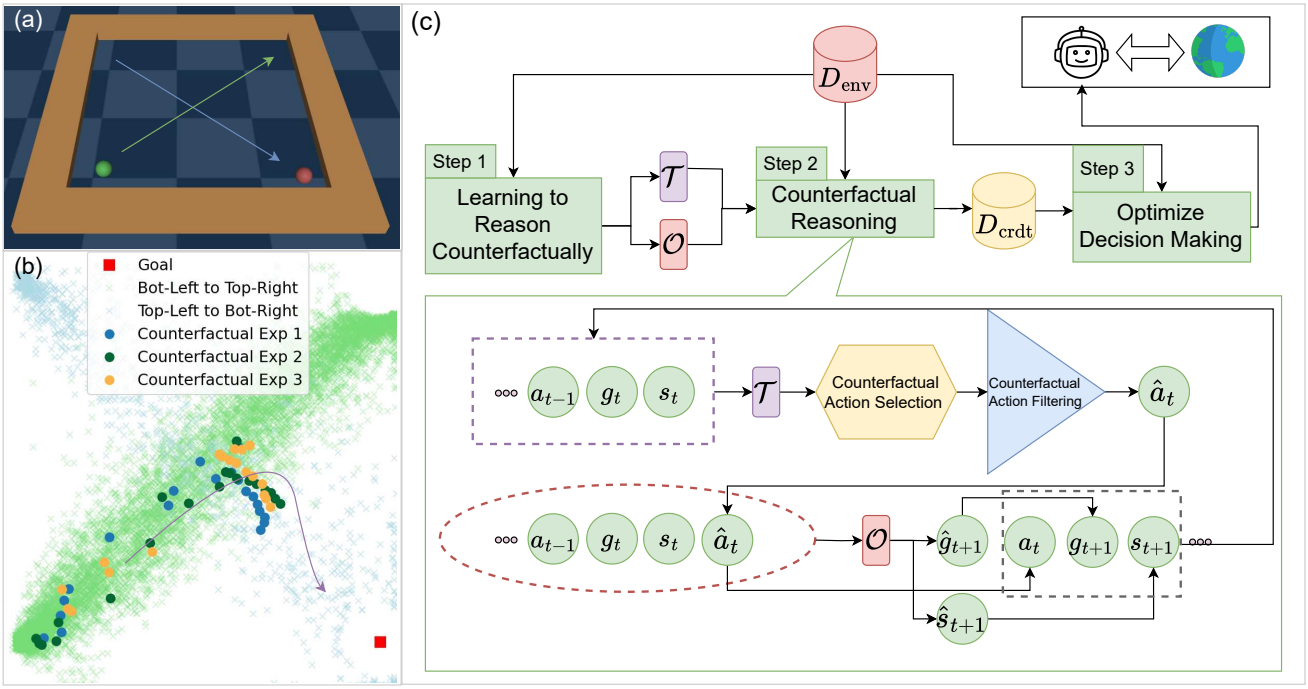

Figure 1: **(a)**: A toy environment where the goal of the agent is to move from the green circle position to the red circle position given that data is biased toward moving from bottom-left to top-right (green trajectory/diagonal line) over top-left to bottom-right (blue trajectory/ diagonal line). When using traditional DT, the agent will most likely follow the green trajectory and fail to reach the goal. **(b)**: The empirical result of the counterfactual reasoning process following CRDT on the toy environment, with the green and blue trajectories forming an intersection. At the intersection, notice that turning right yields a higher potential outcome/return, CRDT generates counterfactual experience accordingly. As shown by the bold yellow, blue, and green dots, none of the counterfactual experiences followed the green trajectory after the crossing point; they all show a clear right turn. Training DT with these counterfactual experiences improved the overall performance (refer to Sect. 5.1 for performance results). **(c) Top**: The CRDT framework follows three steps: first, learning to reason counterfactually with the CRDT agent; second, perform counterfactual reasoning to generate counterfactual experiences; and third, use these experiences to improve decision-making. **Bottom**: A single step in the iterative counterfactual reasoning process of a trajectory. The outcomes of one-step reasoning are the counterfactual action $\hat{a}_t$, the next state $\hat{s}_{t+1}$ and returns-to-go $\hat{g}_{t+1}$ will replace the original values a_t, s_{t+1}, g_{t+1} and the generated data will be used in next iteration.

ing distribution. The model $\mathcal{O}$ is trained to predict the future state and return as outcomes of taking an action. Once these two models are trained using the given offline dataset, we proceed to the second step. We aim to utilize the action selection probabilities and the inferred outcomes to generate counterfactual experiences. Unlike prior approaches that generate counterfactual data simply by perturbing the actions or states [Pitis *et al.*, 2022; Sun *et al.*, 2023; Zhao *et al.*, 2024; Sun *et al.*, 2024] with small noise, we argue that an action should be considered as counterfactual if only it has a low probability of being selected. We employ a mechanism known as *Counterfactual Action Selection* mechanism to identify such actions. However, extreme counterfactual actions may introduce excessive noise or lead to states that are not beneficial for the agent's learning. To mitigate this, we implement a mechanism called *Counterfactual Action Filtering* to eliminate irrelevant actions. The actions that pass the filtering process will be used as inputs for the Outcome model, which predicts the outcomes of these actions. In the final step, we integrate these counterfactual experiences with the offline dataset to train the underlying DT agent. Fig. 1(c) provides an overview of our CRDT framework.[1]

Our experiments in continuous action space environments (Locomotion, Ant, Maze2d benchmarks) and discrete action space environments (Atari games), show that our framework improves the performance of the underlying DT agent. An interesting side effect of CRDT is that the DT agent acquires the "stitching" ability without requiring any modifications to the underlying architecture. Thus, our key contributions are:

1. We propose the CRDT framework, a novel Decision Transformer architecture that enables agents to reason counterfactually, allowing them to explore alternative outcomes and generalize to novel scenarios.
2. Through extensive experiments, we demonstrate that CRDT consistently enhances the performance of the underlying DT agent in standard, smaller datasets, and modified environment settings that require robust generalization. It also provides DT with the stitching ability.

