Offline Trajectory Optimization for Offline Reinforcement Learning

Ziqi Zhao
Shandong University
Qingdao, China
ziqizhao.work@gmail.com

Zhaochun Ren
Leiden University
Leiden, The Netherlands
z.ren@liacs.leidenuniv.nl

Liu Yang
Shandong University
Qingdao, China
yangliushirry@gmail.com

Yunsen Liang
Shandong University
Qingdao, China
yunsenliang@mail.sdu.edu.cn

Fajie Yuan
Westlake University
Hangzhou, China
yuanfajie@westlake.edu.cn

Pengjie Ren
Shandong University
Qingdao, China
renpengjie@sdu.edu.cn

Zhumin Chen
Shandong University
Qingdao, China
chenzhumin@sdu.edu.cn

Jun Ma
Shandong University
Qingdao, China
majun@sdu.edu.cn

Xin Xin*
Shandong University
Qingdao, China
xinxin@sdu.edu.cn

Abstract

Offline reinforcement learning (RL) aims to learn policies without online explorations. To enlarge the training data, model-based offline RL learns a dynamics model which is utilized as a virtual environment to generate simulation data and enhance policy learning. However, existing data augmentation methods for offline RL suffer from (i) trivial improvement from short-horizon simulation; and (ii) the lack of evaluation and correction for generated data, leading to low-qualified augmentation.

In this paper, we propose **o**ffline **t**rajectory op**t**imization for **o**ffline reinforcement learning (OTTO). The key motivation is to conduct long-horizon simulation and then utilize model uncertainty to evaluate and correct the augmented data. Specifically, we propose an ensemble of Transformers, a.k.a. World Transformers, to predict environment state dynamics and the reward function. Three strategies are proposed to use World Transformers to generate long-horizon trajectory simulation by perturbing the actions in the offline data. Then, an uncertainty-based World Evaluator is introduced to firstly evaluate the confidence of the generated trajectories and then perform the correction for low-confidence data. Finally, we jointly use the original data with the corrected augmentation data to train an offline RL algorithm. OTTO serves as a plug-in module and can be integrated with existing model-free offline RL methods. Experiments on various benchmarks show that OTTO can effectively improve the performance of representative offline RL algorithms, including in complex environments with sparse rewards like AntMaze. Codes are available at https://github.com/ZiqiZhao1/OTTO.

CCS Concepts

• **Computing methodologies** → **Reinforcement learning**.

Keywords

Reinforcement learning, Offline reinforcement learning, Environment model, Transformers

*Corresponding author.

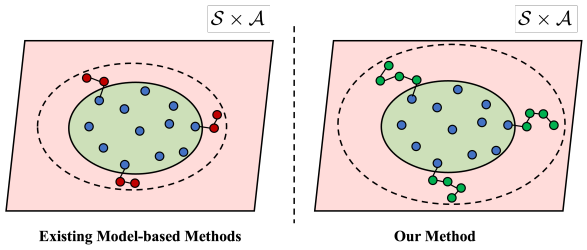

Figure 1: Comparison between existing model-based methods and OTTO. Blue dots refer to the original offline dataset. Red dots on the left represent the short-horizon rollouts from existing model-based methods, resulting in small generalization improvement near the original data. Green dots on the right represent the long-horizon trajectories generated by OTTO, which lead to a broader generalization range.

1 Introduction

Offline reinforcement learning (RL) refers to training RL agents from pre-collected datasets without real-time interactions or online explorations on the true environment [22]. This paradigm plays a crucial role in practical scenarios where collecting online interactions can be expensive or risky, such as healthcare [24, 45], autonomous driving [53], robotics [15, 29, 35, 40] and recommender systems [43, 51]. Standard RL methods often fail in such offline setting due to erroneous estimation of value functions [10].

Existing offline RL methods can be classified into two categories: model-free methods and model-based methods. Model-free offline methods incorporate conservatism into the value function estimation [9, 10, 18, 20, 21, 39]. For example, CQL [21] adds a regularization term into the Q-function update. The conservatism downgrades the value function for unseen states and thus helps to avoid the over-estimation issue. As a result, the learned policy is constrained to near the offline data distribution. Besides, the overly conservatism could lead to underestimation of potentially good actions.

Such issues lead to the poor generalization capability of model-free offline RL methods.

To enhance the generalization capability of offline RL, a viable solution is to generate additional training data. Model-based offline RL algorithms learn a dynamics model based on the offline dataset, which is then utilized as a simulated environment to generate new data [16, 54, 55]. However, we found that the augmented data generated by existing methods is of low qualification, resulting in trivial improvement for policy learning. Specifically, most existing model-based RL approaches only perform short-horizon model rollouts, resulting in marginal generalization improvement only near the offline data, as shown in Figure 1. Besides, when we perform the simulation of long-horizon trajectories using the environment model, the average reward of each interaction step often becomes lower for longer steps, as shown in Figure 2(a). As a result, the performance of the learned policy suffers a sharp decline, as shown in Figure 2(b). Additionally, although there exist some works utilizing the powerful generative model to augment the offline data, these data augmentation methods lack the evaluation and correction for the generated trajectories. Consequently, the augmented trajectories of existing methods could be unreliable, leading to sub-optimal performance. To summarize, existing data augmentation methods for offline RL suffer from (1) trivial improvement from short-horizon rollout simulation and (2) the lack of evaluation and correction for generated trajectories.

To address the aforementioned issues, we propose **o**ffline **t**rajectory op**t**imization for **o**ffline reinforcement learning (**OTTO**). Figure 2(c) demonstrates the framework of OTTO. To address the first issue, we propose to learn the state dynamics transition and the reward function through an ensemble of Transformers [48], a.k.a. World Transformers, based on the offline dataset. To reduce the computational cost, we employ a cyclic annealing schedule to reset the environment model's learning rate and preserve model snapshots, thereby obtaining an ensemble of environment models from a single training process. Then, three strategies are proposed to use World Transformers to generate long trajectory simulation by perturbing the actions in offline data. The generalization and long sequence modeling capability of World Transformers, together with the prior knowledge in the offline dataset, ensure that the generated long-horizon trajectories are high-reward and thus help the model to learn potential good actions. To address the second issue, we propose an uncertainty-based World Evaluator to conduct the evaluation and correction for generated trajectories. The main idea is that if World Transformers are not confident about the state transition and reward, a conservative punishment is introduced to correct the simulated trajectories to avoid potential overestimation. Finally, original offline data together with the corrected augmentations are jointly used to train an offline RL algorithm. Note that OTTO serves as a plug-in module and can be integrated with a wide range of model-free offline RL methods to further enhance the learned policy. The main contributions are as follows:

- We propose World Transformers, which utilize an ensemble of Transformers to learn state dynamics transition and reward functions with good generalization and long sequence modeling capability.
- We propose three trajectory simulation strategies, which generate high-qualified long-horizon trajectory simulations by using the World Transformers.
- We propose an uncertainty-based World Evaluator to evaluate and correct the generated trajectories, which further enhances the reliability of the augmented data.
- Experiments on various benchmarks show that OTTO can effectively improve the performance of representative offline RL algorithms, including in complex environments with sparse rewards like AntMaze.

2 Related Work

Offline RL aims to learn policies from a pre-collected, static dataset of pre-collected trajectories. It is important to various domains in which online explorations are expensive to collect, e.g., healthcare [24, 46], autonomous driving [53, 58], NLP [13, 14], and recommender systems [5, 43].

2.1 Model-free offline RL

Existing model-free offline RL methods are mainly based on restricting the target policy to be near the behavior policy [8, 10, 19, 39, 50], including incorporating value pessimism [18, 21, 52], importance sampling [25, 32, 42], and uncertainty punishment for value functions [1, 20, 41]. Recently, Chen et al. [4], Janner et al. [12] proposed to cast offline RL into a sequence modeling problem and use Transformers to solve it in a supervised manner. However, these methods are limited to the support of offline data distribution. Since the state distribution for practical usage might differ from that of the behavior policy, model-free offline RL encounters the distributional shift issue, leading to sub-optimal performance when evaluating in real environments. In this paper, we propose to use World Transformers and World Evaluator to augment high-qualified data and further enhance the performance of these existing methods.

2.2 Data augmentation for offline RL

Existing model-based offline RL methods [16, 26, 30, 35, 37, 38, 44, 54–56] learn a model of environment and generate synthetic rollouts to improve the policy. However, we found the augmented rollouts generated by these methods are extremely short-horizon (mostly with simulation length $h = 1$), resulting in limited generalization improvement. Although BooT[49] uses a Transformer to model the environment and generate data for bootstrapping, it still can only generate short trajectories with length $h = 1$ in most cases and fails to integrate effectively with other methods. On the contrary, the proposed OTTO aims to generate long-horizon trajectories with high quality through World Transformers, providing more potential for better generalization.

