Reliability-Guaranteed and Reward-Seeking Sequence Modeling for Model-Based Offline Reinforcement Learning

Shenghong He[1], Chao Yu[1,2*], Qian Lin[1], Yile Liang[3], Donghui Li[3], Xuetao Ding[3]

[1]School of Computer Science and Engineering, Sun Yat-Sen University, Guangzhou, China
[2] Pengcheng Laboratory, Shenzhen, China
[3] Meituan, Beijing, China
{heshh23, linq67}@mail2.sysu.edu.cn, yuchao3@mail.sysu.edu.cn, {liangyile, lidonghui03, dingxuetao}@meituan.com

Abstract

As a data-driven learning approach, model-based offline reinforcement learning (MORL) aims to learn a policy by exploiting a dynamics model derived from an existing dataset. Applying conservative quantification to the dynamics model, most existing works on MORL generate trajectories that approximate the real data distribution to facilitate policy learning. However, these methods typically overlook the influence of historical information on environmental dynamics, thus generating unreliable trajectories that fail to align with the true data distribution. In this paper, we propose a new MORL algorithm called **R**eliability-guaranteed and Reward-seeking **T**ransformer (RT). RT can avoid generating unreliable trajectories through the calculation of cumulative reliability of the trajectories, which is a weighted variational distance between the generated trajectory distribution and the true data distribution. Moreover, by sampling candidate actions with high rewards, RT can efficiently generate high-reward trajectories from the existing offline data, thereby further facilitating policy learning. We theoretically prove the performance guarantees of RT in policy learning, and empirically demonstrate its effectiveness against state-of-the-art model-based methods on several offline benchmark tasks and a large-scale industrial dataset from an on-demand food delivery platform.

Introduction

Model-based offline reinforcement learning (MORL) (Kidambi et al. 2020) aims to learn a policy by leveraging a dynamics model derived from an existing dataset, without requiring interaction with the environment. Recent approaches often leverage uncertainty quantification (Yu et al. 2020; Wagenmaker, Shi, and Jamieson 2023; Bhardwaj et al. 2024) and value function penalization (Yu et al. 2021; Sun et al. 2023) to learn the dynamics model, ensuring that the data generated based on current information (e.g., state and action) approximate the real data distribution.

However, these methods implicitly assume that the current information fully provides all relevant information for data generation at the next time step. Thus, unreliable trajectories that may not exist in the real environment can be generated due to the neglect of considering the impact of historical information on the environmental dynamics. For instance, considering a robot performing cleaning tasks in a complex room, at a historical time step $t-k$, the robot takes the action of moving a chair, causing a partial obstruction in the passage. At the current time step t , employing a generative model to generate trajectories could result in the robot passing through the chair due to the model's inability to remember the historical action of moving a chair. This generated trajectory is unreliable because the robot cannot pass through the chair in the real environment. Furthermore, by simply focusing on imitating the observed data transitions, existing methods generally fail to identify states and actions that could lead to high cumulative rewards, thus significantly impairing the learning performance.

In this paper, we propose a novel method called **R**eliability-guaranteed and Reward-seeking **T**ransformer (RT), which integrates sequence modeling with an adaptive reliability estimation mechanism to generate reliable and high-reward trajectories. Specifically, by taking advantage of the sequence modeling capabilities of Transformer, RT can capture historical information between different positions in the input sequence to generate the data sequence (e.g., states, actions, and rewards). Then, RT calculates the variation distance between the generated data and the real data to derive the reliability value of the state-action pair at each time step. Based on these reliability values, RT computes the cumulative reliability of the generated sequence to adaptively determine the generation length, which effectively mitigates error accumulation and ultimately improves the overall reliability of the generated sequences. Moreover, RT estimates the likelihood of future high rewards for candidate actions and uses these estimations as generation conditions to generate high-reward trajectories. Empirical results on various continuous control tasks and a real-world on-demand food delivery task demonstrate that our method achieves superior learning performance compared to existing MORL baselines.

Related Work

Model-based offline RL methods. MORL (Kidambi et al. 2020) learn an approximate dynamics model of the environment and utilize planning algorithms to search for optimal trajectories within that model. However, research indicates that even small generation errors can significantly

*Corresponding author.
Copyright © 2026, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

degrade the performance of multi-step rollouts, as generation errors compound, causing the model to deviate from the high-precision region after a few steps (Talvitie 2017; Asadi, Misra, and Littman 2018). To address this issue, some studies (Yu et al. 2020; Diehl et al. 2021) adopt an ensemble of models to predict states and actions, and use uncertainty quantification to discard samples with potentially large errors. In addition, some studies (Chen et al. 2022) restrict the learning policy from accessing areas with significant differences between the learned and the real dynamics to prevent policy learning failures due to generation errors.

Recent studies (Wang et al. 2021; Lyu, Li, and Lu 2022; Lu et al. 2023) suggest the use of fixed truncation techniques (Janner et al. 2019) to generate shorter trajectories and thus reduce the impact of generation errors on generated data. This generated data is then combined with the original data to create a new dataset, which can be utilized in model-free offline algorithms. However, these studies neglect the influence of historical information on environmental dynamics, potentially resulting in the generation of data that fails to align with the environment. Unlike the above methods, RT extends the offline dataset by capturing historical information between different locations in the input sequence and using reliability assessment to generate reliable trajectories. Moreover, RT calculates the cumulative reliability of the generated trajectory to dynamically adjust the generation length, enabling a more flexible mitigation of the adverse effects of accumulated generation errors.