[1]Source code: https://github.com/mhngu23/Beyond-the-Known-Decision-Making-with-Counterfactual1-Reasoning-Decision-Transformer

2 Related Work

2.1 Offline Reinforcement Learning and Sequence Modeling

Offline RL [Levine *et al.*, 2020] refers to the task of learning policies from a static dataset D_{env} of pre-collected trajectories. Traditional methods used to solve offline RL can be classified into model-free offline RL and model-based RL approaches. Model-free methods aim to constrain the learned policy close to the behaviour policy [Levine *et al.*, 2020], through techniques such as learning conservative Q-values [Xie *et al.*, 2021; Kostrikov *et al.*, 2021a], applying uncertainty quantification to the predicted Q-values [Agarwal *et al.*, 2020; Levine *et al.*, 2020]. Model-based methods [Yu *et al.*, 2020; Kidambi *et al.*, 2020], involve learning the dynamic model of the environment, then, generating rollouts from the model to optimize the policy. Our method is more aligned with model-based, as we use a model to generate samples. The difference is that we only sample low-distribution action.

Before DT, upside-down reinforcement learning [Schmidhuber, 2019] applied supervised learning techniques to address RL tasks. In 2021, Chen [2021] introduced DT and the concept of incorporating returns into the sequential modeling process to predict optimal actions. Inspired by both DT, numerous methods have since been proposed to enhance performance, focusing on areas such as architecture [Kim *et al.*, 2023], pretraining [Xie *et al.*, 2023], online finetuning [Zheng *et al.*, 2022], dynamic programming [Yamagata *et al.*, 2023], and trajectory stitching [Wu *et al.*, 2023; Zhuang *et al.*, 2024]. To our knowledge, no work has integrated counterfactual reasoning with DT.

2.2 Counterfactual Reasoning in Conventional Reinforcement Learning

Several methods have explored the application of counterfactual reasoning in RL [Pitis *et al.*, 2020; Pitis *et al.*, 2022; Killian *et al.*, 2022] and imitation learning (IL) [Sun *et al.*, 2023]. These methods are not directly comparable to CRDT as they rely on a predefined or the learning of a structure causal model (SCM) [Pearl and Mackenzie, 2018]. In contrast, our approach is rooted in the PO framework [Rubin, 1978; Robins and Hernan, 2008], which focuses on estimating the effects without the need for a specified SCM. Avoiding learning the SCM reduces computational costs. Our approach aligns more closely with works that estimate counterfactual outcomes for treatments in sequential data [Melnychuk *et al.*, 2022; Frauen *et al.*, 2023; Wang *et al.*, 2018; Li *et al.*, 2020]. The main contribution of our work lies in integrating these estimated outcomes to enhance the underlying DT agent (refer to Appendix G for the relation of CRDT to causal inference).

3 Preliminaries

3.1 Offline Reinforcement Learning and Decision Transformer

We consider learning in a Markov decision process (MDP) represented by the tuple $(S, A, r, P, \gamma, \rho_0)$, where S is the state space, A is the action space, reward function $r : S \times A \to \mathbb{R}$, γ is the discount factor, and the initial distribution ρ_0. At each timestep t, the agent observes a state $s_t \in S$, takes an action $a_t \in A$ and receives a reward $r_t = R(s_t, a_t)$. The transition to the next state $s_{t+1} \in S$ follows the probability transition function $P(s_{t+1} \mid s_t, a_t)$. The goal of reinforcement learning is to find a policy $\pi(a|s)$ that can maximize the expected return $\mathbb{E}_{\pi,P,\rho_0}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)\right]$.

In **offline RL**, the agent is not allowed to interact with the environment until test time [Levine *et al.*, 2020]. Instead, it is given a static dataset $\mathcal{D}_{\text{env}} = \{(s_0^{(i)}, a_0^{(i)}, r_0^{(i)}, s_1^{(i)}, \ldots, s_t^{(i)}, a_t^{(i)}, r_t^{(i)}, \ldots)\}_{i=1}^{N}$, collected from one or more behaviour policies π_β, to learn from. Generally, learning the optimal policy from a static dataset is challenging or even impossible [Kidambi *et al.*, 2020]. Consequently, the objective is to create algorithms that reduce suboptimality to the greatest extent possible.

DT [Chen *et al.*, 2021] is a pioneering work that frames RL as a sequential modeling problem. The authors introduce a transformer-based agent, denoted as $\mathcal{M}$ with trainable parameters δ, to tackle offline RL environments. While substantial research has built upon this work (see Sect. 2 for a comprehensive review), DT, in its original form, applies minimal modifications to the underlying transformer architecture [Vaswani *et al.*, 2017]. Similar to traditional offline RL approaches, the agent $\mathcal{M}$ in DT is given an offline dataset $\mathcal{D}_{\text{env}}$, which contains multiple trajectories. Each trajectory consists of sequences of states, actions, and rewards. However, rather than simply using past rewards from $\mathcal{D}_{\text{env}}$ as input into $\mathcal{M}$, the authors introduce returns-to-go, denoted as g_t and computed as $g_t = \sum_{t'=t}^{T} r_{t'}$. The agent $\mathcal{M}$ is fed this returns-to-go g_t instead of the immediate reward r_t, allowing it to predict actions based on future desired returns. In Chen [2021], a trajectory $\tau^{(i)}$ is represented as: $\tau^{(i)} = (g_1^{(i)}, s_1^{(i)}, a_1^{(i)}, \ldots, g_T^{(i)}, s_T^{(i)}, a_T^{(i)})$. Agent $\mathcal{M}$ with parameter δ is trained on a next action prediction task. This involves using the experience $h_t = (g_1, s_1, a_1, ..., g_t, s_t, a_t)$, returns-to-go g_{t+1} and state s_{t+1} as inputs and the next action a_{t+1} as output. This can be formalized as:

$$p(a_{t+1} \mid h_t, s_{t+1}, g_{t+1}; \delta) = \mathcal{M}(h_t, s_{t+1}, g_{t+1}; \delta), \quad (1)$$

for discrete action space. And:

$$a_{t+1} = \mathcal{M}(h_t, s_{t+1}, g_{t+1}; \delta), \quad (2)$$

for continuous action space. This action prediction ability is then utilized during the inference and evaluation phases on downstream RL tasks. In addition to the aforementioned process, the authors investigated the potential benefits of integrating additional tasks to predict the next state and returns-to-go into the agent's training to enhance its understanding of the environment's structure, however, it was concluded that such methods do not improve the agent's performance [Chen *et al.*, 2021]. Further, they suggested that this "would be an interesting study for future research" [Chen *et al.*, 2021]. Our method, while not explicitly incorporating such predictions, demonstrates an alternative approach that can effectively use these predictions to improve the agent's performance.

3.2 Potential Outcome and Counterfactual Reasoning

Our work is inspired by the potential outcomes (PO) framework [Neyman, 1923; Rubin, 1978] and its extension to time-varying treatments and outcomes [Robins and Hernan, 2008]. The PO framework estimates causal effects by considering the outcomes for each variable under different treatments [Robins and Hernan, 2008]. Counterfactual reasoning involves imagining what might have happened under alternative conditions that did not occur [Pearl and Mackenzie, 2018]. Under the potential outcome framework, at each timestep $t \in \{1, ..., T\}$, we observe time-varying covariates X_t, treatments A_t, and the outcomes Y_{t+1}. The treatment A_t influences the outcome Y_{t+1}, and all X_t, A_t, and Y_{t+1} affect future treatment. A history at timestep t is denoted as $\bar{H}_t = \{\bar{X}_t, \bar{A}_{t-1}, \bar{Y}_t\}$, where $\bar{X}_t = (X_1, \ldots, X_t)$, $\bar{Y}_t = (Y_1, \ldots, Y_t)$, and $\bar{A}_{t-1} = (A_1, \ldots, A_{t-1})$. The estimated potential outcome for a trajectory of treatment $\bar{a}_t = (a_t, ..., a_{t+\xi-1})$ is expressed as $\mathbb{E}[Y_{t+\xi}(\bar{a}_{t:t+\xi-1}) \mid \bar{H}_t]$ where $\xi \geq 1$ is the treatment horizon for ξ steps prediction.

Mapping to this paper, the time-varying covariates correspond to the agent's past observations and the returns-to-go it has received. The treatment corresponds to the action taken, and the outcome is the subsequent observation and returns. A counterfactual treatment refers to an action $\hat{a}_t$ the agent could have taken but did not. For each timestep t, we aim to estimate the outcome of counterfactual action $\hat{a}_t$ or $\mathbb{E}[\hat{s}_{t+1}, \hat{g}_{t+1} \mid \hat{h}_t]$, where $\hat{s}_{t+1}$ and $\hat{g}_{t+1}$ denote the counterfactual state and returns-to-go corresponding to taking $\hat{a}_t$. $\hat{h}_t$ is the new historical experience $(g_1, s_1, a_1, \ldots, g_t, s_t, \hat{a}_t)$, given that we have taken an action $\hat{a}_t$ that is different from the original action a_t in D_{env}.

Our framework satisfies the three key assumptions—(1) consistency, (2) sequential ignorability, and (3) sequential overlap—ensuring the identifiability of counterfactual outcomes from the factual observational data $\mathcal{D}_{\text{env}}$. In the context of reinforcement learning, the assumption of consistency implies that for any given action a_t, the observed next state s_{t+1} and returns-to-go g_{t+1} accurately represent the true outcome of the action. This assumption holds since $\mathcal{D}_{\text{env}}$ is collected from behaviour policies π_β trained within the same environment, ensuring that the data faithfully captures the environment's true dynamics. The assumption of sequential overlap states that for any observed history h_t, every action a_t has a non-zero probability of being selected. If the behavior policy π_β used to collect the data explores a diverse range of actions across different histories, this assumption is likely to hold. Furthermore, sequential ignorability implies that the history h_t contains all relevant information that influences the agent's actions and future outcomes. This assumption rests on the premise that the dataset sufficiently captures the key factors affecting the treatments and their resulting outcomes. In prior works, these assumptions have been used in both environments with discrete or continuous treatments [Melnychuk *et al.*, 2022; Bahadori *et al.*, 2022; Frauen *et al.*, 2023].

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 PO framework 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.

4 Methodology

This section introduces the Counterfactual Reasoning Decision Transformer framework, our approach to empowering the DT agent with counterfactual reasoning capability.[2] The framework follows three steps: first, we train the Treatment and Outcome Networks to reason counterfactually; then, we use these two networks to generate counterfactual experiences and add these to a buffer D_{crdt}; and finally, we train the underlying agent with these new experiences.