Besides, some works propose to utilize the powerful diffusion-based model to augment the offline data [23, 27]. For instance, SER [27] learns the environment transition tuples and samples new tuples through diffusion models to augment the data. DiffStitch [23] selects a low-reward trajectory and a high-reward trajectory, then generates a sub-trajectory to stitch them together by the diffusion model. However, these data augmentation methods lack the evaluation and correction for the generated trajectories. Consequently, the augmented trajectories of existing methods could be unreliable.

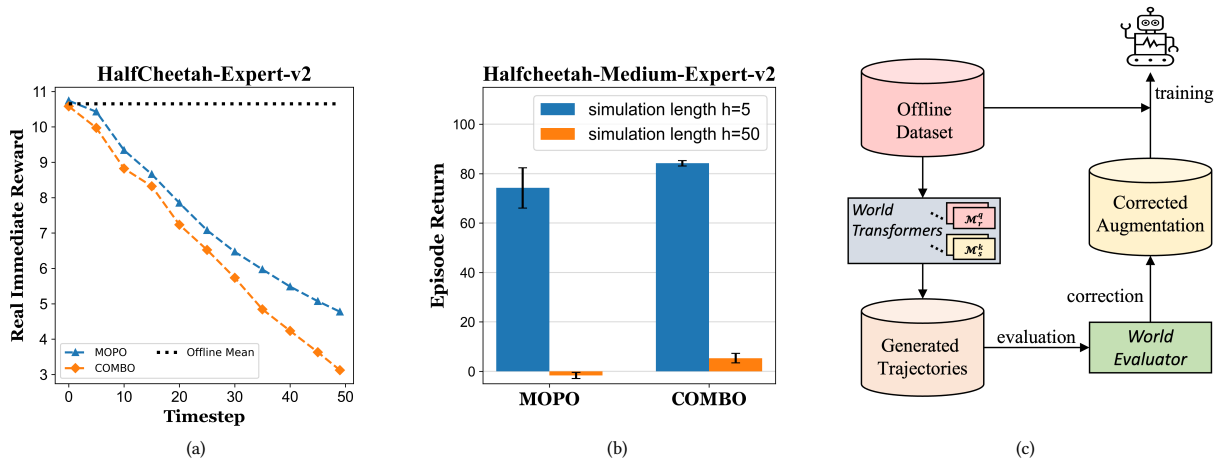

Figure 2: (a) Average reward of each step. The average immediate reward of each interaction step in long simulation trajectories generated by two representative model-based methods MOPO [55] and COMBO [54]. The reward becomes lower for longer steps. (b) Policy performance of MOPO and COMBO with short and long simulations over 5 different seeds. Long simulation with lower reward downgrades the performance. (c) The framework of the proposed OTTO.

In this paper, the proposed OTTO introduces a World Evaluator to evaluate the uncertainty of generated trajectories and further uses a conservative punishment to correct low-confidence trajectories.

Some online RL works [3, 31, 57] also use Transformers to model the environment, but these works are primarily focused on utilizing Transformers to improve sampling efficiency, and they cannot be directly applied to offline RL. In contrast, our work leverages Transformers to improve the generalization capability of offline RL by generating long-horizon trajectories. The problem we aim to solve is fundamentally different from that of online RL.

3 Methodology

This section presents the details of OTTO. We first describe the task in section 3.1. Then, the World Transformers are described in section 3.2. The long-horizon trajectory generation is described in section 3.3. Finally, the World Evaluator is described in section 3.4.

3.1 Task description

A Markov Decision Process (MDP) is defined by the tuple $< \mathcal{S}, \mathcal{A}, T, r, \mu_0, \gamma >$, where $\mathcal{S}, \mathcal{A}$ refer to state space and action space, respectively. A policy $\pi(a|s)$ defines a mapping from state $s \in \mathcal{S}$ to a probability distribution over action $a \in \mathcal{A}$. Given a (s, a) pair, $T(s'|s, a)$ and $r(s, a)$ represent the distribution of the next state s' and the obtained immediate reward, respectively. μ_0 is the initial state distribution and $\gamma \in (0, 1)$ is the discount factor for future reward. The goal of RL is to find an optimal policy $\pi^* = \text{argmax}_\pi \mathop{\mathbb{E}}_{\pi,T}[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)|s_0 \sim \mu_0]$ that maximizes the expected cumulative reward. t denotes the interaction step.

Conventional RL performs online exploration through sampling (s, a) according to the target policy π, which could be expensive under practical usage. To address this issue, offline RL aims to learn a policy from a static dataset $\mathcal{D}$, which contains trajectories $\tau = \{s_t, a_t, r_t\}_{t=0}^{|\tau|}$ pre-collected with an unknown behavior policy. However, offline RL algorithms often fail to tackle unseen states and actions during online inference in a real environment. In this work, we aim to improve the performance of offline RL through generating high-qualified long-horizon simulated trajectories, which enlarge the observed states and actions during model training. $\mathcal{D}_{aug}$ is used to denote the augmented dataset. $\delta = |\mathcal{D}_{aug}|/|\mathcal{D}|$ is used to denote the augmentation ratio. The process for trajectory augmentation of OTTO is visualized in Figure 3.

3.2 World transformers

To perform trajectory augmentation, an environment simulator needs to be implemented to model the state transition distribution and the reward function. We propose to use Transformers to build the simulator, a.k.a. World Transformers. Precisely, World Transformers consist of State Transformers and Reward Transformers to predict the next state and reward, respectively. It has been shown that Transformers can effectively learn from long input sequences and produce promising outputs [2, 4, 6, 36].

Model inputs. The key factors to construct the environment model inputs are: (1) the input should contain all the factors determining the environment dynamics; (2) the input should enable World Transformers to learn from continuous sequential interactions; (3) the input should be as simple as possible to avoid unnecessary noise. Considering that the environment provides feedback simply based on the agent's action and the current state, the model input for World Transformers contains multiple continuous state and action tuples (s, a). Formally, we define the model input as

$$inputs = \{\cdots, s_t, a_t, s_{t+1}, a_{t+1}, \cdots\} \tag{1}$$

Both State Transformers and Reward Transformers take the above inputs for environment simulation.

Environment modeling. We use State Transformers and Reward Transformers to model the next state transition and the reward function, respectively. Specifically, State Transformers are defined as $\mathbf{M}_s = \{\mathcal{M}_s^k | k \in \{1, 2, ...K\}\}$, and Reward Transformers are defined as $\mathbf{M}_r = \{\mathcal{M}_r^q | q \in \{1, 2, ...Q\}\}$. K and Q are the number of

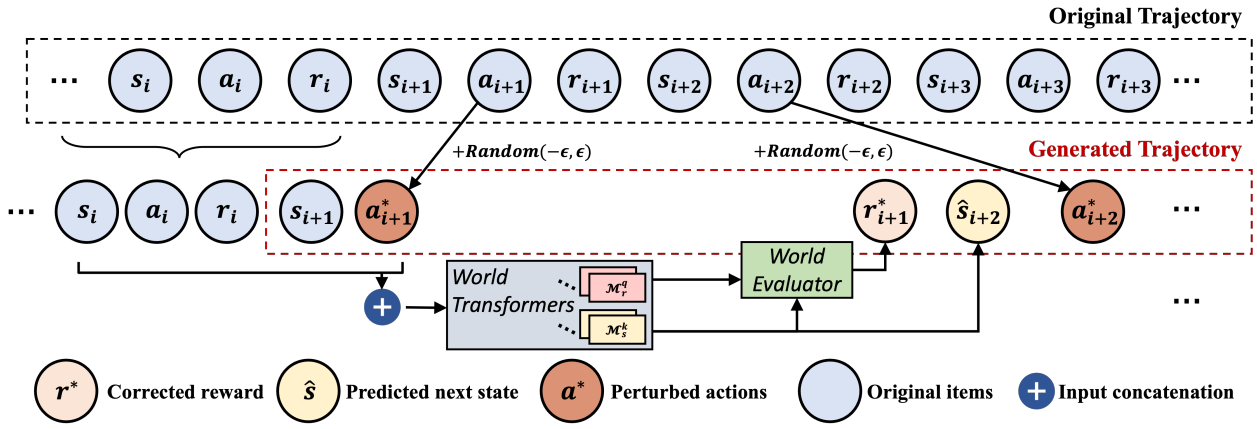

Figure 3: Trajectories augmentation of OTTO. Given an original trajectory and the augmentation step $t = i + 1$, random noise of range $(-\epsilon, \epsilon)$ is introduced to perturb a_{i+1} and forms a new action a^*_{i+1}. Then the corresponding r^*_{i+1} and $\hat{s}_{i+2}$ can be inferred from World Transformers and World Evaluator. This process is repeated for h times to generate the new trajectory for augmentation.

