DiffStitch: Boosting Offline Reinforcement Learning with Diffusion-based Trajectory Stitching

Guanghe Li [1] **Yixiang Shan** [1] **Zhengbang Zhu** [2] **Ting Long** [1] **Weinan Zhang** [2]

Abstract

In offline reinforcement learning (RL), the performance of the learned policy highly depends on the quality of offline datasets. However, in many cases, the offline dataset contains very limited optimal trajectories, which poses a challenge for offline RL algorithms as agents must acquire the ability to transit to high-reward regions. To address this issue, we introduce Diffusion-based Trajectory Stitching (DiffStitch), a novel diffusion-based data augmentation pipeline that systematically generates stitching transitions between trajectories. DiffStitch effectively connects low-reward trajectories with high-reward trajectories, forming globally optimal trajectories to address the challenges faced by offline RL algorithms. Empirical experiments conducted on D4RL datasets demonstrate the effectiveness of DiffStitch across RL methodologies. Notably, DiffStitch demonstrates substantial enhancements in the performance of one-step methods (IQL), imitation learning methods (TD3+BC), and trajectory optimization methods (DT).

1. Introduction

Recently, offline reinforcement learning (RL) (Levine et al., 2020; Agarwal et al., 2020; Fujimoto & Gu, 2021; Janner et al., 2021; Kidambi et al., 2020), which focuses on learning a policy from pre-collected, static datasets without directly interacting with the environment, has gained much attention. It has particularly wide application in scenarios where obtaining real-time feedback is costly, time-consuming, or impractical (Levine et al., 2020). By learning from the offline dataset, offline RL avoids the necessity of direct interaction with the environment, effectively eliminating the associated costs and risks.

While offline RL has achieved notable sucess in commercial

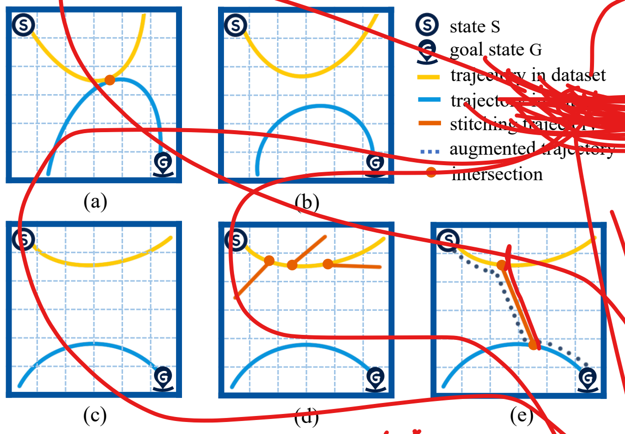


Figure 1. An illustration of trajectory stitching. Suppose there are two trajectories (blue and yellow) in the offline dataset, and the objective for the agent is to learn a policy that starts from S and reaches G. (a) and (b) present the scenarios where the trajectories in the offline dataset intersect or are in close proximity, making it easier to learn a policy that leads to G. (c) presents the scenario where trajectories are far apart, posing a challenge for learning a viable policy. (d) illustrates previous solutions that generate trajectories based on original data to enhance policy learning. Although many branches extend from the original trajectory that starts at S, none of them formalizes a sample trajectory that starts from S and reaches G. (e) illustrates our solution which stitches the trajectory starting from S and ending at G, facilitating policy learning by providing a clear path to follow.

recommendation , health care (Fatemi et al., 2022), dialog (Jaques et al., 2020) and autonomous driving (Shi et al., 2021), its performance is heavily dependent on the quality of the offline dataset. When the offline dataset suffers from inherent deficiencies or shortcomings, the performance of offline RL will remarkably decline. Consider the task in Figure 1, where an agent starts from state S and receives a positive reward only when it reaches the goal state G. Suppose there are only two trajectories in the offline dataset, one starting from state S, and one ending at state G. If the states of two trajectories intersect, as depicted in Figure 1(a), it is easy for a typical offline RL algorithm such as TD3+BC (Fujimoto & Gu, 2021) to learn a policy that starts from state S and reaches state G through temporal difference learning(Sutton & Barto, 2018). In cases where

[1]Jilin University [2]Shanghai Jiao Tong University. Correspondence to: Ting Long <longting@jlu.edu.cn>.

Preprint. Under review.

two trajectories are disjoint but have some very close states, as shown in Figure 1(b), learning a policy that starts from S and achieves G is also possible due to the generalization capabilities of value networks. However, in more challenging cases where two trajectories are disjoint and distant from each other as illustrated in Figure 1(c), learning a policy that starts from S and achieves G becomes much more difficult.