4.1 Learning to Reason Counterfactually

As mentioned in Sect. 3.2, counterfactual reasoning involves estimating how outcomes would differ under unobserved treatments [Pearl and Mackenzie, 2018]. This process is often broken down into learning the selection probability of the agent's treatment and learning the outcomes of the treatments. This means that we must be able to estimate the probability of selecting actions a_t, at timestep t, given historical experiences $h_{t-1} = (g_1, s_1, a_1, \ldots, g_{t-1}, s_{t-1}, a_{t-1})$, the current outcome state s_t, and returns-to-go g_t. Knowing the distribution enables exploration of counterfactual actions $\hat{a}_t$ (actions with low selection probability). By using these counterfactual actions as new treatment, we can estimate their corresponding counterfactual outcomes, the next state $\hat{s}_{t+1}$ and the next returns-to-go $\hat{g}_{t+1}$. To address these steps, we introduce two separate transformer models: the Treatment model ($\mathcal{T}$) and the Outcome model ($\mathcal{O}$). The model $\mathcal{T}$, parameterized by θ, learns the probability of selecting treatments (i.e., the agent's action). The model $\mathcal{O}$, with parameters η, estimates the outcomes of actions. Together, these models enable the agent to reason counterfactually, by learning the probability of selecting actions and the potential outcomes of unchosen actions.

Treatment Model Training. We want to use the model $\mathcal{T}$ to estimate the probability of selecting a specific action. In discrete action space environment, this can be formalized as:

$$p(a_t \mid h_{t-1}, s_t, g_t; \theta) = \mathcal{T}(h_{t-1}, s_t, g_t; \theta). \tag{3}$$

The model can be trained using a cross-entropy (CE) loss:

$$\mathcal{L}_{\mathcal{T}(\theta)} = -\frac{1}{N} \sum_{i=1}^{N} a_t^{*(i)} \log \left(p(a_t^{(i)} \mid h_{t-1}^{(i)}, s_t^{(i)}, g_t^{(i)}; \delta) \right), \tag{4}$$

[2] From this point forward, we will use the notations a_t^*, s_t^*, g_t^* for the factual values and notations a_t, s_t, g_t for the predicted values. $\hat{a}_t, \hat{s}_t, \hat{g}_t$ will be used to denote counterfactual related values.

where $a_t^{*(i)}$ is the encoded true label for action of the i-th instance of N samples, and $p(a_t^{(i)} \mid h_{t-1}^{(i)}, s_t^{(i)}, g_t^{(i)}; \delta)$ is the predicted probability of action $a_t^{(i)}$. For continuous action space, following PO practices [Zhu *et al.*, 2015; Bahadori *et al.*, 2022], we assume that actions follow a Gaussian distribution and estimate its mean and variance using a neural network, thus, $a_t \sim \mathcal{N}(\mu_t, \sigma_t^2)$, where $\mu_t, \sigma_t^2 = \mathcal{T}(h_{t-1}, s_t, g_t; \theta)$. Model $\mathcal{T}$ is trained to minimize:

$$\mathcal{L}_{\mathcal{T}(\theta)} = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{(a_t^{*(i)} - \mu_t^{(i)})^2}{2\sigma_t^{2(i)}} + \frac{1}{2}\log(2\pi\sigma_t^{2(i)})\right). \quad (5)$$

Outcome Model Training. To predict outcome of taking an action, the $\mathcal{O}$ model is trained to minimize the loss between predicted state s_{t+1} and returns-to-go g_{t+1} and their factual values. This objective can be achieved using the Mean Squared Error (MSE) loss.

$$\mathcal{L}_{\mathcal{O}(\eta)} = \frac{1}{N}\sum_{i=1}^{N}\left(\|s_{t+1}^{*(i)} - s_{t+1}^{(i)}\|^2 + \|g_{t+1}^{*(i)} - g_{t+1}^{(i)}\|^2\right). \quad (6)$$

Here, $s_{t+1}, g_{t+1} = \mathcal{O}(h_t; \eta)$ with input trajectory $h_t = (g_1, s_1, a_1, ..., g_t, s_t, a_t)$.

4.2 Counterfactual Reasoning with CRDT

This section describes the agent's iterative counterfactual reasoning process using model $\mathcal{T}$ and $\mathcal{O}$. At each timestep t, model $\mathcal{T}$ is provided with (h_{t-1}, s_t, g_t) to compute the action distribution. Using this distribution, a counterfactual action $\hat{a}_t$ is drawn according to the Counterfactual Action Selection (See sub-section below). Next, model $\mathcal{O}$ is used to generate the counterfactual state $\hat{s}_{t+1}$ and returns-to-go $\hat{g}_{t+1}$. The trajectory is then updated with the counterfactual experience, forming new input $(h_{t-1}, s_t, g_t, \hat{a}_t, \hat{s}_{t+1}, \hat{g}_{t+1})$ for the next iteration. Counterfactual reasoning for a trajectory is deemed successful if the iterative process proceeds to the end of the trajectory without violating the Counterfactual Action Filtering mechanism (See sub-section below). Successful reasoning trajectories are added to counterfactual experience buffer, denoted as D_{crdt}, if the number of experiences in D_{crdt} is less than a hyperparameter n_e.

Counterfactual Action Selection. Our goal is to sample n_a actions that can be classified as counterfactual actions, which are passed to the filtering process. Rather than just adding small noise, we aim to identify counterfactual actions as outliers, thereby, encouraging the exploration of less supported outcomes. In a discrete action space, as the output of the Treatment model is the probability of the action, we can simply select all actions whose probability of being selected is less than a threshold γ. For continuous action spaces, we draw inspiration from the maximum of Gaussian random variables, as discussed in Kamath [2015], to derive our bound to identify counterfactual actions. The upper bound of the expectation of the maximum of Gaussian random variables is used. Applying to action a_t, this is written as:

$$\mathbb{E}\left[\max(a_t)\right] \leq \mu_t + \sqrt{2}\sigma_t\sqrt{\ln(n_{enc})}. \quad (7)$$