models, respectively. $\mathcal{M}^k_s$ and $\mathcal{M}^q_r$ are the k-th State Transformer and the q-th Reward Transformer. We introduce a learnable embedding matrix to represent the interaction step t, which is similar to the positional embedding in language modeling. Noticing that GPT-series models have achieved remarkable success across various NLP tasks [2, 33, 34], we use the similar architecture to model the states transition and reward function. At timestep t, $\mathcal{M}^k_s$ and $\mathcal{M}^q_r$ model the following distributions:

$$\hat{s}^k_{t+1} \sim p_{\mathcal{M}^k_s}(\hat{s}_{t+1} | \{s, a\}^t_{t-L_s+1}) \tag{2}$$

$$\hat{r}^q_t \sim p_{\mathcal{M}^q_r}(\hat{r}_t | \{s, a\}^t_{t-L_r+1}), \tag{3}$$

where $\{s, a\}^t_{t-L+1}$ represents the model input sequence $\{s_{t-L+1}, a_{t-L+1}, s_{t-L+2}, a_{t-L+2}, ..., s_t, a_t\}$ as shown in Eq.(1). L_s, L_r represent the hyperparameters that define the length of the model inputs for $\mathcal{M}^k_s, \mathcal{M}^q_r$ respectively. We also tried to model both distributions in one unified Transformer, but no better performance was observed. Both $\mathcal{M}^k_s$ and $\mathcal{M}^q_r$ are trained in a supervised manner upon the offline dataset. We use mean-squared error for both the next state prediction and reward prediction.

Cyclic annealing schedule. To further enhance the stability, an ensemble of environment models is involved to generate the simulation. Most existing ensemble-based methods independently train individual models by setting different initial random seeds, which can be computationally resource-consuming[11, 16, 28]. In contrast, we adopt a training efficiency approach to obtain an ensemble of environment models from a single training process. Specifically, we divide the training process into multiple cycles. Each cycle begins with a relatively large learning rate, which is subsequently gradually reduced to a smaller value through annealing. Learning rate decay enables the model to rapidly converge to a local minimum, while a larger learning rate allows the model to escape from a critical point. Besides, to stabilize training and improve convergence, we initially employed the warmup technique before the first cycle. The cyclic annealing learning rate schedule can be defined as:

$$\alpha(t) = \begin{cases} \frac{(t+1)\alpha_0}{T_{\text{wp}}} & \text{if} \quad t < T_{\text{wp}} \\ \frac{\alpha_0}{2}\left(1 + \cos\left(\pi \cdot \frac{(t - T_{\text{wp}}) \mod T_{\text{cyc}}}{T_{\text{cyc}}}\right)\right) & \text{if} \quad t \geq T_{\text{wp}} \end{cases} \tag{4}$$

where T_{wp} and T_{cyc} are hyperparameters representing the number of warmup steps and steps in each cycle, respectively. At the end of each training cycle, we save the model weights as an individual model of the ensemble. Once a predefined number of environment models have been obtained, the state and reward prediction is formulated as the mean of the ensemble: $\hat{s}_{t+1} = \overline{\hat{s}^k_{t+1}}$, and $\hat{r}_t = \overline{\hat{r}^q_t}$.

3.3 Trajectory generation

We now discuss how to generate trajectories with World Transformers. The key idea in trajectory generation is to expand the observed state-action tuples, thereby improving the performance of offline RL. Motivated by this, we introduce uniform noise to the action a_t in the original trajectory, forming a perturbed action a^*_t. In particular, we add a random number of range $(-\epsilon, \epsilon)$ to each dimension of a_t. Then we predict the corresponding next state and reward with World Transformers. The alteration of action results in changes to the next state, therefore a new tuple $\{a^*_t, \hat{r}_t, \hat{s}_{t+1}\}$ is generated. We repeat this process of introducing noise to the original action and predicting the next state and reward from timestep $t = t_s$ to $t = t_s + h - 1$, where h defines the length of simulation.

Generating simulation for all timesteps in the offline dataset is not appropriate since (1) this may introduce extra noise and (2) not every tuple $\{s_t, a_t, r_t\}$ in a trajectory is worthy of augmenting. We prefer to generate trajectories with high total rewards, which are good demonstrations for policy improvement. In this paper, three trajectory generation strategies are proposed:

Random : For each trajectory $\tau = \{s_t, a_t, r_t\}^{|\tau|}_{t=0}$, we randomly select the start timestep $t_s \in [0, |\tau| - h + 1]$, and generate trajectory τ^* based on the trajectory segment $\{s_t, a_t, r_t\}^{t_s+h-1}_{t=t_s}$.

Top-N : We first split all original trajectories into segments of length h. Then we sort these segments by their cumulative rewards in descending order and generate trajectories based on the Top-N

segments, where $N = |\mathcal{D}_{aug}|/h$.

Softmax : We also split the trajectory segments as the Top-N strategy, but we choose the segments according to their probabilities calculated by a softmax function over segment cumulative reward.

3.4 World evaluator

Although the high-reward trajectories could be generated by the manually designed strategies in section 3.3, the generated trajectories might be overestimated, especially when World Transformers have high uncertainty regarding their predictions. This overestimation might also mislead the policy learning, resulting in sub-optimal performance. To address this issue, the World Evaluator is introduced to evaluate the confidence of generated trajectories and give a conservative punishment for trajectories with high uncertainty.

The uncertainty evaluation is based on variances of both state predictions $\hat{s}^k_{t+1}$ and reward predictions $\hat{r}^q_t$. Specifically, for each tuple $(a^*_t, \hat{r}_t, \hat{s}_{t+1})$ in generated trajectories, we calculate the standard deviation, i.e., σ^s_{t+1} and σ^r_t, over $\hat{s}^k_{t+1}$ and $\hat{r}^q_t$. Higher deviation denotes higher uncertainty about trajectory generation, and thus the predicted reward should be downgraded. Then we correct $\hat{r}_t$ as:

$$r^*_t = (1 - \frac{\mathrm{e}^{(\sigma^s_{t+1}/\omega)}}{\sum_{t=t_s}^{t_s+h-1} \mathrm{e}^{(\sigma^s_{t+1}/\omega)}})(\hat{r}_t - \sigma^r_t), \tag{5}$$

where ω is a hyperparameter representing the temperature. $\hat{r}_t - \sigma^r_t$ ensures a lower bound of the predicted reward and $1 - \frac{\mathrm{e}^{(\sigma^s_{t+1}/\omega)}}{\sum_{t=t_s}^{t_s+h-1} \mathrm{e}^{(\sigma^s_{t+1}/\omega)}}$ evaluates the confidence level of the predicted next states. If World Transformers are not confident about the state transition and reward, a conservative punishment will be introduced to downgrade the predicted reward. Finally, the tuple $(a^*_t, r^*_t, \hat{s}_{t+1})$ is augmented to enlarge the offline data. The whole process for trajectory generation and correction is visualized in Figure 3. After data augmentation, an offline RL algorithm is trained upon the augmented dataset to gain better policy performance.

4 Experiment

In this section, we conduct experiments to answer the following research questions: (1) Can OTTO effectively improve the performance of existing offline model-free RL methods? (2) How does OTTO perform compared to state-of-the-art data augmentation methods for offline RL? (3) How does the design of OTTO affect the policy performance?

Baselines. To answer question (1), we integrate OTTO with various representative model-free offline RL approaches and compare the policy performance with or without OTTO. These model-free offline RL approaches include DT [4], TD3+BC [8], CQL [21] and IQL [19]. DT casts offline RL into a sequence modeling problem. TD3+BC is an imitation learning method, which jointly learns the policy and value function. CQL and IQL are both based on TD learning, where CQL is a multi-step dynamic programming method and IQL is a one-step method. Besides, we also compare OTTO with three representative model-based offline RL approaches (COMBO [54], BooT [49] and MOPO [55]). To answer question (2), we compare OTTO with two recently proposed diffusion-based augmentation methods, i.e., SER [27] and DStitch [23].