To deal with the aforementioned challenges of data deficiency, a straightforward solution is augmenting the dataset: generating synthetic data to facilitate learning a better policy. Therefore, previous works (Lu et al., 2023; Zhang et al., 2023; Wang et al., 2022) propose to learn the state transition of offline datasets and generate relatively short sub-trajectories from a randomly selected state, as Figure 1(d) illustrated. However, without specifying a desired target state, the generated sub-trajectories may not effectively enhance the policy to achieve a higher return. As shown in Figure 1(d), although many branches extend from the original trajectory that starts at S, none of them formalize a sample trajectory that starts from S and reaches G, learning the policy of reaching G is still difficult. Intuitively, if we augment the offline dataset by stitching together low-reward trajectories with high-reward trajectories, the policy may learn the ability to transit from low-reward states to high-reward states and ultimately achieve a higher overall return. For instance, if we generate a sub-trajectory to connect (stitch) the trajectory that starts from S and the trajectory that ends at G as Figure 1(e), the new trajectory (the dashed line in Figure 1(e)) will facilitate the policy learning.

Considering that, in this paper, we propose Diffusion-based Trajectory Stitching(DiffStitch), a novel data augmentation method designed to stitch (connect) the low-reward trajectories with high-reward trajectories in offline dataset to enhance the learning of policies. Specifically, it randomly selects a low-reward trajectory and a high-reward trajectory, and generates a sub-trajectory to stitch them together. Through our paradigm, one can easily transfer the trajectories trapped in low rewards to the one with high rewards, enhancing the learning of policy and significantly improving the performance of offline RL algorithms.To the best of our knowledge, DiffStitch is the first offline RL augmentation method that generates sub-trajectories to stitch any two trajectories in the dataset. We evaluate DiffStitch on various offline datasets, and results demonstrate that augmented trajectories generated by DiffStitch are effective in enhancing the performance of different types of offline RL algorithms.

Our contributions are summarized as follows:

- We propose DiffStitch, a novel paradigm for augmenting the offline RL dataset by trajectory stitching. To the best of our knowledge, DiffStitch is the first offline RL augmentation method that generates sub-trajectories to stitch any two trajectories in the dataset.
- DiffStitch benefits the performance of various offline RL algorithms, including one-step methods(IQL), imitation learning methods(TD3+BC), and sequential optimization methods(DT).
- The extensive experiments on the widely-used D4RL datasets demonstrate the superiority of our method.

2. Related Work

Offline RL. According to (Prudencio et al., 2023), offline RL algorithms could be grouped into four groups: model-based methods, one-step methods, imitation learning, and trajectory optimization methods. Model-based methods (Kidambi et al., 2020; Matsushima et al., 2020; Yu et al., 2020) learn policy by using the offline dataset to model the dynamic of the environment. One-step methods (Kostrikov et al., 2021; Brandfonbrener et al., 2021) learn policy by performing in-sample Bellman updates to learn an accurate estimate of the Q function, which is followed by a single policy improvement step to find the best possible policy. Imitation learning methods (Chen et al., 2020; Wang et al., 2020; Siegel et al., 2020) learn the policy by mimicking the optimal behaviors and filtering out suboptimal behaviors. Trajectory optimization (Chen et al., 2021; Janner et al., 2021) methods focus on training a joint state-action model over entire trajectories. Although these offline RL methods have made significant achievements in recent years, their performance is highly dependent on the offline dataset.

Data Augmentation in Offline RL. Some of the previous works propose to enhance the performance of offline RL algorithms by data augmentation. We group these methods into single-way rollout (Wang et al., 2021; Lyu et al., 2022; Zhang et al., 2023) and dynamic tuple simulation (Lu et al., 2023). The single-way rollout methods design (inverse) dynamic models and a rollout policy to synthesize the augmented trajectories. For instance, TATU (Zhang et al., 2023) employs a forward dynamic model to roll out trajectories and adopt a truncates mechanism to guarantee the accumulated certainty. The dynamic tuple simulation methods apply generative models to simulate the environment dynamics and generate new transition tuples. For instance, SER (Lu et al., 2023) applies diffusion models to capture the joint distribution of transition tuples in the offline dataset and augment the dataset by sampling new tuples. Although these methods did enhance the performance of offline RL, they are incapable of generating long trajectories and are limited in challenging scenarios where very few high-reward trajectories are available in the original offline dataset.