Here, n_{enc} denotes the number of times the model has encountered an input (h_t, s_{t+1}, g_{t+1}). This bound indicates the expected range for the action, and any action that exceeds this bound is considered a counterfactual action. Based on this, we derive the formula to search for potential actions in the counterfactual action set:

$$a_t^{(j)} = \mu_t - \Phi^{-1}\left(0.08 - j \cdot \beta\right)\sigma_t\sqrt{\ln(n_{enc})}, \quad \text{for} \quad j = 0, 1, \ldots, n_a, \quad (8)$$

where β is the step size and j indicates the index of the j-th action from the total n_a sampled actions. Φ^{-1} is the quantile function of the standard normal distribution. When $j = 0$, $\Phi^{-1}\left(0.08 - j \cdot \beta\right) = \Phi^{-1}\left(0.08\right) \approx -\sqrt{2}$, thus, Eq. 8 is approximately equal to the RHS of Eq. 7. By using Eq. 8, we ensure that at each time step t, we can explore a diverse range of candidate counterfactual actions.

Counterfactual Action Filtering. This mechanism is proposed to filter counterfactual actions that are not beneficial to the agent. For each candidate action, we generate subsequent outcomes using $\mathcal{O}$ to construct candidate counterfactual trajectories. The trajectories are then filtered based on 2 criteria: (1) high accumulated return and (2) high prediction confidence. The motivation behind sampling high return actions is because DT improves with higher return data [Bhargava *et al.*, 2024; Zhao *et al.*, 2024], aligning with our approach to introduce counterfactual experiences that can lead to better outcomes. Therefore, we look for actions that resulted in the lowest counterfactual returns-to-go **(equivalent to higher return)**, $\hat{g}_{t+1}$, lower than returns-to-go g_{t+1} in D_{env}.

Regarding the second criterion, we introduce an uncertainty estimator function to determine low prediction confidence states and exclude actions that lead to these states, stopping and discarding the counterfactual trajectory if the uncertainty is too high. In our framework, the model $\mathcal{O}$ is trained with dropout regularization layers. This allows us to run multiple forward passes through the model, with the dropout layer activated, to check the uncertainty of the output state. The output of m forward passes, at timestep t, is the matrix of state predictions, $\mathbf{S}_{t+1} = \begin{bmatrix} s_{t+1}^{(1)} & s_{t+1}^{(2)} & \cdots & s_{t+1}^{(m)} \end{bmatrix}$. $\mathbf{S}_{t+1} \in \mathbb{R}^{m \times d}$, where d is the state dimension. We denote $\text{Var}(\mathbf{S}_k) \in \mathbb{R}$, where k is a timestep, as the function that calculates the maximum variance across all dimensions j' of s_k, where $j' = 1, 2, \ldots, d$. This can be obtained from the covariance matrix of $\mathbf{S}_k$ (see derivation in Appendix D). The maximum variance across all dimensions is used as the variance of the predictions and the uncertainty value. Our uncertainty filtering mechanism, checking the accumulated maximum variance, can be written as:

$$U^{\alpha}(\mathbf{S}_{t+1}) = \begin{cases} \textbf{TRUE} \text{ (Unfamiliar)}, & \text{if } \sum_{k=t_0}^{t+1}\text{Var}(\mathbf{S}_k) > \alpha, \\ \textbf{FALSE} \text{ (Familiar)}, & \text{otherwise.} \end{cases} \quad (9)$$

Here, $\sum_{k=t_0}^{t+1}\left(\text{Var}(\mathbf{S}_k)\right)$ is the accumulated maximum variances of state prediction from a timestep t_0 that we start the reasoning process to current timestep $t+1$. The function $U^{\alpha}(\mathbf{S}_{t+1})$ returns **TRUE** if the state s_{t+1} is unfamiliar. If the

uncertainty is low, we will run a final forward pass through the model, with the dropout layer deactivated, to get the deterministic state and returns-to-go output. This helps avoid noisy trajectories and benefits counterfactual reasoning.

4.3 Optimizing Decision-Making with Counterfactual Experience

In this section, we describe how our counterfactual reasoning capability has been applied to improve the agent's decision-making. To demonstrate the effectiveness, we have selected the original DT model introduced by Chen [2021] as the main backbone for the experiment. The learning agent in this paper, denoted as $\mathcal{M}$, is trained following Eq. 1 to minimize either CE loss for discrete action space environments or Eq. 2 with MSE loss for continuous action space environments. For discrete action space, the loss function is defined as:

$$\mathcal{L}_{\mathcal{M}(\delta)} = -\frac{1}{N}\sum_{i=1}^{N} a_{t+1}^{*(i)} \log\left(p(a_{t+1}^{(i)} \mid h_t^{(i)}, s_{t+1}^{(i)}, g_{t+1}^{(i)}; \delta)\right), \tag{10}$$

where $a_{t+1}^{*(i)}$ is encoded true label for the action of the i-th instance of N samples, and $p(a_{t+1}^{(i)} \mid h_t^{(i)}, s_{t+1}^{(i)}, g_{t+1}^{(i)}; \delta)$ is the probability outputed from the model. For continuous actions, the loss is:

$$\mathcal{L}_{\mathcal{M}(\delta)} = \frac{1}{N}\sum_{i=1}^{N}\left(\|a_{t+1}^{*(i)} - a_{t+1}^{(i)}\|^2\right). \tag{11}$$

At each training step, we sample equal batches of trajectories from both the environment dataset D_{env} and the counterfactual experience buffer D_{crdt}. The agent $\mathcal{M}$ is trained on both data sources, with the total loss calculated as the combination of the two losses $\mathcal{L}_{\mathcal{M}(\delta)} = \mathcal{L}^{\text{env}}_{\mathcal{M}(\delta)} + \mathcal{L}^{\text{crdt}}_{\mathcal{M}(\delta)}$.