Datasets. We evaluate our methods on a wide range of datasets in Gym MuJoCo and AntMaze tasks from D4RL benchmark[7]. Specifically, we use three different environments in Gym MuJoCo domains (*HalfCheetah*, *Hopper*, *Walker2d*), each with three datasets (*Medium*, *Medium-Replay*, *Medium-Expert*). We also evaluate the performance in AntMaze tasks, which aim to control a robot and navigate to reach a goal. There are three different size of maze (*umaze*, *medium*, *large*) and different dataset types (*fixed*, *play*, *diverse*). In AntMaze tasks, the agent receives a sparse reward only if the goal is reached. Besides, it has a larger state space (27 dimensions) and action space (8 dimensions), making it more challenging compared to Gym MuJoCo tasks.

Implementation details. For World Transformers, the number of State Transformers K the number of Reward Transformers Q are both set to 4. The interaction steps of input for the State Transformers L_s and Reward Transformers L_r are both set to 20. For trajectory generation, the horizon of generation is set to 50 for both MuJoCo and AntMaze tasks. The temperature ω of World Evaluator is set to 0.7. We set the augmentation ratio δ ranging from 5% to 20% depending on the dataset. The number of warmup steps T_{wp} and the number of steps in each cycle T_{cyc} are set to 10^5 and 5×10^5, respectively. We need four cycles in total to get four different models. The full experimental details are provided in Appendix A.

4.1 Performance with OTTO augmentation

To answer the RQ1, we evaluate the performance of model-free offline RL methods with the integration of OTTO on both standard MuJoCo tasks and challenging AntMaze tasks.

Evaluation on MuJoCo tasks. Table 1 shows the results of four representative model-free offline RL methods (IQL, TD3+BC, DT, CQL) on MuJoCo environment. We also include three representative model-based offline RL approaches (COMBO, BooT, MOPO) for a more comprehensive comparison. From Table 1 we can observe that: (1) OTTO significantly outperforms the original scores in all settings, achieving consistent improvement in MuJoCo average score among four representative model-free methods. This suggests that OTTO is robust to different datasets and can generalize to multiple model-free algorithms. (2) OTTO achieves significant improvement in the datasets where the original score is relatively low such as *Hopper-Medium* datasets. This observation suggests that OTTO can potentially generate good demonstrations in datasets that do not include expert data. (3) The performance of COMBO and BooT was initially superior to that of model-free RL algorithms. After the augmentation of OTTO, two of four model-free RL algorithms (IQL and TD3+BC) outperform COMBO and BooT and the remaining two methods achieve comparable results. All four model-free RL algorithms achieve better performance than MOPO after OTTO augmentation. This indicates that the long-horizon high-confidence trajectories generated by OTTO provide more significant generalization improvement than the short-horizon rollouts generated by existing model-based methods.

Evaluation on AntMaze tasks. We also conduct experiments on the challenging AntMaze tasks and present the results in Table 2. We can observe that OTTO improves the policy performance in most cases, and obtains the best average score for both IQL and

Table 1: Performance comparison when integrating OTTO with existing offline RL methods on Gym MuJoCo-v2. The normalized scores are computed over 5 random seeds. Med-Exp represents *medium-expert* and Med-Rep represents *medium-replay*. Boldface denotes the improved performance after integration.

	Environment	IQL		TD3+BC		DT		CQL		COMBO	BooT	MOPO
		Original	**OTTO**	Original	**OTTO**	Original	**OTTO**	Original	**OTTO**			
Med-Exp	HalfCheetah	92.1±1.2	**93.8**±0.6	93±1.2	**95.6**±0.3	87.7±1.4	**92.8**±1.9	89.9±2.8	**91.4**±0.5	90	94	63.3
	Hopper	102±5.6	**113.4**±0.3	101.6±2.9	**110.6**±1.4	105.3±6	**110.3**±0.8	99.1±4.3	**107.1**±1.3	111.1	102.3	23.7
	Walker2d	111.4±0.5	**112.9**±0.1	109.4±0.1	**110.9**±0.1	108.1±0.7	**108.9**±0.3	109.2±0.1	**110.5**±0.1	103.3	110.4	44.6
Medium	HalfCheetah	48.5±0.3	**49.3**±0.1	48.5±0.1	**49.9**±0.2	41.4±0.1	**44.6**±0.2	46.8±0.1	**48.1**±0.1	54.2	50.6	42.3
	Hopper	61.2±4.7	**78.6**±3.5	60.6±1	**74.5**±3.7	64.9±4.6	**74.2**±1.3	58.6±0.8	**62.2**±0.7	97.2	70.2	28
	Walker2d	80.4±0.1	**83.5**±1.5	81.8±0.6	**83.7**±0.1	73.9±2.7	**77.1**±0.1	81.3±0.4	**82.7**±0.3	81.9	82.9	17.8
Med-Rep	HalfCheetah	43.8±0.2	**44.8**±0.1	44.9±0.2	**45.8**±0.2	35.5±0.4	**38.8**±0.5	45.4±0.3	**46.3**±0.2	55.1	46.5	53.1
	Hopper	94.3±4.6	**102.4**±1	72.8±4.9	**80.8**±13.2	71.2±3.6	**89.5**±5	88.4±7.6	**92.9**±15.4	89.5	92.9	67.5
	Walker2d	86.2±3.6	**86.9**±2	87.3±1.9	**90.7**±0.3	70.2±1.2	**75.8**±2.6	78.7±1.1	**91.1**±0.4	56.0	87.6	39
MuJoCo Average		80	**85.1**	77.8	**82.5**	73.1	**79.1**	77.5	**81.4**	82	81.9	42.1

Table 2: The average score of AntMaze. The normalized scores are computed over 5 random seeds. Boldface denotes the highest score.

Dataset	IQL		TD3+BC	
	Original	OTTO	Original	OTTO
umaze	**73.6**±2.8	73±2.1	93.1±1.4	**95.7**±2.8
umaze-diverse	58±2	**60.4**±2.3	40.5±3.9	**53.3**±5.4
medium-play	67.3±1.9	**72.4**±3.2	0.4±0.2	**2.1**±0.2
medium-diverse	**68.4**±1.8	67.8±2.5	0.3±0.1	**0.5**±0.1
large-play	41±7.4	**43.2**±3.3	0±0	0±0
large-diverse	31.2±2.6	**36.8**±2.1	0±0	0±0
Antmaze Average	56.6	**58.9**	22.3	**25.3**

TD3+BC. AntMaze is a sparse-reward environment that requires "stitching" parts of sub-optimal trajectories [19]. This indicates that OTTO can generalize between different states and generate trajectories from different start points to the goal point. This observation shows the effectiveness of OTTO in complex environments.

To summarize, OTTO effectively improves the performance of various existing model-free RL algorithms across different environments including complex environments like AntMaze.

4.2 Comparison with augmentation methods

To answer RQ2, we compared OTTO with the state-of-the-art diffusion-based data augmentation algorithms[23, 27].

Table 3 shows the comparison between OTTO and state-of-the-art diffusion-based data augmentation baselines (SER, DStitch) on Gym MuJoCo tasks. The result of SER for DT is omitted since SER cannot be utilized for sequence modeling-based DT [23]. From Table 3 we can observe that OTTO achieves better scores than SER and DStitch in 18 out of 27 settings, and achieves comparable results in the remaining settings. The reason is that although SER and DStitch utilize the powerful diffusion model to generate trajectory augmentation, they do not conduct the evaluation and correction for generated trajectories. On the contrary, OTTO introduces an uncertainty-based World Evaluator to evaluate the generated data and uses a conservative punishment to correct low-confidence trajectories, leading to higher-confidence augmentation. This observation verifies the effectiveness of the World Evaluator.

4.3 Method investigation

4.3.1 Effect of trajectory generation strategies. In this section, we evaluate the three trajectory generation strategies on Gym MuJoCo tasks. Table 4 summarizes the experimental results for IQL and DT. Experiment results in Table 4 show that all of the three strategies can outperform the original performance, which demonstrates that all of these strategies can benefit the existing offline RL algorithms.

In addition, for each strategy, we have the following key observations: (1) The Random strategy exhibits lower standard deviation, indicating that augmenting trajectories through random selection preserves the distribution of the augmented trajectories in alignment with the original dataset. As a result, this strategy produces more stable outcomes and demonstrates reduced standard deviation. (2) The Top-N strategy achieves better performance when there are expert data in the dataset (*Medium-Expert*), demonstrating its ability to effectively select high-reward trajectories for augmentation. (3) The Softmax strategy achieves the best score in most cases, demonstrating that it not only generates high-reward trajectories but also enhances robustness. Based on the aforementioned observations, we provide guidance on how to select an appropriate strategy according to the given environment. To ensure a lower bound on performance, the Random strategy can be selected to obtain relatively stable outputs, as it exhibits the smallest standard deviation. If expert data is present in the environment, the Top-N strategy should be chosen to better leverage the expert knowledge. In other cases, or when facing an unknown environment, the Softmax strategy is recommended.