Sequence modeling. Recent research (Janner, Li, and Levine 2021; Chen et al. 2021; Yamagata, Khalil, and Santos-Rodríguez 2023; Badrinath et al. 2023; Gao et al. 2024) shifts the paradigm of RL from traditional MDP to sequence modeling. In particular, Trajectory Transformer (TT) (Janner, Li, and Levine 2021) treats offline RL as a sequence modeling problem and further leverages the capabilities of sequence models by using beam search to incorporate states, actions, and rewards. At the some time, Decision Transformer (DT) (Chen et al. 2021) learns the distribution of trajectories and predicts actions based on the given target reward and preceding states, rewards, and actions. Unlike these works, our work focuses on addressing the problem of extrapolation errors (Fujimoto, Meger, and Precup 2019) caused by insufficient data coverage in offline RL. To this end, RT generates high-reward trajectories from suboptimal datasets to facilitate policy learning in model-based RL through reliability-guaranteed sequence modeling.

Preliminaries

MDPs and offline RL. A Markov Decision Process (MDP) can be defined as $M = (S, A, R, P, \rho_0, \gamma)$, where S is the state space, A is the action space, $R(s, a)$ is the reward function, $P(s'|s, a)$ is the transition function, ρ_0 is the initial state distribution, and $\gamma \in (0, 1)$ is the discount factor. A policy $\pi : S \times A \to [0, 1]$ takes action a at a state s with probability $\pi(a|s)$. The goal of RL is to find the optimal policy π^* that maximizes the expected return $\pi^* = \operatorname{argmax}_\pi \mathbb{E}_\pi[\sum_{t=0}^{T} \gamma^t R(s_t, a_t)]$. In the offline RL setting, the agent only has access to a static dataset $\mathcal{D}_{\text{env}} = \{(s, a, r, s')\}$. The agent's objective is to learn a policy without interaction with the environment for any additional online exploration.

Model-based offline RL. Model-based offline RL methods use a dataset to learn the dynamics model $\hat{P}$, which is usually trained by maximum likelihood estimation: $\min_{\hat{P}} \mathbb{E}_{(s,a,s')\sim D_{env}}[-\log \hat{P}(s'|s, a)]$. Additionally, when the reward function is unknown, a model of the reward function $\hat{R}(s, a)$ can be trained. Once the dynamics model is trained, samples generated by $\hat{P}$ are then placed into the model buffer $\mathcal{D}_{\text{model}}$, which is merged with the offline data $\mathcal{D}_{\text{env}} \cup \mathcal{D}_{\text{model}}$ to learn a policy.

Method

The overall process of RT is described in Algorithm 1. RT first learns a dynamics model from the offline dataset and determines a reliability threshold via the reconstruction error of the VAE (lines 1 to 11). Next, RT randomly selects a trajectory from the offline dataset and generates a reverse trajectory from any subsequent state along that trajectory (lines 12 to 16). Then, RT calculates the accumulated reliability of the trajectory and compares it with a reliability threshold to ensure that the generated trajectory aligns with the environment dynamics. This generated trajectory is then combined with the original data to create a new dataset, which is used to train the model-free algorithm (lines 17 to 21).

Reliability-Guaranteed Sequence Generation

Inspired by previous sequence modeling learning (Chen et al. 2021), we treat each input trajectory τ as a sequence and add return-to-go $R_t = \sum_{h=t}^{T} \gamma^{h-t} r_h$ as auxiliary information at each time step t, which acts as future heuristics for further generating data. Specifically, each trajectory $\tau = (s_1, a_1, r_1, R_1, \cdots, s_T, a_T, r_T, R_T)$ is used as an input sequence for RT, which is trained using the standard teacher-forcing method (He et al. 2022), as expressed by:

$$\begin{aligned} \log P_\theta(\tau_t|\tau_{<t}) = &\log P_\theta(s_t|\tau_{<t}) + \log P_\theta(a_t|s_t, R_t, \tau_{<t}) \\ &+ \log P_\theta(r_t|a_t, s_t, \tau_{<t}) \\ &+ \log P_\theta(R_t|r_t, a_t, s_t, \tau_{<t}), \end{aligned} \tag{1}$$

where $\tau_{<t}$ represents the trajectory from the initial state to time step $t-1$, θ is the parameter of RT, and P_θ is the induced conditional probability. The training objective is to maximize the log-likelihood of the trajectory:

$$\mathcal{L}_\tau = \sum_{t=1}^{T} \log P_\theta(\tau_t|\tau_{<t}). \tag{2}$$

However, during the sequence generation, the model generation error at the moment t is continuously accumulated by the subsequent generation process, which may cause the generated data to deviate from the distribution of the original data. In order to solve this issue, we propose a reliability

Algorithm 1: The RT algorithm

Input: Offline dataset D_{env}, generation number H, iteration N, model-free offline RL algorithm (e.g., BCQ)
Parameter: Randomly initialize RT parameters θ, ψ and ϕ,
Output: a learned policy

// Train RT
for i=0, $\cdots$, N **do**
  Compute $\mathcal{L}_\tau$ using the dataset D_{env}
  Updata RT network parameter θ
end for
// Train VAE to get the cumulative threshold α
for for j=0, $\cdots$, N **do**
  Compute $L(\psi, \phi)$ using offline dataset D_{env}
  Update VAE network parameter ψ and ϕ
end for
Use VAE to calculate the maximum distribution error of offline dataset to get a cumulative reliability threshold α
//Generate trajectories
Initialize the buffer D_{model}
for i=0, $\cdots$, H **do**
  Sample state s_{t+1} from the τ in D_{env}
  Generate backward trajectory $\hat{\tau} = \{s_{t-k}, a_{t-k}, r_{t-k}\}_{k=0}^{t}$ using RT and reliability estimation.
  Merge trajectories $\tau' = \hat{\tau}_{<t} + \tau_{\geq t}$
  $D_{model} \leftarrow D_{model} \cup \tau'$
end for
Get composite dataset $D \leftarrow D_{env} \cup D_{model}$
Learn the model-free offline algorithm using D

estimation mechanism, which automatically determines the generated trajectory lengths based on the cumulative reliability along the trajectory and incorporates these reliability values into the pessimistic MDP (Blanchet et al. 2023; Zhang et al. 2023) to provide performance bounds for the learned policy. We first define the cumulative reliability of trajectories and the truncation metric:

Definition 1 (**Cumulative Reliability**). *Given a trajectory τ, the cumulative reliability along the trajectory at step t is defined as:*

$$\begin{aligned}\Gamma(s_t, \tau_{<t}) &= \mathbb{D}(P(\cdot|s_i, \tau_{<i}), \hat{P}(\cdot|s_i, \tau_{<i})) \\ &= \sum_{i=1}^{t} \frac{\exp(e_i)}{\sum_{j=1}^{n} \exp(e_j)} \cdot D_{IST}(P(\cdot|s_i, a_i), \hat{P}(\cdot|s_i, a_i)) \end{aligned} \quad (3)$$

where $\hat{P}$ is the estimated transition probability of the environment after training, P is the true environment dynamics, e is an attention value, n is the trajectory segment length, and D_{IST} represents the total variation distance (Ho and Yeung 2010) between two distributions $\hat{P}$ and P.

Definition 2 (**Truncation Metric**). *Given a cumulative reliability Γ, the truncation metric at t as:*

$$U_t = \begin{cases} 0 & (i.e.,\ reliable)\ if\ \Gamma(s_t, \tau_{<t}) \leq \alpha \\ 1 & (i.e., unreliable)\ otherwise \end{cases}, \quad (4)$$

where α *represents the cumulative threshold for generating reliable trajectories*.

Γ provides a metric of the cumulative reliability of the trajectories generated by the model, while U defines the truncation point of the generated trajectory. To directly quantify the impact of generation errors on policy learning, we integrate the reliability values into the pessimistic MDP framework (Kidambi et al. 2020; Blanchet et al. 2023; Zhang et al. 2023). Specifically, the α-pessimistic MDP is defined by the tuple $\hat{M}_p = \{S, A, \hat{R}, \hat{P}_p, \rho_p, \gamma\}$, where S,A, ρ_p and γ retain the same definitions as in the original MDP. The transition function and reward function are dynamically updated and evaluated using the truncation metric, which can be defined as follows:

$$\hat{P}(\cdot|s', \tau_{<t}) = \begin{cases} 0 & if\ U_t = 1 \\ \hat{P}(\cdot|s_t, \tau_{<t}) & otherwise, \end{cases} \quad (5)$$

$$\hat{R}(s_t, a_t) = \begin{cases} 0 & if\ U_t = 1 \\ r(s_t, a_t) - \beta \frac{\Gamma(s_t, \tau_{<t})}{\alpha}) & otherwise, \end{cases} \quad (6)$$

where β is the reliability penalty hyperparameters for each state-action pair (s, a).

By introducing the α-pessimistic MDP, we incorporate trajectory generation errors into the reward function, ensuring that trajectories with larger errors receive smaller reward values. In this way, the policy learns a lower Q-value when encountering such trajectories, thereby becoming more conservative and robust (Kidambi et al. 2020; Kumar et al. 2020). Moreover, if the truncation metric $U_t = 1$, indicating that the cumulative reliability Γ at step t exceeds the threshold α, the trajectory generation process will be terminated. By using reliability to guide policy learning, we can theoretically derive a performance bound for the policy learned from the generated trajectories.

Theorem 1. *Let π be the policy learned from the α-pessimistic MDP, and π^* be the optimal policy in the true MDP M. Then the performance difference between the two policies in the true MDP satisfies: $\left|V^P(\pi^*) - V^P(\pi)\right| \leq C\ \alpha\,\beta$, where C is constants and β is the penalty hyperparameter introduced in Eq. (6), and $V^P(\pi)$ denotes the value function of the policy π in the real M.*

Proof. For proof, see Appendix B. □

Theorem 1 demonstrates that incorporating cumulative reliable values into an α-pessimistic MDP can provide a strict performance lower bound for a policy in the real MDP. Simply put, by properly setting the hyperparameters α and β, the performance gap between the learned policy and the true optimal policy can be controlled within the $O(\alpha\beta)$.

However, the cumulative reliability and threshold α cannot be directly derived through the calculation of D_{IST} because $P(\cdot|s, \tau_{<t})$ are typically unknown. To solve this issue, we employ a Variational Autoencoder (VAE) (Li and Yao 2024; Zhao et al. 2024) to calculate the distribution error of (s, a, s') as a replacement for the total variation distance $D_{IST}(P(\cdot|s, a), \hat{P}(\cdot|s, a))$. Specifically, an encoder maps $x = (s, a, s')$ to a latent vector z, while a decoder reconstructs x from z. The conditional probability of the encoder

is $q(z|x)$, and the constrained probability distribution of the decoder is $p(x|z)$. Then, the loss function of the VAE is expressed as:

$$L(\psi, \phi) = -\mathbb{E}_{z\sim q(z|x)}[\log(p(x|z))] + D_{\mathrm{KL}}(q(z|x)||p(z)), \tag{7}$$

where ψ is the encoder parameter, ϕ is the decoder parameter, and D_{KL} is KL divergence. By learning the VAE, we identify the maximum reconstruction error σ in the offline data and set it as the cumulative threshold α:

$$\alpha = \sum_{i=1}^{t} \frac{\exp(e_i)}{\sum_{j=1}^{n} \exp(e_j)} \cdot \sigma, \tag{8}$$

Next, the generated states and actions use the VAE to calculate the D_{IST}, which is then used as a reliable value for the generated data to compute the cumulative reliability of the trajectory.