5 Experiments

We conducted experiments on both continuous action space environments (Locomotion, Ant, and Maze2d [Fu *et al.*, 2020]) and discrete action space environments (Atari [Bellemare *et al.*, 2013]) to address several key research questions.

We compare against several baselines, including sequential modeling techniques and conventional RL. Sequential modeling baselines include simple DT [Chen *et al.*, 2021], which also serves as the backbone, EDT [Wu *et al.*, 2023] and state-of-the-art (SOTA) approach Reinformer (REINF) [Zhuang *et al.*, 2024]. Conventional methods include Behavior Cloning (BC) [Pomerleau, 1988], model-free offline methods, such as CQL [Kumar *et al.*, 2020] and IQL [Kostrikov *et al.*, 2021b] and model-based offline methods, such as MOPO [Yu *et al.*, 2020] and MOReL [Kidambi *et al.*, 2020].

Does CRDT enhance the DT in continuous action space environments? In Table 1, we summarize the total scores of experimental results for CRDT and baseline methods on the Locomotion and Ant tasks from the D4RL dataset. The D4RL dataset serves as a standard benchmark for offline RL, with Locomotion comprising three environments (*walker2d*, *hopper*, and *halfcheetah*) and Ant consisting of a single environment (*ant*). For the Locomotion tasks, we evaluate performance using three D_{env} dataset: *medium-replay*, *medium*, and *medium-expert*, while for the Ant task, we use two: *medium-replay* and *medium*. CRDT consistently improves upon the simple backbone DT model across all datasets, achieving an average performance gain of 3.5% on Locomotion tasks and 2.7% on the Ant task. Notably, the largest improvement is observed on the walker2d-med-rep dataset, with a significant 16.1% increase. Overall, CRDT emerges as the best-performing method on average, outperforming other approaches on Locomotion tasks and achieving performance comparable to the state-of-the-art RL method, IQL, and the sequential modeling method, REINF, on the Ant task.

Can CRDT improve DT's performances given limited training dataset? To evaluate the generalizability improvements of CRDT over DT, we conducted experiments using only a limited subset of the D_{env} dataset. The experiments were carried out on Locomotion and Maze2d (more challenging environments as they required the ability to stitch suboptimal trajectories [Zhuang *et al.*, 2024]). We compared CRDT's performance against the backbone DT model and REINF, the second-best DT method according to Table 1. The results are in Fig. 2. Our method experiences the smallest performance degradation in this setting. In *hopper* and *halfcheetah* environments, while all three methods exhibit similar performance at 100% dataset size, our method demonstrates only about a 15% drop when trained on 10% of the dataset. In contrast, both REINF and DT degrade by over 21%, with extreme cases of 40%. On the Maze2d tasks, CRDT performances drop approximately 25% on umaze and 3% on large dataset. While DT cannot learn these environments (drop more than 90%) and REINF performance drops approximately 45%.

Does CRDT enhance the DT in discrete action space environments? We conducted experiments on Atari games (Breakout, Qbert, Pong, and Seaquest), which feature discrete action spaces and more complex observation spaces. The normalized scores are shown in Table 2. Given the increased difficulty of the observation space, we expected that CRDT might not always outperform DT, as it could introduce noise, even with mechanisms in place to prevent noise accumulation. Nevertheless, CRDT improved DT in 3 out of the 4 games (highest improvement of 25% on Breakout). We believe that for these complex environments, a larger neural network could lead to greater performance gains.

5.1 Model Analysis and Ablation Study

Comparing CRDT with Varying Action Selection Methods. We conduct an ablation study on the two mechanisms that define our method: Counterfactual Action Filtering and Counterfactual Action Selection. In Table 3, we compare the performance of the full CRDT against several variations: the version that does not compare the returns-to-go (W/o comparing g), the version that does not utilize the uncertainty quantifier $U^{\alpha}(\mathbf{S}_k)$, the variation that simply samples an action a without considering whether a is low distribution, and the variation that samples an action $a + \epsilon$, where ϵ is random Gaussian noise sampled from the range [0.01, 0.05]. The results show that simply adding data will still improve

Dataset	Traditional Methods					Sequence Modeling Methods			
	BC	CQL	IQL	MOPO	MOReL	DT	EDT	REINF	CRDT (Ours)
Locomotion	466.7	*698.5*	692.6	378.0	656.5	677.0	621.4	698.0	**701.38±1.5**
Ant	-	-	*186*	-	-	181.5	175.7	184.3	**186.86±8.5**

Table 1: Performance comparison on Locomotion (9 tasks) and Ant (2 tasks). We report the total score across all environments within each category. Results are averaged over 5 seeds, with evaluation conducted over 100 episodes per seed. The best result is highlighted in **bold**, and the second-best in *italic*.

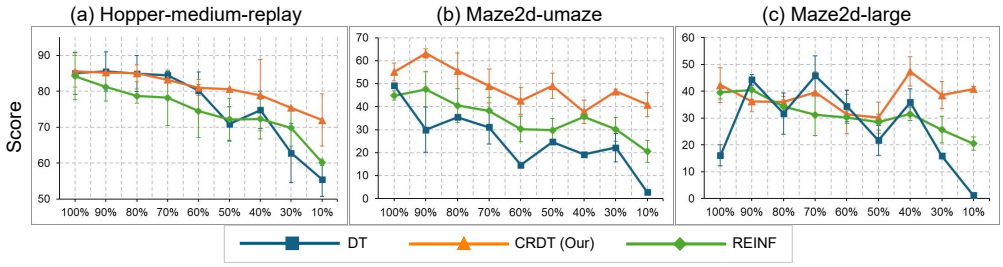

Figure 2: Performance comparison on limited subset of D_{env}. The results are over 5 seeds. For each seed, evaluation is conducted over 100 episodes. The X-axis represents the percentage of the dataset used in the experiment.