4.3.2 Ablation study. To investigate the impact of World Transformers and World Evaluator, we use IQL to conduct an ablation

Table 3: Comparison with diffusion-based augmentation baselines. The normalized scores are computed over 5 random seeds. Boldface denotes the highest score.

Dataset	Environment	IQL			TD3+BC			DT	
		SER	DStitch	**OTTO**	SER	DStitch	**OTTO**	DStitch	**OTTO**
Med-Expert	HalfCheetah	88.9	**94.4**	93.8±0.6	86.5	**96**	95.6±0.3	92.6	**92.8**±1.9
Med-Expert	Hopper	110.4	110.9	**113.4**±0.3	104	107.1	**110.6**±1.4	109.4	**110.3**±0.8
Med-Expert	Walker2d	111.7	111.6	**112.9**±0.1	110.5	110.2	**110.9**±0.1	108.6	**108.9**±0.3
Medium	HalfCheetah	49.3	**49.4**	49.3±0.1	48.4	**50.4**	49.9±0.2	44.2	**44.6**±0.2
Medium	Hopper	66.6	71	**78.6**±3.5	56.4	60.3	**74.5**±3.7	60.5	**74.2**±1.3
Medium	Walker2d	**85.9**	83.2	83.5±1.5	**84.9**	83.4	83.7±0.1	72	**77.1**±0.1
Med-Replay	HalfCheetah	46.6	44.7	**44.8**±0.1	45.2	44.7	**45.8**±0.2	**41**	38.8±0.5
Med-Replay	Hopper	102.4	102.1	**102.4**±1	56.8	79.6	**80.8**±13.2	**96.1**	89.5±5
Med-Replay	Walker2d	85.7	86	**86.9**±2	89.1	89.7	**90.7**±0.3	60.2	**75.8**±2.6
MuJoCo Average		83.1	83.8	**85.1**	75.8	80.2	**82.5**	76.1	**79.1**

Table 4: Results of IQL and DT with three trajectory generation strategies of OTTO. The normalized scores are computed over 5 random seeds. Boldface denotes the highest score.

Dataset	Environment	IQL				DT			
		Original	Random	Top-N	Softmax	Original	Random	Top-N	Softmax
Med-Expert	HalfCheetah	92.1±1.2	93.3±0.3	**93.8**±0.6	91.1±0.3	87.7±1.4	91.4±0.9	**92.8**±1.9	92.5±1.8
Med-Expert	Hopper	102±5.6	111.3±0.2	112.6±0.8	**113.4**±0.3	105.3±6	109.7±0.4	**110.3**±0.8	110.2±1.4
Med-Expert	Walker2d	111.4±0.5	112.5±0.1	112.8±0.3	**112.9**±0.1	108.1±0.7	108.6±0.1	**108.9**±0.3	108.5±0.2
Medium	HalfCheetah	48.5±0.3	48.6±0.1	48.5±0.3	**49.3**±0.1	41.4±0.1	43.5±0.1	44±1.1	**44.6**±0.2
Medium	Hopper	61.2±4.3	72.3±2.7	69.7±4.6	**78.6**±3.5	64.9±4.6	71±1.5	69.4±2.4	**74.2**±1.3
Medium	Walker2d	80.4±0.1	82.3±0.4	82.1±1.8	**83.5**±1.5	73.9±2.7	**77.1**±0.1	77±1.2	76.9±1.3
Med-Replay	HalfCheetah	43.8±0.2	44.7±0.1	44.3±0.1	**44.8**±0.1	35.5±0.4	37±0.6	**38.8**±0.5	38.6±0.7
Med-Replay	Hopper	94.3±4.6	100.4±1.2	99.8±2.3	**102.4**±1	71.2±3.6	87.7±3.4	89.4±4.6	**89.5**±5
Med-Replay	Walker2d	86.2±3.6	86.4±1.5	86.7±2.8	**86.9**±2	70.2±1.2	72.3±1.5	71±1.6	**75.8**±2.6
MuJoCo-v2 Average		80	83.5	83.4	**84.8**	73.1	77.6	78	**79**

study on Gym-MuJoCo tasks. The results are presented in Table 5. Specifically, we have the variants:

Original: train the offline RL algorithms on the original dataset.

Single: only use a single $\mathcal{M}_r$ and a a single $\mathcal{M}_s$ to predict the reward and next states, respectively.

No-Correct: use an ensemble of Reward Transformers and State Transformers to predict the reward and next states, but remove the correction of World Evaluator.

From Table 5 we observe that: (1) Single and Original show comparable results and the score standard deviation of Single is higher, especially in *Med-Replay-Hopper* (±15.5) and *Med-Replay-Walker2d* (±14.7), which suggests that the trajectories generated by using a single State Transformer and Reward Transformer could be unreliable. (2) No-Correct outperforms Original and Single in most cases, which demonstrates the necessity of utilizing an ensemble of Reward Transformers and State Transformers to model the environment dynamics. (3) OTTO achieves the best performance in most cases, indicating that the correction of the World Evaluator effectively improves policy learning.

Besides, we also evaluate the quality of trajectories by calculating the accuracy of the predicted rewards of Single and OTTO, the results are presented in Appendix B. We can observe that OTTO makes more accurate prediction than Single, indicating that the correction of the World Evaluator improves the quality of generated trajectories. To summarize, the ensemble of World Transformers together with the World Evaluator are both essential to improve the offline RL performance.

Table 5: Ablation study. The normalized scores are computed over 5 random seeds. Boldface denotes the highest score.

Dataset	Original	Single	No-Correct	OTTO
Med-Expert-Halfcheetah	92.1±1.2	82.5±2.8	84.2±2.2	**93.8**±0.6
Med-Expert-Hopper	102±5.6	105.9±1.6	111.9±0.2	**113.4**±0.3
Med-Expert-Walker2d	111.4±0.5	111.5±0.1	111.5±0.5	**112.9**±0.1
Medium-Halfcheetah	48.5±0.3	47.5±0.1	48.5±0.4	**49.2**±0.1
Medium-Hopper	61.2±4.3	66.6±14.2	77.9±8.5	**78.6**±3.5
Medium-Walker2d	80.4±0.1	78.3±0.5	82.7±2.3	**83.5**±1.5
Med-Replay-Halfcheetah	43.8±0.2	44.1±0.1	43.3±0.5	**44.8**±0.1
Med-Replay-Hopper	94.3±4.6	90.9±15.5	100.6±0.6	**102.4**±1
Med-Replay-Walker2d	86.2±3.6	83.7±14.7	83.4.2±0.8	**86.9**±2
MuJoCo Average	80	79.5	82.7	**85.1**

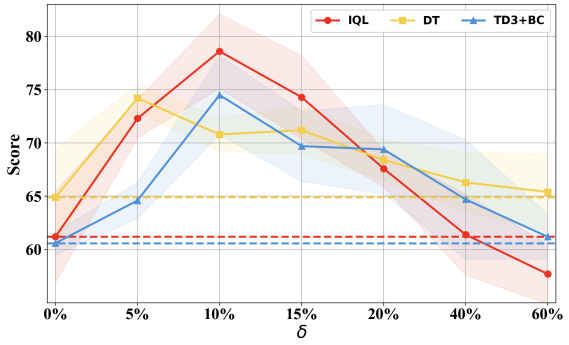

Figure 4: Impact of δ. Dashed lines are original scores.

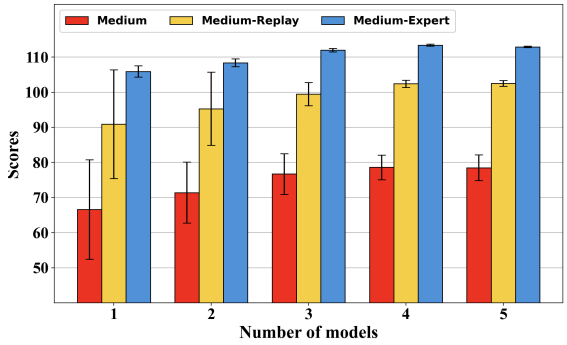

Figure 5: Impact of the number of models K and Q.

Table 6: Impact of ϵ. The normalized scores are computed over 5 random seeds. Boldface denotes the highest score.