Reward-Seeking Trajectory Generation

While reliability-guaranteed sequence generation can generate trajectories that align with environment dynamics, it does not guarantee the generation of high-reward trajectories that effectively facilitate policy learning. To address this issue, we use the return at time t as a condition to guide the model to effectively generate the action at $t+1$. Specifically, RT applies a binary classifier $\mathbb{P}(H_t|\cdots)$ to identify whether an action brings a high reward before taking an action at time t, and applies the Bayes' rule to approximate the high reward distribution $\mathbb{P}(R_t, \ldots|H_t) \propto \mathbb{P}(H_t|R_t, \ldots)\mathbb{P}(R_t, \ldots)$ (Lee et al. 2022), where H_t denotes the high reward at time t. This probabilistic modeling forms a simple autoregressive process in which R_t is first generated based on a high and reasonable log-probability, and actions are then generated according to $P_\theta(a_t \mid R_t, \ldots)$, which can be considered a variant of the class-conditional model that automatically takes actions with high rewards at each timestep to generate high-reward trajectories.

To further enhance the generation of high-reward trajectories, RT employs a backward generation mechanism to generate trajectories by randomly selecting a state from the trajectory as the starting point and modeling the states, actions, and cumulative rewards in a backward sequence, thus ensures that the generated trajectories include the actual goal state[1] typically associated with high rewards. Specifically, each trajectory $\tau_{back} = (s_T, a_T, r_T, R_T, \cdots, s_1, a_1, r_1, R_1)$ is used as an input sequence for RT, which is expressed as follows:

$$\begin{aligned}\log P_\theta(\tau_t|\tau_{>t}) = &\log P_\theta(s_t|\tau_{>t}) + \log P_\theta(a_t|s_t, R_t, \tau_{>t}) \\ &+ \log P_\theta(r_t|a_t, s_t, \tau_{>t}) \\ &+ \log P_\theta(R_t|r_t, a_t, s_t, \tau_{>t}),\end{aligned} \tag{9}$$

where τ denotes τ_{back}, and $\tau_{>t}$ represents the trajectory from time step t to the terminal state. The training objective is to maximize:

$$\mathcal{L}_\tau = \sum_{t=1}^{T} \log P_\theta(\tau_{<t}|\tau_{\geq t}). \tag{10}$$

[1]Within the scope of this research, the goal state can be defined as any state along the trajectory that yields a maximal reward

After the training process, the learned RT generates trajectories to augment the offline dataset D. For the original trajectories $\tau \in D$ in the original dataset, RT begins generating trajectories backward from time step T':

$$\begin{aligned}\hat{\tau} = (&\hat{s}_0, \cdots, \hat{s}_{T'-1}, \hat{a}_{T'-1}, \hat{r}_{T'-1}, \hat{R}_{T'-1}, \\ &s_{T'}, a_{T'}, r_{T'}, R_{T'}, \cdots) \sim P_\theta(\tau_{<T'}|\tau_{\geq T'}),\end{aligned} \tag{11}$$

where $\tau_{\geq T'}$ denotes the original trajectory truncated at time step T'. Then, the generated sequence is concatenated with the truncated original trajectory, resulting in the formation of a new trajectory $\tau' = \hat{\tau}_{<T'} + \tau_{\geq T'}$, where $+$ is the concatenation operation. By adopting this approach, we are able to augment a solitary high-reward trajectory into the trajectories, consequently enriching the offline dataset.

Experiments

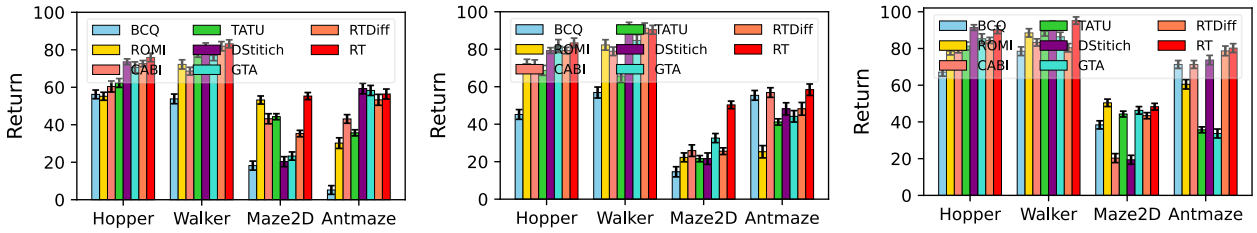
In this section, we first compare RT with existing mainstream offline RL methods across various domains in the D4RL benchmark (Fu et al. 2020), including Maze2D, MuJoCo, AntMaze (see Appendix C.2 for a detailed information) and an industrial on-demand food delivery (see Appendix E.1), and then analyze the impact of different mechanisms or components of RT on its performance.

Comparison Methods

Since our goal is to augment offline datasets to improve the performance of offline RL algorithms, we compare RT with recent augmentation algorithms, including **ROMI** (Wang et al. 2021), **CABI** (Lyu, Li, and Lu 2022)), **TATU** (Zhang et al. 2023), **DStitch** (Li et al. 2024), **GTA** (Lee et al. 2024) and **RTDiff** (Yang and Wang 2025). In addition, in order to intuitively understand the performance difference between RT and notable offline RL algorithms, RT is compared to the following notable methods, including offline model-free algorithms (i.e., **BCQ** (Fujimoto, Meger, and Precup 2019), **CQL** (Kumar et al. 2020), **IQL** (Kostrikov, Nair, and Levine 2022), and **DT** (Chen et al. 2021)), as well as offline model-based algorithms (i.e., **MOPO** (Yu et al. 2020), **MOReL** (Kidambi et al. 2020) and **MOPP** (Zhan, Zhu, and Xu 2022). More description of the above methods is provided in the Appendix C.5.