Game	BC	DT	CRDT (Ours)
Breakout	138.9±54.6	198.6±1.8	**248.9±58.9**♠
Qbert	**17.4±13.4**	7.2±0.2	7.5±0.6♠
Pong	85.2±78.3	**140.2±63.6**	102.2±67.6
Seaquest	2.1±0.2	5.7±6.3	**7.4±0.5**♠
Average	60.9±36.6	87.9±17.9	**91.5±31.9**♠

Table 2: Performance comparison on Atari games (1% DQN-replay dataset). We report the human-normalized scores 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.

Variations	Score
DT	62.1±2.2
W/o comparing g	67.4±2.1
W/o $U^{\alpha}(\mathbf{S}_k)$	69.6±2.8
a	68.4±3.45
a + noise ϵ	69.3±4.4
CRDT (Ours)	**72.3±0.1**

Table 3: Performance comparison with different action selection methods on walker2d-med-rep.

the performance DT, however, the improvement is less significant than when CRDT is used. Full CRDT improves the performance by 16%, while the closet variations, do not utilize $U^{\alpha}(\mathbf{S}_k)$, achieving only 12.0%.

Can CRDT enable DT to stitch trajectories? Table 4 presents results of the experiment conducted in environment in Fig. 1. In this environment, all states, apart from the goal, receive a reward of 0. Reaching the goal state receives a reward of +1. We expect that, if traditional DT is used, the agent would struggle to learn this environment due to the lack of

Dataset Ratio	DT	CRDT (Ours)
10:1	0.37±0.30	**0.83±0.14**
20:1	0.41±0.36	**0.90±0.07**
50:1	0.39±0.18	**0.92±0.15**

Table 4: Performance comparison on the toy environment in Fig. 1. The dataset ratio is between the number of bad (green) trajectories versus good (blue) trajectories.

stitching ability. The results support this, showing that the traditional DT achieves only around a 40% success rate, whereas our CRDT approach achieves nearly 90%. Although our approach is not specifically designed to achieve stitching ability during training, as seen in Wu [2023] and Zhuang [2024], our agent interestingly acquires this ability. This occurs because the generated training data is effectively stitched through the ongoing process of seeking higher returns. This also explains the performance in Ant in Table 1 and Maze2d in Fig. 2, both of which require stitching.

6 Discussion

We present the CRDT framework, which integrates counterfactual reasoning with DT. Our experiments show that CRDT improves DT and its variants on standard benchmarks and in scenarios with small datasets or wit modified evaluation environments. Additionally, the agent achieves trajectory stitching without architectural changes. However, training separate Transformer models adds complexity. Future work could explore combining these models, as they share inputs, or training in an iterative manner using generated counterfactual samples as training data. Another potential approach is to used an weighted approach instead of a hard cut-off for the uncerainty filter, which might lead to better results.

References

[Agarwal *et al.*, 2020] Rishabh Agarwal, Dale Schuurmans, and Mohammad Norouzi. An optimistic perspective on offline reinforcement learning. In *Proceedings of the 37th International Conference on Machine Learning*, 2020.

[Bahadori *et al.*, 2022] Taha Bahadori, Eric Tchetgen Tchetgen, and David Heckerman. End-to-end balancing for causal continuous treatment-effect estimation. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 1313–1326. PMLR, 17–23 Jul 2022.

[Bellemare *et al.*, 2013] Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013.

[Bhargava *et al.*, 2024] Prajjwal Bhargava, Rohan Chitnis, Alborz Geramifard, Shagun Sodhani, and Amy Zhang. When should we prefer decision transformers for offline reinforcement learning? In *The Twelfth International Conference on Learning Representations*, 2024.

[Chen *et al.*, 2021] Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Michael Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: reinforcement learning via sequence modeling. In *Proceedings of the 35th International Conference on Neural Information Processing Systems*, pages 15084–15097, 2021.

[Frauen *et al.*, 2023] Dennis Frauen, Tobias Hatt, Valentyn Melnychuk, and Stefan Feuerriegel. Estimating average causal effects from patient trajectories. In *Proceedings of the Thirty-Seventh AAAI Conference on Artificial Intelligence*. AAAI Press, 2023.

[Fu *et al.*, 2020] Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

[Kamath, 2015] Gautam Kamath. Bounds on the expectation of the maximum of samples from a gaussian. *URL http://www. gautamkamath. com/writings/gaussian max. pdf*, 10(20-30):31, 2015.

[Kidambi *et al.*, 2020] Rahul Kidambi, Aravind Rajeswaran, Praneeth Netrapalli, and Thorsten Joachims. Morel: model-based offline reinforcement learning. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, pages 21810–21823, 2020.

[Killian *et al.*, 2022] Taylor W Killian, Marzyeh Ghassemi, and Shalmali Joshi. Counterfactually guided policy transfer in clinical settings. In *Conference on Health, Inference, and Learning*, pages 5–31. PMLR, 2022.

[Kim *et al.*, 2023] Jeonghye Kim, Suyoung Lee, Woojun Kim, and Youngchul Sung. Decision convformer: Local filtering in metaformer is sufficient for decision making. *arXiv preprint arXiv:2310.03022*, 2023.

[Kostrikov *et al.*, 2021a] Ilya Kostrikov, Rob Fergus, Jonathan Tompson, and Ofir Nachum. Offline reinforcement learning with fisher divergence critic regularization. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 5774–5783. PMLR, 18–24 Jul 2021.

[Kostrikov *et al.*, 2021b] Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. *arXiv preprint arXiv:2110.06169*, 2021.

[Kumar *et al.*, 2020] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, 2020.

[Le Pham Van *et al.*, 2024] Linh Le Pham Van, Hung The Tran, and Sunil Gupta. Policy learning for off-dynamics RL with deficient support. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems*, pages 1093–1100, 2024.