3. Preliminaries

Consider a standard Markov Decision Process (MDP) $\langle \mathcal{S}, \mathcal{A}, \rho_0, p, r, \gamma \rangle$, where $\mathcal{S}$ denotes the state space, $\mathcal{A}$ de-

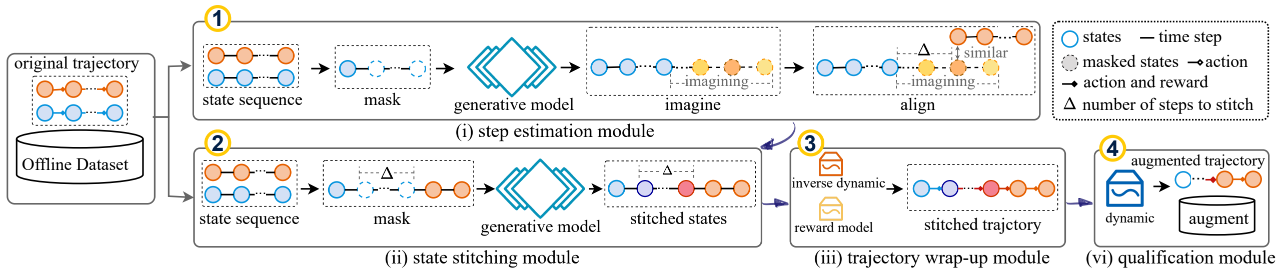


Figure 2. The overall pipeline framework of DiffStitch.

notes the action space, ρ_0 denotes the initial state distribution, p denotes the transition function, γ represents the discount factor, and r is the instant reward. At each time step t, an agent takes action $\boldsymbol{a}_t (\boldsymbol{a}_t \in \mathcal{A})$ to respond to the state of environment $\mathbf{s}_t (\boldsymbol{s}_t \in \mathcal{S})$ according to a policy $\pi(\boldsymbol{a}_t|\boldsymbol{s}_t)$, and gains an instant reward r_t. Then the time step transfers to $t+1$, and the state of the environment transits to $\boldsymbol{s}_{t+1}$ with transition probability $p(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t, \boldsymbol{a}_t)$. The interaction between the agent and environment could be written as a trajectory $\tau = \{(\boldsymbol{s}_1, \boldsymbol{a}_1, r_1), (\boldsymbol{s}_2, \boldsymbol{a}_2, r_2), \dots\}$. The goal of the agent is to learn a policy π^* that can maximize expected discounted return, *i.e.*, $\sum_{t=0}^{\infty} \gamma^t r_t$.

In the setting of offline RL, given an offline dataset $\mathcal{D}$ which is composed of pre-collected trajectories, the agent is supposed to learn an optimal policy from $\mathcal{D}$ (no further interactions are allowed). Hence, for an arbitrary offline RL algorithm Γ, it is first trained with the offline dataset $\mathcal{D}$ to obtain the optimal policy π, and we have $\pi = \Gamma(\mathcal{D})$. Subsequently Γ is tested in the environment to evaluate its performance. Specifically, the performance is computed by:

$$\mathbb{E}_{\boldsymbol{\tau} \sim \Gamma(\mathcal{D})} \left[\sum_{t=0}^{\infty} \gamma^t r_t \right] . \qquad (1)$$

Previous works mainly focused on maximizing Eq. (1) by designing a better learning algorithm Γ. Obviously, the performance of policy π also largely depends on the quality of offline dataset $\mathcal{D}$. Therefore, in this paper, we aim to generate an augmented dataset $\mathcal{D}^*$, such that for any algorithm Γ in a set of reasonable learning algorithms, the policy learned from $\mathcal{D}^*$ would achieve better performances compared to learning from $\mathcal{D}$:

$$\mathbb{E}_{\boldsymbol{\tau} \sim \Gamma(\mathcal{D})} \left[\sum_{t=0}^{\infty} \gamma^t r_t \right] < \mathbb{E}_{\boldsymbol{\tau} \sim \Gamma(\mathcal{D}^*)} \left[\sum_{t=0}^{\infty} \gamma^t r_t \right] . \qquad (2)$$

4. Method

To obtain the augmented dataset $\mathcal{D}^*$, we propose DiffStitch, solving the problem from the perspective of trajectory stitching. In the following, we will discuss the detailed pipeline of DiffStitch and subsequently the implementation and training of the models involved.