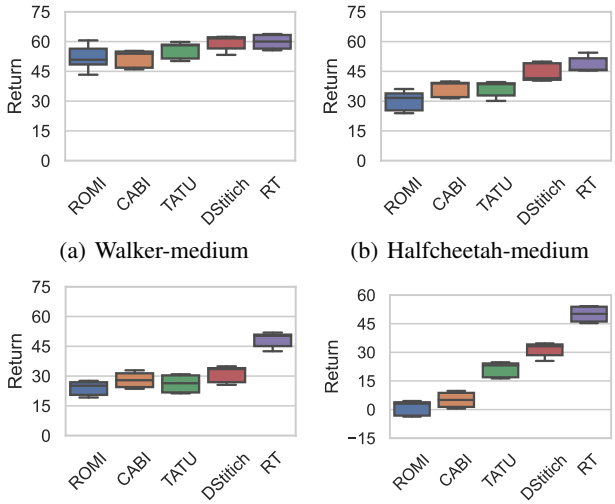
Results in the D4RL Benchmark

Comparison to augment methods. ROMI, CABI, TATU, DStitch and RT are combined with model-free methods (e.g., BCQ, CQL and IQL) and evaluated on Hopper-medium, Walker-medium, sparse Maze2D-medium and diverse Antmaze-medium. The experimental results shown in Figure 1 demonstrate that RT consistently outperforms ROMI CABI and TATU. Moreover, both ROMI, CABI and TATU perform worse than IQL alone in the Antmaze-medium environment because their inability to generate high-reward trajectories introduces a large amount of unrewarded data, leading to inaccurate policy constraints and then a suboptimal policy. Although DStitch can merge high-reward and low-return trajectories, the presence of a high proportion of suboptimal data poses a significant challenge to its ability to find suitable trajectories for merging, as in the



(a) BCQ (b) CQL (c) IQL

Figure 1: Performance of RT, ROMI, CABI, TATU and DStitch combined with different model-free algorithms (averaged over 10 random seeds)



(a) Walker-medium (b) Halfcheetah-medium

(c) AntMaze-umaze (d) Maze2D-umaze

Figure 2: Results of BC learned using trajectories generated by RT, ROMI, CABI, TATU and DStitch (averaged over 10 random seeds)

Maze2D environment. Similarly, GTA lacks reliability verification for generated data, causing deviation from the original distribution and consequently limited performance improvement. Likewise, while RTDiff eliminates overestimation risks by avoiding planning over unknown states, it sacrifices the ability to effectively explore high-reward states. In contrast, RT employs sequential modeling to generate high-reward trajectories starting from the goal state and utilizes reliability estimation to automatically determine the length of generated trajectories, resulting in more robust performance across different environments.

Next, to evaluate the return of the trajectories generated by RT, we employ Behavioral Cloning (BC) (Torabi, Warnell, and Stone 2018) to learn a policy from the generated data, as the effectiveness of BC is highly dependent on the quality and quantity of the high-reward data. We train ROMI, CABI, TATU, DStitch and RT on the Walker-random/medium, Halfcheetah-random/medium, Antmaze-umaze/medium, and Maze2D-umaze/medium environments, and then select 50 trajectories from these environments to generate new trajectories by sampling each selected trajectory 10 times for learning BC. The results in Figure 2 show that RT shows strong learning performance (see Appendix D.1 for the full results), while ROMI, CABI, TATU and DStitch exhibit large performance fluctuations in different environments, which can be attributed to the fact that ROMI, CABI, TATU and DStitch prioritize generating trajectories that adhere to the original data distribution without considering the returns of the generated trajectories, preventing BC from learning an effective policy from the generated trajectories. In contrast, RT takes into account both the reliability and high-reward of the data when generating trajectories, which allows to achieve better learning performance across different environments.

Comparison to notable offline methods. Table 1 shows the results of RT+BCQ compared with recent offline RL methods (refer to the Appendix D.3 for the full results). The experiments demonstrate that RT can significantly enhance the performance of the original algorithm and outperform all the baselines. It can be seen that the model-based methods have large performance fluctuations in different environments due to the failure in generating high-reward trajectories from offline data; for instance, MOReL shows performance comparable to model-free methods in the MuJoCo environment but accumulates almost zero rewards in Maze2D and Antmaze. RT uses Reliability-guaranteed sequence modeling to adaptively determine trajectory lengths and employs high rewards as conditions to generate high-reward trajectories from offline data, making it highly effective across diverse environments.

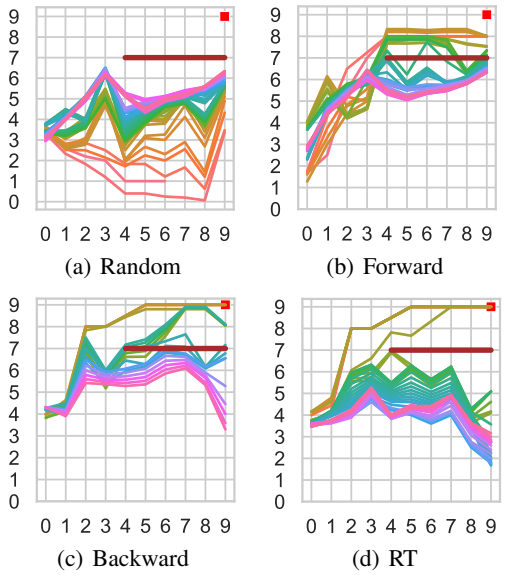
Further Analysis

To better understand the effectiveness of RT in addressing unreliable trajectory problems, we construct a game called BoxBall, where the ball needs to move from the starting point to the goal state using up, down, left, and right actions while navigating around a wall in front of the goal state (see Appendix C.3 for more environmental information).