Dataset	Original	$\epsilon = 0.02$	$\epsilon = 0.1$	$\epsilon = 0.5$
Med-Expert-Halfcheetah	92.1±1.2	**93.8**±0.6	93±0.8	92.5±1.2
Med-Expert-Hooper	102±5.6	112.4±1.2	**113.4**±0.3	108.8±0.4
Med-Expert-Walker2d	111.4±0.5	**112.9**±0.1	112.2±0.1	111.7±0.3
Med-Halfcheetah	48.5±0.3	**49.3**±0.1	49.1±0.1	48.2±0.1
Med-Hooper	61.2±4.3	76.7±2.6	**78.6**±3.5	56.3±7.2
Med-Walker2d	80.4±0.1	82.5±3.2	**83.5**±2.3	80.7±5.2
Med-Replay-Halfcheetah	43.8±0.2	**44.8**±0.2	**44.8**±0.1	44.5±0.9
Med-Replay-Hooper	94.3±4.6	**102.4**±1	100.5±2.4	92.3±4.6
Med-Replay-Walker2d	86.2±3.6	86.8±2.4	**86.9**±2	73.9±5.4
MuJoCo Average	80	84.6	**84.7**	78.8

4.3.3 Hyperparameter study. In this section, we perform a hyperparameter study on augmentation ratio δ, noise range ϵ, and the number of models K and Q.

Impact of augmentation ratio δ. To further investigate the impact of augmentation ratio δ, we conduct experiments by evaluating IQL on *Hopper-Medium* dataset with different augmentation ratios δ. Specifically, we set the augmentation ratio from 5% to 60% and present the results in Figure 4. We can observe that the performance of all three offline RL algorithms first increases and then decreases. When the augmentation ratio increases at the beginning, the performance of OTTO improves because the mixed dataset with more augmented data is able to provide more high-reward trajectories to enlarge the observed states and actions. However, when the augmentation ratio is too large, the performance will drop because too much noise could be introduced to the dataset.

Impact of noise range ϵ. We perform the hyperparameter study on ϵ by evaluating IQL on Gym MuJoCo tasks. Table 6 shows the results. We can observe that when ϵ is too large, e.g. $\epsilon = 0.5$, the average score is lower than Original. This demonstrates that too large perturbation could downgrade the method's performance. Besides, $\epsilon = 0.1$ achieves comparable performance with $\epsilon = 0.02$ and both of them outperform the original score. In simple environments like Hopper, $\epsilon = 0.1$ performs better. This indicates that the estimation is more accurate in such simple environments and a larger perturbation range leads to a broader generalization range and a better performance.

Impact of the number of models K and Q. We conduct experiments by evaluating IQL on *Hopper* environment with different number of models. Specifically, we set the number of State Transformers K equal to the number of Reward Transformers Q, and increase this number from 1 to 5. The results are shown in Figure 5. We observed that as the number of models increases, the performance also improves, while the standard deviation decreases, indicating that increasing the number of models can lead to more accurate predictions. In addition, the performance when the number of models is 4 is comparable to that when the number is 5. Considering that using more models increases training time, we use $K = Q = 4$ as a default setting.

5 Conclusion

In this work, we have presented offline trajectory optimization for offline reinforcement learning (OTTO). In particular, we have trained World Transformers to simulate the environment and proposed three generation strategies to generate long-horizon trajectories. Then World Evaluator is introduced to evaluate and correct the generated trajectories. The RL agent is trained upon both the original data and augmented trajectories. Extensive experiments on D4RL benchmarks show that OTTO improves the performance of representative model-free offline RL algorithms and outperforms strong model-based baselines. For future work, how to improve the World Evaluator and generate a large volume of augmented data to further enhance policy learning is a promising direction.

Acknowledgments

This research was (partially) supported by the Natural Science Foundation of China (62202271, 62472261, 62372275, 62272274, T2293773), the National Key R&D Program of China with grant No. 2024YFC3307300 and No. 2022YFC3303004, the Provincial Key R&D Program of Shandong Province with grant No. 2024CXGC010108, the Natural Science Foundation of Shandong Province with grant No. ZR2024QF203, the Technology Innovation Guidance Program of Shandong Province with grant No. YDZX2024088. All content represents the opinion of the authors, which is not necessarily shared or endorsed by their respective employers and/or sponsors.

References

[1] Gaon An, Seungyong Moon, Jang-Hyun Kim, and Hyun Oh Song. 2021. Uncertainty-based offline reinforcement learning with diversified q-ensemble. *Advances in neural information processing systems* 34 (2021), 7436–7447.

[2] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems* 33 (2020), 1877–1901.

[3] Chang Chen, Yi-Fu Wu, Jaesik Yoon, and Sungjin Ahn. 2022. Transdreamer: Reinforcement learning with transformer world models. *arXiv preprint arXiv:2202.09481* (2022).

[4] Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. 2021. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems* 34 (2021), 15084–15097.

[5] Minmin Chen, Can Xu, Vince Gatto, Devanshu Jain, Aviral Kumar, and Ed Chi. 2022. Off-policy actor-critic for recommender systems. In *Proceedings of the 16th ACM Conference on Recommender Systems*. 338–349.

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

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

[8] Scott Fujimoto and Shixiang Shane Gu. 2021. A minimalist approach to offline reinforcement learning. *Advances in neural information processing systems* 34 (2021), 20132–20145.

[9] Scott Fujimoto, Herke Hoof, and David Meger. 2018. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*. PMLR, 1587–1596.

[10] Scott Fujimoto, David Meger, and Doina Precup. 2019. Off-policy deep reinforcement learning without exploration. In *International conference on machine learning*. PMLR, 2052–2062.

[11] Gao Huang, Yixuan Li, Geoff Pleiss, Zhuang Liu, John E Hopcroft, and Kilian Q Weinberger. 2017. Snapshot ensembles: Train 1, get m for free. *arXiv preprint arXiv:1704.00109* (2017).

[12] Michael Janner, Qiyang Li, and Sergey Levine. 2021. Offline reinforcement learning as one big sequence modeling problem. *Advances in neural information processing systems* 34 (2021), 1273–1286.

[13] Natasha Jaques, Asma Ghandeharioun, Judy Hanwen Shen, Craig Ferguson, Agata Lapedriza, Noah Jones, Shixiang Gu, and Rosalind Picard. 2019. Way off-policy batch deep reinforcement learning of implicit human preferences in dialog. *arXiv preprint arXiv:1907.00456* (2019).

[14] Natasha Jaques, Judy Hanwen Shen, Asma Ghandeharioun, Craig Ferguson, Agata Lapedriza, Noah Jones, Shixiang Shane Gu, and Rosalind Picard. 2020. Human-centric dialog training via offline reinforcement learning. *arXiv preprint arXiv:2010.05848* (2020).

[15] Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, and Sergey Levine. 2018. QT-Opt: Scalable Deep Reinforcement Learning for Vision-Based Robotic Manipulation. arXiv:1806.10293 [cs.LG]

[16] Rahul Kidambi, Aravind Rajeswaran, Praneeth Netrapalli, and Thorsten Joachims. 2020. Morel: Model-based offline reinforcement learning. *Advances in neural information processing systems* 33 (2020), 21810–21823.

[17] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[18] Ilya Kostrikov, Rob Fergus, Jonathan Tompson, and Ofir Nachum. 2021. Offline reinforcement learning with fisher divergence critic regularization. In *International Conference on Machine Learning*. PMLR, 5774–5783.

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

[20] Aviral Kumar, Justin Fu, Matthew Soh, George Tucker, and Sergey Levine. 2019. Stabilizing off-policy q-learning via bootstrapping error reduction. *Advances in Neural Information Processing Systems* 32 (2019).

[21] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. 2020. Conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems* 33 (2020), 1179–1191.

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

[23] Guanghe Li, Yixiang Shan, Zhengbang Zhu, Ting Long, and Weinan Zhang. 2024. DiffStitch: Boosting Offline Reinforcement Learning with Diffusion-based Trajectory Stitching. *arXiv preprint arXiv:2402.02439* (2024).

[24] Siqi Liu, Kay Choong See, Kee Yuan Ngiam, Leo Anthony Celi, Xingzhi Sun, and Mengling Feng. 2020. Reinforcement learning for clinical decision support in critical care: comprehensive review. *Journal of medical Internet research* 22, 7 (2020), e18477.

[25] Yao Liu, Adith Swaminathan, Alekh Agarwal, and Emma Brunskill. 2019. Off-policy policy gradient with state distribution correction. *arXiv preprint arXiv:1904.08473* (2019).

[26] Kendall Lowrey, Aravind Rajeswaran, Sham Kakade, Emanuel Todorov, and Igor Mordatch. 2018. Plan online, learn offline: Efficient learning and exploration via model-based control. *arXiv preprint arXiv:1811.01848* (2018).

[27] Cong Lu, Philip Ball, Yee Whye Teh, and Jack Parker-Holder. 2024. Synthetic experience replay. *Advances in Neural Information Processing Systems* 36 (2024).