4.1. DiffStitch

We aim to augment the offline dataset $\mathcal{D}$ by stitching, which generates the transitions that "link" low-reward trajectories to high-reward trajectories, as illustrated in Figure 1(e). Suppose there is a low-reward trajectory $\tau = \{(\boldsymbol{s}_1, \boldsymbol{a}_1, r_1), \dots, (\boldsymbol{s}_T, \boldsymbol{a}_T, r_T)\}$ and a high-reward trajectory $\tau' = \{(\boldsymbol{s}'_1, \boldsymbol{a}'_1, r'_1), \dots, (\boldsymbol{s}'_{T'}, \boldsymbol{a}'_{T'}, r'_{T'})\}$, where T and T' denote the length of τ and τ'. A straightforward way to stitch them together is directly feeding a masked trajectory composed of the elements of τ and τ', *i.e.*, $\{(\boldsymbol{s}_1, \boldsymbol{a}_1, r_1), \dots, (\boldsymbol{s}_T, \boldsymbol{a}_T, r_T)$, `[MASK]`, ..., `[MASK]`, $(\boldsymbol{s}'_1, \boldsymbol{a}'_1, r'_1)$, ..., $(\boldsymbol{s}'_{T'}, \boldsymbol{a}'_{T'}, r'_{T'})\}$, to a generative model and predict the `[MASK]`. However, this naive approach has a significant drawback in that it can not determine the number of `[MASK]` (*i.e.*, the time steps lying between τ and τ'). A fixed time step inserted between two trajectories could be either too large or too small, which leads to a violation of environmental dynamics and resides in out-of-distribution (OOD) regions. Besides, rewards and actions usually do not change smoothly as states across time, so generating them with a single model might bring some potential issues (Ajay et al., 2022).

To tackle this problem, our proposed DiffStitch first estimates the steps for stitching and then generates the states, actions, and rewards separately. Specifically, as illustrated in Figure 2, it is composed of four modules: step estimation module, state stitching module, trajectory wrap-up module, and qualification module. The step estimation module estimates the number of time steps required for stitching two trajectories. Then, the estimated number of time steps is input into the state stitching module, which generates states consistent with the number of estimated steps. Next, the trajectory wrap-up module predicts the rewards and actions based on the states generated by the state stitching module to obtain a new trajectory. Finally, the qualification module evaluates the quality of the new trajectory and augments the dataset with the new trajectory if it is qualified.

4.1.1. Step Estimation Module

As discussed in previous work (Ajay et al., 2022) that the state transitions are smooth within a trajectory, we use states to estimate the number of steps for stitching. Therefore, the task of step estimation is transformed that given the states of two trajectory $\tau_s = (s_1, s_2 ... s_T)$ and $\tau'_s = (s'_1, s'_2, ..., s'_{T'})$, estimate the number of the step required for state $\boldsymbol{s}_T$ to transit naturally to state $\boldsymbol{s}'_1$. Here, $\boldsymbol{s}_T$ could be viewed as the start state and $\boldsymbol{s}'_1$ could view as the target state for stitching. Suppose $\boldsymbol{s}_T$ can reach state x in exactly i steps. The higher the similarity between $\boldsymbol{s}'_1$ and x, the more probable it is for $\boldsymbol{s}_T$ to reach $\boldsymbol{s}'_1$ in exactly i steps as well. Inspired by that, to identify the number of steps required for stitching $\boldsymbol{s}_T$ to $\boldsymbol{s}'_1$, we first "imagine" the subsequent states of state $\boldsymbol{s}_T$, find the state closest to $\boldsymbol{s}'_1$, and treating the interval between $\boldsymbol{s}_T$ and the state closest to $\boldsymbol{s}'_1$ as the number of required step.

Specifically, we first initialize a masked sequence starting with state $\boldsymbol{s}_T$, *i.e.*, $(\boldsymbol{s}_T, \texttt{[MASK]}, ..., \texttt{[MASK]})$, and feed it to a conditional generative model to generate the subsequent states of $\boldsymbol{s}_T$:

$$\begin{aligned} \tau_s^m &= (\boldsymbol{s}_1^m, \boldsymbol{s}_2^m, ..., \boldsymbol{s}_{H-1}^m), \\ &= \mathcal{G}_\theta\left((\boldsymbol{s}_T, \texttt{[MASK]}, ..., \texttt{[MASK]})\right), \end{aligned} \tag{3}$$

$+$

where $\mathcal{G}_\theta$ denotes the generative model, the implementation and training of $\mathcal{G}_\theta$ will be discussed in Section 4.2. H is the generating horizon of the generative model $\mathcal{G}_\theta$. It is worth noting that, $\boldsymbol{s}_T$ is considered as the only condition in Eq. (3).