In this environment, we use DQN to collect a dataset and use it to train forward, backward, and RT models. Then, we use the trained forward model, backward model and RT to generate and visualize 50 trajectories. As shown in Figure 3, the trajectory of the random strategy does not include the action of walking through the wall, while the trajectories gen-

	Environment	BCQ	CQL	IQL	DT	MOReL	RT+BCQ	RT+CQL	RT+IQL
medium-replay	Hopper	33.3	30.6	96.8	83.6	93.8	99.3±7.2	106.3±3.5	108.3±6.7
	Walker	16.8	15.8	74.9	67.2	48.7	70.3±5.8	86.5±8.7	94±5.7
	Halfcheetah	39.2	40.7	45.1	40.3	39.8	47±3.5	75.6±7.8	100±5.6
sparse	Maze2D-umaze	49.3	19.1	20.1	52.6	0.0	54.3±5.6	45.6±6.8	44.3±7.2
	Maze2D-medium	18.2	14.6	10.6	13.2	1.3	55.3±4.8	50.3±4.8	48.3±6.7
	Maze2D-large	32.6	17.2	19.3	3.4	0.0	40.8±5.2	44.6±7.4	39.8±6.9
diverse	Antmaze-umaze	48.8	83.6	63.8	53.6	0.0	71.3±4.5	90.6±5.7	86.5±7.8
	Antmaze-medium	5.2	55.4	71.3	44.2	0.6	56.4±5.4	58.4±6.5	80.2±5.9
	Antmaze-large	1.9	13.6	44.2	24.8	0.0	46.8±3.9	53.2±4.7	60.0±3.2

Table 1: Normalized returns of different methods on the D4RL benchmark (Bold black font indicates the highest return)



(a) Random (b) Forward (c) Backward (d) RT

Figure 3: Trajectory Visualization of BoxBall. (a)∼(d) denote the visualization trajectories of the random policy, forward model, backward model and RT, respectively.

erated by the forward model contain many unreliable trajectory (i.e., trajectories cross the wall) and fail to reach the goal state. The values of unreliable trajectories may be arbitrarily misestimated, and then the erroneous values may be propagated throughout the state-action space leading to overestimation of Q-values during training. Despite employing the same backward trajectory generation method, the backward model generates some unreliable trajectories due to cumulative errors and model lacks of ability to capture long-term dependencies. RT utilizes sequence modeling techniques to effectively learn the distribution of trajectories and employs reliability estimation to adaptively determine the generated trajectories, which ensures that actions involving passing through walls do not occur. A more detailed analysis of generating reliable trajectories is discussed in the Appendix D.4.

In order to explore whether backward generation is more advantageous than forward generation, we conduct forward and backward comparison experiments on Maze2D and Antmaze. experimental results (see Appendix D.5) shows that RT enables safe generalization across all layouts (i.e., RT can ensure that the generated trajectories contain goal states), thus ensuring RT+BCQ significantly outperforms FT+BCQ and BCQ. Moreover, we conduct experiments to explore the impact of reliability estimation on RT performance. Experiments demonstrate that RT uses reliability estimation, which makes it more flexible and diverse when generating trajectories, thereby improving the performance of IQL (refer to Appendix D.6).

In addition, we conduct an ablation study over $\beta \in 0.1, 0.3, 0.5, 0.7, 0.8, 1.0, 1.3, 1.5, 1.8, 2.0, 2.3, 2.6, 3.0$, guided by established ranges in prior work. Specifically, the β-ablation experiments are designed to analyze the effect of β on the IQL algorithm, and are conducted across four benchmark environments: Hopper-medium, Walker2d-medium, dense Maze2D-medium, and diverse Antmaze-medium. The experimental results in Appendix D.7 show that a moderate value (e.g., $\beta = 1$) yields a good balance between data reliability and policy performance.

Real-world Industrial OFD Data Experiments

To evaluate the performance of RT in real-world deployment, we conduct experiments within a real-world on-demand food delivery (OFD) system. Specifically, OFD service system (Liang et al. 2023) consists of consumers, a scheduling platform, merchants and delivery riders, where consumers submit orders at different times and from diverse geographic locations, while the scheduling platform gathers these orders in real-time, assigns them to the corresponding merchants, and schedules riders to pick up and deliver them within a specified time limit. As the central hub connecting the entire service system, the scheduling platform not only effectively links consumers, merchants and delivery riders but also needs to precisely balance the interests of all participating parties. Specifically, consumers prioritize on-time delivery for an optimal dining experience, and merchants emphasize food freshness to enhance customer satisfaction, while delivery riders complete multiple orders through intelligent route planning to earn higher incomes. To reconcile these diverse needs, the scheduling platform employs dynamic resource allocation and demand forecasting to maximize operational efficiency while ensuring that all stakeholders achieve satisfactory outcomes.

Using existing algorithms to augment the OFD data is

OT	Origial	ROMI	CABI	TATU	DStitch	RT
1	75.4	65.3	80.6	75.4	56.7	79.8
2	80.6	80.3	75.4	60.3	75.8	80.2
3	85.3	71.3	69.7	85.4	69.3	77.8
4	73.6	85.3	64.8	91.5	76.8	84.2
5	68.4	55.4	65.3	70.4	61.7	73.5
6	65.3	93.6	57.8	88.5	57.6	79.3
8	73.4	56.3	50.4	65.5	73.4	70.2
9	67.6	48.7	56.8	71.2	73.4	75.6
10	78.6	76.5	84.3	98.7	84.6	78.6
TCR	**10**	7	7	9	7	**10**

Table 2: The consistency ratio of different augmentation algorithms (TCR denotes the Total Consistency Ratio.)

challenging because the dynamic transitions in OFD are influenced not only by historical orders but also by uncertain factors, such as the meal preparation time of restaurants, traffic conditions and the personal circumstances of delivery riders. To verify whether RT can effectively address this challenge, we evaluate the impact of RT and other algorithms (i.e., ROMI, CABI, TATU and DStitch) on the performance of scheduling policy. Specifically, we collect one week of consecutive OFD scheduling data from Ankang City, averaging approximately 80,000 records per day. Then, we use 5 days of data as the training data and apply augmentation algorithms to randomly augment 30% of this data, while 2 days of data are used as the test data. This split ensures sufficient training volume while maintaining temporal consistency in the test data. Detailed data feature description and processing are provided in Appendix E.1.