[28] Cong Lu, Philip J Ball, Jack Parker-Holder, Michael A Osborne, and Stephen J Roberts. 2021. Revisiting design choices in offline model-based reinforcement learning. *arXiv preprint arXiv:2110.04135* (2021).

[29] Ajay Mandlekar, Fabio Ramos, Byron Boots, Silvio Savarese, Li Fei-Fei, Animesh Garg, and Dieter Fox. 2020. Iris: Implicit reinforcement without interaction at scale for learning control from offline robot manipulation data. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 4414–4420.

[30] Tatsuya Matsushima, Hiroki Furuta, Yutaka Matsuo, Ofir Nachum, and Shixiang Gu. 2020. Deployment-Efficient Reinforcement Learning via Model-Based Offline Optimization. arXiv:2006.03647 [cs.LG]

[31] Vincent Micheli, Eloi Alonso, and François Fleuret. 2022. Transformers are sample-efficient world models. *arXiv preprint arXiv:2209.00588* (2022).

[32] Ofir Nachum, Bo Dai, Ilya Kostrikov, Yinlam Chow, Lihong Li, and Dale Schuurmans. 2019. Algaedice: Policy gradient from arbitrary experience. *arXiv preprint arXiv:1912.02074* (2019).

[33] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. 2018. Improving language understanding by generative pre-training. (2018).

[34] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog* 1, 8 (2019), 9.

[35] Rafael Rafailov, Tianhe Yu, Aravind Rajeswaran, and Chelsea Finn. 2021. Offline Reinforcement Learning from Images with Latent Space Models. , 1154–1168 pages.

[36] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research* 21, 1 (2020), 5485–5551.

[37] Marc Rigter, Bruno Lacerda, and Nick Hawes. 2022. Rambo-rl: Robust adversarial model-based offline reinforcement learning. *Advances in neural information processing systems* 35 (2022), 16082–16097.

[38] Julian Schrittwieser, Thomas Hubert, Amol Mandhane, Mohammadamin Barekatain, Ioannis Antonoglou, and David Silver. 2021. Online and offline reinforcement learning by planning with a learned model. *Advances in Neural Information Processing Systems* 34 (2021), 27580–27591.

[39] Noah Y Siegel, Jost Tobias Springenberg, Felix Berkenkamp, Abbas Abdolmaleki, Michael Neunert, Thomas Lampe, Roland Hafner, Nicolas Heess, and Martin Riedmiller. 2020. Keep doing what worked: Behavioral modelling priors for offline reinforcement learning. *arXiv preprint arXiv:2002.08396* (2020).

[40] Avi Singh, Albert Yu, Jonathan Yang, Jesse Zhang, Aviral Kumar, and Sergey Levine. 2020. Cog: Connecting new skills to past experience with offline reinforcement learning. *arXiv preprint arXiv:2010.14500* (2020).

[41] Samarth Sinha, Ajay Mandlekar, and Animesh Garg. 2022. S4rl: Surprisingly simple self-supervision for offline reinforcement learning in robotics. In *Conference on Robot Learning*. PMLR, 907–917.

[42] Richard S Sutton, A Rupam Mahmood, and Martha White. 2016. An emphatic approach to the problem of off-policy temporal-difference learning. *The Journal of Machine Learning Research* 17, 1 (2016), 2603–2631.

[43] Adith Swaminathan and Thorsten Joachims. 2015. Batch learning from logged bandit feedback through counterfactual risk minimization. *The Journal of Machine Learning Research* 16, 1 (2015), 1731–1755.

[44] Phillip Swazinna, Steffen Udluft, and Thomas Runkler. 2021. Overcoming Model Bias for Robust Offline Deep Reinforcement Learning. arXiv:2008.05533 [cs.LG]

[45] Shengpu Tang, Maggie Makar, Michael Sjoding, Finale Doshi-Velez, and Jenna Wiens. 2022. Leveraging Factored Action Spaces for Efficient Offline Reinforcement Learning in Healthcare. In *Decision Awareness in Reinforcement Learning Workshop at ICML 2022*. https://openreview.net/forum?id=wl_o_hilncS

[46] Shengpu Tang and Jenna Wiens. 2021. Model Selection for Offline Reinforcement Learning: Practical Considerations for Healthcare Settings. arXiv:2107.11003 [cs.LG]

[47] Denis Tarasov, Alexander Nikulin, Dmitry Akimov, Vladislav Kurenkov, and Sergey Kolesnikov. 2022. CORL: Research-oriented Deep Offline Reinforcement Learning Library. In *3rd Offline RL Workshop: Offline RL as a "Launchpad"*. https://openreview.net/forum?id=SyAS49bBcv

[48] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).

[49] Kerong Wang, Hanye Zhao, Xufang Luo, Kan Ren, Weinan Zhang, and Dongsheng Li. 2022. Bootstrapped transformer for offline reinforcement learning. *Advances in Neural Information Processing Systems* 35 (2022), 34748–34761.

[50] Yifan Wu, George Tucker, and Ofir Nachum. 2019. Behavior regularized offline reinforcement learning. *arXiv preprint arXiv:1911.11361* (2019).
[51] Teng Xiao and Donglin Wang. 2021. A general offline reinforcement learning framework for interactive recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 4512–4520.
[52] Tengyang Xie, Ching-An Cheng, Nan Jiang, Paul Mineiro, and Alekh Agarwal. 2021. Bellman-consistent pessimism for offline reinforcement learning. *Advances in neural information processing systems* 34 (2021), 6683–6694.
[53] Fisher Yu, Haofeng Chen, Xin Wang, Wenqi Xian, Yingying Chen, Fangchen Liu, Vashisht Madhavan, and Trevor Darrell. 2020. Bdd100k: A diverse driving dataset for heterogeneous multitask learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2636–2645.
[54] Tianhe Yu, Aviral Kumar, Rafael Rafailov, Aravind Rajeswaran, Sergey Levine, and Chelsea Finn. 2021. Combo: Conservative offline model-based policy optimization. *Advances in neural information processing systems* 34 (2021), 28954–28967.
[55] Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Y Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. 2020. Mopo: Model-based offline policy optimization. *Advances in Neural Information Processing Systems* 33 (2020), 14129–14142.
[56] Xianyuan Zhan, Xiangyu Zhu, and Haoran Xu. 2022. Model-Based Offline Planning with Trajectory Pruning. arXiv:2105.07351 [cs.AI]
[57] Weipu Zhang, Gang Wang, Jian Sun, Yetian Yuan, and Gao Huang. 2024. STORM: Efficient stochastic transformer based world models for reinforcement learning. *Advances in Neural Information Processing Systems* 36 (2024).
[58] Zeyu Zhu and Huijing Zhao. 2021. A survey of deep rl and il for autonomous driving policy learning. *IEEE Transactions on Intelligent Transportation Systems* 23, 9 (2021), 14043–14065.

A Experimental details

A.1 Hyperpatameter setting

In this section, we discuss the hyperparameters that we use for OTTO. Table 7 lists the hyperparameters we use to train the World Transformers. We use the same hyperparameters for both the State Transformers and Reward Transformers. Both the State Transformer and Reward Transformer are optimized by Adam [17] optimizer.

Table 7: Hyperparameters of World Transformers.

Hyperparameter	Value
Embedding dimension	128
Activation function	ReLU
Batch size	64
L_s, L_r (interaction steps of input)	20
K, Q (number of models)	4
Number of layers	10
Number of attention heads	4
Dropout	0.1
Learning rate	10^{-4}
Grad norm clip	0.25
Weight decay	10^{-4}
Number of warmup steps T_{wp}	10^5
Number of steps in each cycle T_{cyc}	5×10^5

Table 8 lists the hyperparameters we use to generate the trajectories.

A.2 Implementation details

For a fair comparison, we adopt the same training hyperparameters according to their original papers [4, 8, 19, 21] with all the model-free offline RL methods (IQL, DT, CQL, TD3+BC) in all the experiments.

Table 8: Hyperparameters of generating trajectories.

Hyperparameter	Value (Search Range)
augmentation ratio δ	{5%,10%,15%,20%}
range of noise ϵ	{0.02, 0.1}
trajectory length h	50
temperatureω	0.7

For IQL, TD3+BC and CQL, we use the CORL (Clean Offline Reinforcement Learning) codebase [47], which can be found in https://github.com/tinkoff-ai/CORL and is released under an Apache-2.0 license. For DT, we use the official codebase, which can be found in https://github.com/kzl/decision-transformer and is released under an MIT license.

The D4RL dataset we use is released under Creative Commons Attribution 4.0 License (CC BY) License, which can be found in https://github.com/Farama-Foundation/D4RL/tree/master.