[Levine *et al.*, 2020] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

[Li *et al.*, 2020] Rui Li, Zach Shahn, Jun Li, Mingyu Lu, Prithwish Chakraborty, Daby Sow, Mohamed Ghalwash, and Li-wei H Lehman. G-net: a deep learning approach to g-computation for counterfactual outcome prediction under dynamic treatment regimes. *arXiv preprint arXiv:2003.10551*, 2020.

[Melnychuk *et al.*, 2022] Valentyn Melnychuk, Dennis Frauen, and Stefan Feuerriegel. Causal transformer for estimating counterfactual outcomes. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 15293–15329. PMLR, 17–23 Jul 2022.

[Neyman, 1923] Jerzy Neyman. On the application of probability theory to agricultural experiments. essay on principles. *Ann. Agricultural Sciences*, pages 1–51, 1923.

[Pearl and Mackenzie, 2018] Judea Pearl and Dana Mackenzie. The book of why: The new science of cause and effect, 2018.

[Pitis *et al.*, 2020] Silviu Pitis, Elliot Creager, and Animesh Garg. Counterfactual data augmentation using locally factored dynamics. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA, 2020.

[Pitis *et al.*, 2022] Silviu Pitis, Elliot Creager, Ajay Mandlekar, and Animesh Garg. Mocoda: model-based counterfactual data augmentation. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA, 2022.

[Pomerleau, 1988] Dean A. Pomerleau. Alvinn: an autonomous land vehicle in a neural network. In *Proceedings of the 2nd International Conference on Neural Information Processing Systems*, NIPS'88, page 305–313, Cambridge, MA, USA, 1988. MIT Press.

[Robins and Hernan, 2008] James Robins and Miguel Hernan. Estimation of the causal effects of time-varying exposures. *Chapman & Hall/CRC Handbooks of Modern Statistical Methods*, pages 553–599, 2008.

[Rubin, 1978] Donald B Rubin. Bayesian inference for causal effects: The role of randomization. *The Annals of statistics*, pages 34–58, 1978.

[Schmidhuber, 2019] Juergen Schmidhuber. Reinforcement learning upside down: Don't predict rewards–just map them to actions. *arXiv preprint arXiv:1912.02875*, 2019.

[Silver *et al.*, 2017] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017.

[Sun *et al.*, 2023] Zexu Sun, Bowei He, Jinxin Liu, Xu Chen, Chen Ma, and Shuai Zhang. Offline imitation learning with variational counterfactual reasoning. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, 2023.

[Sun *et al.*, 2024] Yuewen Sun, Erli Wang, Biwei Huang, Chaochao Lu, Lu Feng, Changyin Sun, and Kun Zhang. Acamda: improving data efficiency in reinforcement learning through guided counterfactual data augmentation. In *Proceedings of the Thirty-Eighth AAAI Conference on Artificial Intelligence*. AAAI Press, 2024.

[Sutton and Barto, 2018] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. A Bradford Book, Cambridge, MA, USA, 2018.

[van Hoof *et al.*, 2015] Herke van Hoof, Tucker Hermans, Gerhard Neumann, and Jan Peters. Learning robot in-hand manipulation with tactile features. In *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, pages 121–127. IEEE, 2015.

[Vaswani *et al.*, 2017] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30, 2017.

[Wang *et al.*, 2018] Lu Wang, Wei Zhang, Xiaofeng He, and Hongyuan Zha. Supervised reinforcement learning with recurrent neural network for dynamic treatment recommendation. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 2447–2456, 2018.

[Wu *et al.*, 2023] Yueh-Hua Wu, Xiaolong Wang, and Masashi Hamaya. Elastic decision transformer. In *Proceedings of the International Conference on Neural Information Processing Systems*, pages 18532–18550, 2023.

[Xie *et al.*, 2021] Tengyang Xie, Ching-An Cheng, Nan Jiang, Paul Mineiro, and Alekh Agarwal. Bellman-consistent pessimism for offline reinforcement learning. In *Proceedings of the 35th International Conference on Neural Information Processing Systems*, NIPS '21, 2021.

[Xie *et al.*, 2023] Zhihui Xie, Zichuan Lin, Deheng Ye, Qiang Fu, Wei Yang, and Shuai Li. Future-conditioned unsupervised pretraining for decision transformer. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023.

[Yamagata *et al.*, 2023] Taku Yamagata, Ahmed Khalil, and Raul Santos-Rodriguez. Q-learning decision transformer: Leveraging dynamic programming for conditional sequence modelling in offline RL. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 38989–39007. PMLR, 23–29 Jul 2023.

[Yu *et al.*, 2020] Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. Mopo: model-based offline policy optimization. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, pages 14129–14142, 2020.

[Zhao *et al.*, 2024] Ziqi Zhao, Zhaochun Ren, Liu Yang, Fajie Yuan, Pengjie Ren, Zhumin Chen, Xin Xin, et al. Offline trajectory generalization for offline reinforcement learning. *arXiv preprint arXiv:2404.10393*, 2024.

[Zheng *et al.*, 2022] Qinqing Zheng, Amy Zhang, and Aditya Grover. Online decision transformer. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 27042–27059. PMLR, 17–23 Jul 2022.

[Zhu *et al.*, 2015] Yeying Zhu, Donna L Coffman, and Debashis Ghosh. A boosting algorithm for estimating generalized propensity scores with continuous treatments. *Journal of causal inference*, 3(1):25–40, 2015.

[Zhuang *et al.*, 2024] Zifeng Zhuang, Dengyun Peng, Jinxin Liu, Ziqi Zhang, and Donglin Wang. Reinformer: max-return sequence modeling for offline rl. In *Proceedings of the 41st International Conference on Machine Learning*, pages 62707–62722, 2024.