Consistency ratio. Table 2 presents the consistency ratio (CR) under different OT learned by the scheduling policy on the augmented OFD dataset, where CR is defined as the proportion of cases in which the policy selection order is consistent with the true historical order, and OT refers to the number of orders a delivery rider takes in a time step. In Table 2, Original denotes the scheduling policy trained on original OFD data, and TCR represents the number of CR qualified for different OT types. In the online deployment requirements, the CR of the scheduling policy and existing platform policy is maintained between 65% and 90%. A low CR indicates a significant difference between the learned scheduling policy and the existing platform policy, which may lead to undesirable outcomes (e.g., assigning all orders to a single rider). A high CR suggests that the learned scheduling policy closely resembles the existing platform policy, hindering significant performance improvements.

The results demonstrate that RT successfully meets the online deployment requirements for all OTs and significantly outperforms other augmentation methods. Specifically, the CR of the ROMI method cannot meet the deployment requirements for OTs under 5, 8 and 9. Similarly, CABI and DStitch also exhibit a decrease in CR for OTs under 1, 4, 5, 6 and 9. While TATU introduces uncertainty to ensure that the generated data distribution is maintained around the original data, its CR for OT 2 is still below the online deployment requirement. These shortcomings stem from a common limitation: ROMI, CABI, TATU, and DStitch generate subsequent states, actions, and rewards solely based on current state-action pairs. Consequently, they cannot effectively capture historical sequence information, which is crucial for planning riders' new order pickup and delivery routes.

Cumulative returns. To test the performance improvement of ROMI, CABI, TATU, DStitch and RT on the scheduling policy, we assess the cumulative return (i.e., reward) of the scheduling policy in an OFD test system, where the reward at each time step is composed of platform revenue, rider revenue, and user benefits (see Appendix E.2 for detailed experiments). The experimental results show RT can effectively capture the dependency between historical orders and the current moment, and utilizes a reliability estimation mechanism to ensure the reliability of the generated trajectories.

Order quantity. An efficient scheduling algorithm (i.e., expert policy) should enable riders to accept the maximum number of orders without triggering order timeouts. In order to clearly understand the difference between generated trajectories and expert data, we calculate the in-transit order (i.e., the order that the rider has picked up but has not yet been delivered) quantity for each rider at every time step within these augmented trajectories and calculate the average in-transit order value for each trajectory. Experiment results indicate that the trajectories generated by RT contain more efficient order-matching data, which can effectively assist the scheduling policy to learn higher-performance order-matching behaviors (see Appendix E.3 for results).

Conclusion

In this paper, we propose a novel data-driven RL method RT to facilitate policy learning by utilizing reliable and high-reward trajectories generated by a dynamics model. RT leverages reliability-guaranteed sequence generation techniques and conditions the generation process on high rewards to generate high-reward trajectories that are legitimate in the environment. Extensive experiments demonstrate that RT can be effectively integrated with existing model-free algorithms and achieve better performance against existing offline RL benchmarks. However, RT presupposes a fully reversible state transition function in its experience evaluations. For environments with irreversible state transition functions, RT can only generate data using forward modeling, which could potentially impair its ability to produce high-reward trajectories. Future work includes addressing this issue and expanding RT to more real-world application scenarios, such as advertising bidding or recommendation systems based on RL.

Acknowledgments

We gratefully acknowledge the support from the Distinguished Young Scholars Project funded by the Natural Science Foundation of Guangdong Province (No. 2025B1515020060), the Basic and Applied Basic Research

Program of the Guangzhou Science and Technology Plan (No. 2025A04J7141), and Meituan.

References

Asadi, K.; Misra, D.; and Littman, M. L. 2018. Lipschitz Continuity in Model-based Reinforcement Learning. In *ICML*, volume 80 of *Proceedings of Machine Learning Research*, 264–273. PMLR.

Badrinath, A.; Flet-Berliac, Y.; Nie, A.; and Brunskill, E. 2023. Waypoint Transformer: Reinforcement Learning via Supervised Learning with Intermediate Targets. In *NeurIPS*, 78006–78027.

Bhardwaj, A.; Rothfuss, J.; Sukhija, B.; As, Y.; Hutter, M.; Coros, S.; and Krause, A. 2024. Data-Efficient Task Generalization via Probabilistic Model-Based Meta Reinforcement Learning. *IEEE Robotics Autom. Lett.*, 9(4): 3918–3925.

Blanchet, J. H.; Lu, M.; Zhang, T.; and Zhong, H. 2023. Double Pessimism is Provably Efficient for Distributionally Robust Offline Reinforcement Learning: Generic Algorithm and Robust Partial Coverage. In *NeurIPS*, 66845–66859.

Chen, L.; Lu, K.; Rajeswaran, A.; Lee, K.; Grover, A.; Laskin, M.; Abbeel, P.; Srinivas, A.; and Mordatch, I. 2021. Decision Transformer: Reinforcement Learning via Sequence Modeling. In *NeurIPS*, 15084–15097.

Chen, X.; Ghadirzadeh, A.; Yu, T.; Wang, J.; Gao, A. Y.; Li, W.; Bin, L.; Finn, C.; and Zhang, C. 2022. Lapo: Latent-variable advantage-weighted policy optimization for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 35: 36902–36913.

Diehl, C.; Sievernich, T.; Krüger, M.; Hoffmann, F.; and Bertram, T. 2021. UMBRELLA: Uncertainty-Aware Model-Based Offline Reinforcement Learning Leveraging Planning. *CoRR*, abs/2111.11097.

Fu, J.; Kumar, A.; Nachum, O.; Tucker, G.; and Levine, S. 2020. D4RL: Datasets for Deep Data-Driven Reinforcement Learning. *CoRR*, abs/2004.07219.

Fujimoto, S.; Meger, D.; and Precup, D. 2019. Off-Policy Deep Reinforcement Learning without Exploration. In *ICML*, volume 97 of *Proceedings of Machine Learning Research*, 2052–2062. PMLR.

Gao, C.; Wu, C.; Cao, M.; Kong, R.; Zhang, Z.; and Yu, Y. 2024. ACT: Empowering Decision Transformer with Dynamic Programming via Advantage Conditioning. In *AAAI*, 12127–12135. AAAI Press.