We use NVIDIA GeForce RTX 2080 Ti to train each model. It takes 1.8-2.5 GPUhours to train one single State Transformer or Reward Transformer and 4-8 hours to generate trajectories, which depends on the different datasets.

B Additional experiments.

To further investigate the influence of the ensemble of World Transformers, we calculate the average L1 loss of states and reward between the real value and the predicted value of each interaction step in the trajectories generated by Single and OTTO. The results are presented in Figure 6 and Figure 7. We can observe that the predicted accuracy of OTTO is better than Single in most settings, which demonstrates the necessity of utilizing an ensemble of Reward Transformers and State Transformers to model the environment dynamics and utilizing World Evaluator to evaluate and correct the generated trajectories.

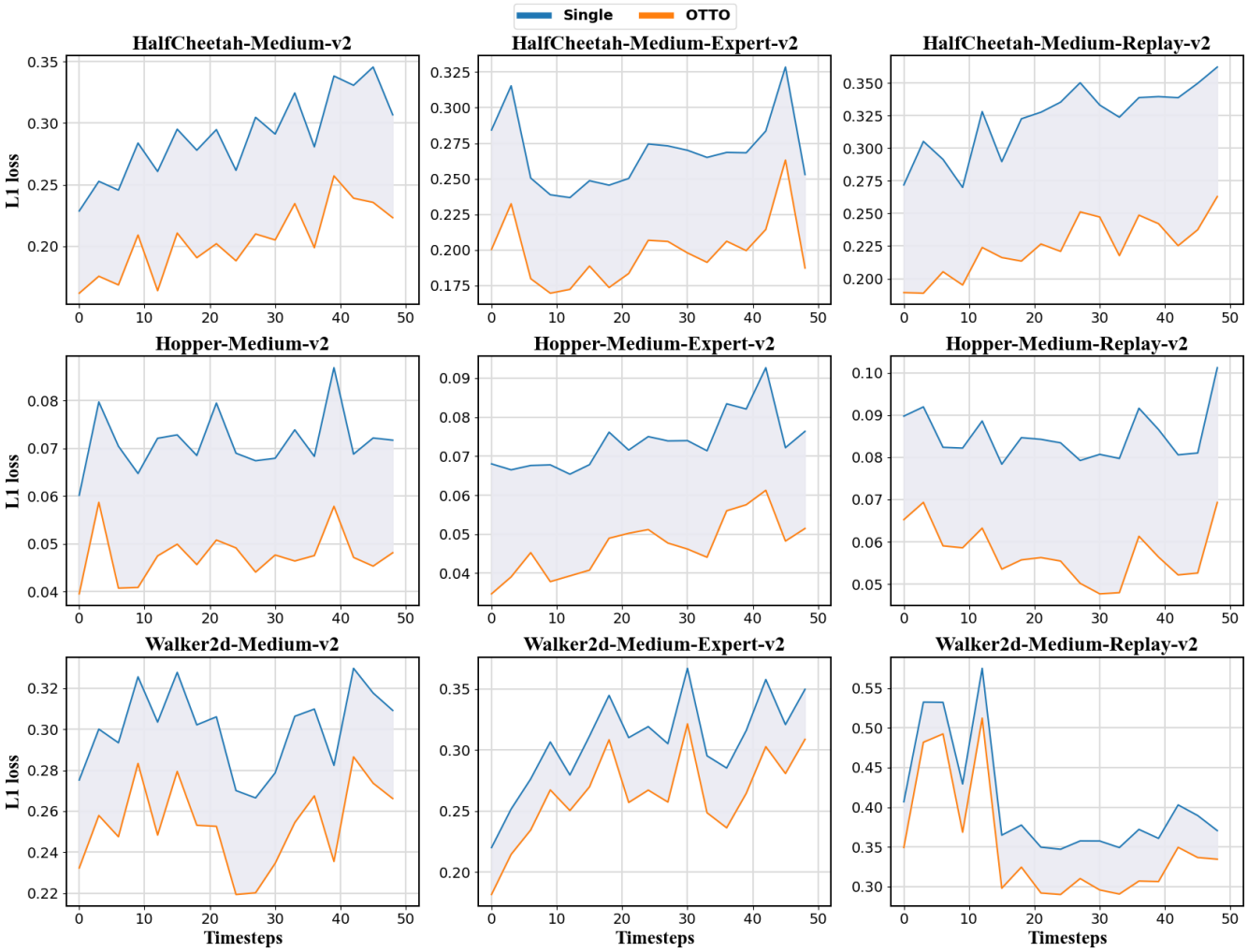

Figure 6: The average L1 loss between the real states and the predicted states of each interaction step in the trajectories generated by Single and OTTO.

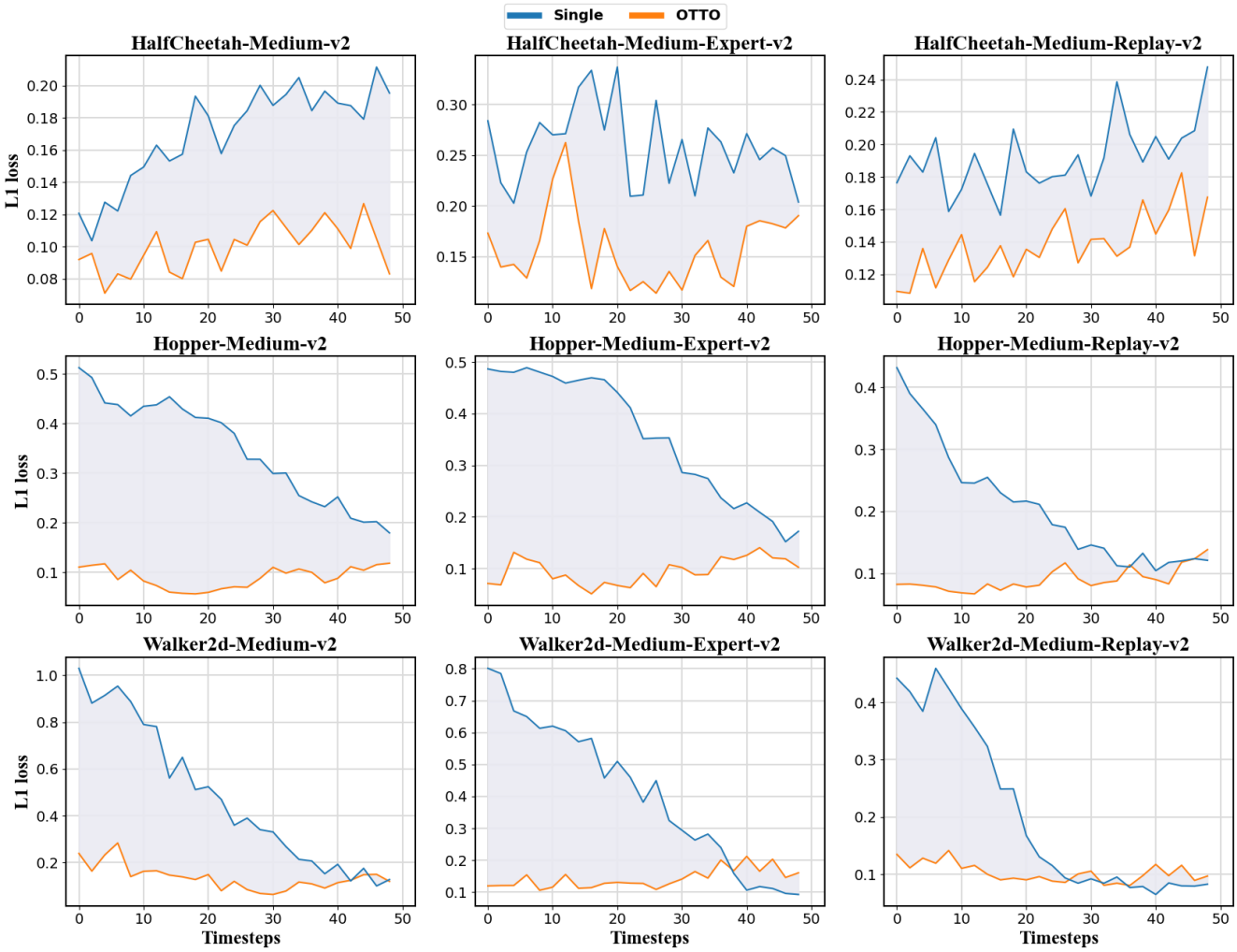

Figure 7: The average L1 loss between the real rewards and the corrected rewards of each interaction step in the trajectories generated by Single and OTTO.