He, Z.; Wang, X.; Wang, R.; Shi, S.; and Tu, Z. 2022. Bridging the Data Gap between Training and Inference for Unsupervised Neural Machine Translation. In *ACL*, 6611–6623. Association for Computational Linguistics.

Ho, S.; and Yeung, R. W. 2010. The Interplay Between Entropy and Variational Distance. *IEEE Trans. Inf. Theory*, 56(12): 5906–5929.

Janner, M.; Fu, J.; Zhang, M.; and Levine, S. 2019. When to Trust Your Model: Model-Based Policy Optimization. In *NeurIPS*, 12498–12509.

Janner, M.; Li, Q.; and Levine, S. 2021. Offline Reinforcement Learning as One Big Sequence Modeling Problem. In *NeurIPS*, 1273–1286.

Kidambi, R.; Rajeswaran, A.; Netrapalli, P.; and Joachims, T. 2020. MOReL: Model-Based Offline Reinforcement Learning. In *NeurIPS*, 21810–21823.

Kostrikov, I.; Nair, A.; and Levine, S. 2022. Offline Reinforcement Learning with Implicit Q-Learning. In *ICLR*, 1–2052. OpenReview.net.

Kumar, A.; Zhou, A.; Tucker, G.; and Levine, S. 2020. Conservative Q-Learning for Offline Reinforcement Learning. In *NeurIPS*, 1179–1191.

Lee, J.; Yun, S.; Yun, T.; and Park, J. 2024. GTA: Generative Trajectory Augmentation with Guidance for Offline Reinforcement Learning. In *NeurIPS*.

Lee, K.; Nachum, O.; Yang, M.; Lee, L.; Freeman, D.; Guadarrama, S.; Fischer, I.; Xu, W.; Jang, E.; Michalewski, H.; and Mordatch, I. 2022. Multi-Game Decision Transformers. In *NeurIPS*, 27921–27936.

Li, G.; Shan, Y.; Zhu, Z.; Long, T.; and Zhang, W. 2024. DiffStitch: Boosting Offline Reinforcement Learning with Diffusion-based Trajectory Stitching. In *ICML*, 1342–1678. OpenReview.net.

Li, X.; and Yao, L. 2024. Distribution-Conditioned Adversarial Variational Autoencoder for Valid Instrumental Variable Generation. In *AAAI*, 13664–13672. AAAI Press.

Liang, Y.; Li, D.; Zhao, J.; Ding, X.; Lian, H.; Hao, J.; and He, R. 2023. Enhancing Dynamic On-demand Food Order Dispatching via Future-informed and Spatial-temporal Extended Decisions. In *CIKM*, 4702–4708. ACM.

Lu, C.; Ball, P. J.; Teh, Y. W.; and Parker-Holder, J. 2023. Synthetic Experience Replay. In *NeurIPS*, 46323–46344.

Lyu, J.; Li, X.; and Lu, Z. 2022. Double Check Your State Before Trusting It: Confidence-Aware Bidirectional Offline Model-Based Imagination. In *NeurIPS*, 38218–38231.

Sun, Y.; Zhang, J.; Jia, C.; Lin, H.; Ye, J.; and Yu, Y. 2023. Model-Bellman Inconsistency for Model-based Offline Reinforcement Learning. In *ICML*, volume 202 of *Proceedings of Machine Learning Research*, 33177–33194. PMLR.

Talvitie, E. 2017. Self-Correcting Models for Model-Based Reinforcement Learning. In Singh, S.; and Markovitch, S., eds., *AAAI*, 2597–2603. AAAI Press.

Torabi, F.; Warnell, G.; and Stone, P. 2018. Behavioral Cloning from Observation. In *IJCAI*, 4950–4957. ijcai.org.

Wagenmaker, A.; Shi, G.; and Jamieson, K. G. 2023. Optimal Exploration for Model-Based RL in Nonlinear Systems. In *NeurIPS*, 15406–15455.

Wang, J.; Li, W.; Jiang, H.; Zhu, G.; Li, S.; and Zhang, C. 2021. Offline Reinforcement Learning with Reverse Model-based Imagination. In *NeurIPS*, 29420–29432.

Yamagata, T.; Khalil, A.; and Santos-Rodríguez, R. 2023. Q-learning Decision Transformer: Leveraging Dynamic Programming for Conditional Sequence Modelling in Offline RL. In *ICML*, volume 202 of *Proceedings of Machine Learning Research*, 38989–39007. PMLR.

Yang, Q.; and Wang, Y.-X. 2025. RTDiff: Reverse Trajectory Synthesis via Diffusion for Offline Reinforcement Learning. In *ICLR*.

Yu, T.; Kumar, A.; Rafailov, R.; Rajeswaran, A.; Levine, S.; and Finn, C. 2021. COMBO: Conservative Offline Model-Based Policy Optimization. In *NeurIPS*, 28954–28967.

Yu, T.; Thomas, G.; Yu, L.; Ermon, S.; Zou, J. Y.; Levine, S.; Finn, C.; and Ma, T. 2020. MOPO: Model-based Offline Policy Optimization. In *NeurIPS*, 14129–14142.

Zhan, X.; Zhu, X.; and Xu, H. 2022. Model-Based Offline Planning with Trajectory Pruning. In Raedt, L. D., ed., *IJCAI*, 3716–3722. ijcai.org.

Zhang, J.; Lyu, J.; Ma, X.; Yan, J.; Yang, J.; Wan, L.; and Li, X. 2023. Uncertainty-Driven Trajectory Truncation for Data Augmentation in Offline Reinforcement Learning. In *ECAI*, volume 372 of *Frontiers in Artificial Intelligence and Applications*, 3018–3025. IOS Press.

Zhao, R.; Zhang, L.; Fu, B.; Hu, C.; Su, J.; and Chen, Y. 2024. Conditional Variational Autoencoder for Sign Language Translation with Cross-Modal Alignment. In *AAAI*, 19643–19651. AAAI Press.