Next, we compare the similarity of $\mathbf{s}'_1$ and the subsequent states of $\boldsymbol{s}_T$, *i.e.*, $(\boldsymbol{s}_1^m, ..., \boldsymbol{s}_{H-1}^m)$, to identify the number of steps required to transit between two trajectories, which could be calculated by:

$$\Delta = \arg\max_i \text{sim}(\boldsymbol{s}_i^m, \boldsymbol{s}'_1), \tag{4}$$

where Δ denotes the number of time steps to connect $\boldsymbol{s}_T$ and $\boldsymbol{s}'_1$. Obviously, $\Delta \leq H-2$. Here, we implement sim$(\cdot)$ with cosine similarity.

4.1.2. State Stitching Module

With Δ, we are able to know the exact number of states to be generated between the last state of trajectory $\tau = \{(\boldsymbol{s}_1, \boldsymbol{a}_1, r_1), (\boldsymbol{s}_2, \boldsymbol{a}_2, r_2), \ldots, (\boldsymbol{s}_T, \boldsymbol{a}_T, r_T)\}$ and the first state of trajectory $\tau' = \{(\boldsymbol{s}'_1, \boldsymbol{a}'_1, r'_1), (\boldsymbol{s}'_2, \boldsymbol{a}'_2, r'_2), \ldots, (\boldsymbol{s}'_{T'}, \boldsymbol{a}'_{T'}, r'_{T'})\}$. Applying this information, we first construct a masked sequence composed of the last states of τ and the initial states in τ'_s:

$$\tau_{s,m} = (\boldsymbol{s}_T, \underbrace{\texttt{[MASK]}, \ldots, \texttt{[MASK]}}_{\Delta \text{ masks}}, \boldsymbol{s}'_1, ..., \boldsymbol{s}'_{H-1-\Delta}), \tag{5}$$

Here, the required elements of mask sequence $\tau_{s,m}$ are state $\boldsymbol{s}_T$ and and $\boldsymbol{s}'_1$. $(\boldsymbol{s}'_2, \boldsymbol{s}'_3, ..., \boldsymbol{s}'_{H-1-\Delta})$ is used to pad the sequence if $\Delta < H-2$. Then, we apply $\mathcal{G}_\theta$ to reconstruct the masked elements, and states that stitch $\boldsymbol{s}_T$ to $\boldsymbol{s}'_1$ can be generated as follows:

$$\begin{aligned} \tau_s^s &= \mathcal{G}_\theta(\tau_{s,m}) \\ &= (\boldsymbol{s}_T, \underbrace{\widetilde{\boldsymbol{s}}_1, \ldots, \widetilde{\boldsymbol{s}_\Delta}}_{\Delta \text{ stitching states}}, \boldsymbol{s}'_1, ..., \boldsymbol{s}'_{H-1-\Delta}), \end{aligned} \tag{6}$$

Here, we denote the states $\widetilde{\tau_s} = (\widetilde{\boldsymbol{s}}_1, \ldots, \widetilde{\boldsymbol{s}_\Delta})$ are the **stitching states**.

4.1.3. Trajectory Wrap-up Module

We aim to wrap-up the states with actions and rewards to obtain the sub-trajectory for stitching. Specifically, for each state pair $(\hat{\boldsymbol{s}}_t, \hat{\boldsymbol{s}}_{t+1}) \in \{(\boldsymbol{s}_T, \widetilde{\boldsymbol{s}}_1), (\widetilde{\boldsymbol{s}}_1, \widetilde{\boldsymbol{s}}_2), \ldots, (\widetilde{\boldsymbol{s}_\Delta}, \boldsymbol{s}'_1)\}$, we apply a inverse dynamic model f_ψ to predict the actions between two adjacent states, that is:

$$\boxed{\widetilde{\boldsymbol{a}}_t = f_\psi(\hat{\boldsymbol{s}}_t, \hat{\boldsymbol{s}}_{t+1})} \tag{7}$$

For each state-action pair $(\hat{\boldsymbol{s}}_t, \widetilde{\boldsymbol{a}}_t) \in \{(\boldsymbol{s}_T, \widetilde{\boldsymbol{a}_T}), (\widetilde{\boldsymbol{s}}_1, \widetilde{\boldsymbol{a}}_1), \ldots, (\widetilde{\boldsymbol{s}_\Delta}, \widetilde{\boldsymbol{a}_\Delta})\}$, we apply a reward model f_ϕ to predict the reward between two adjacent states:

$$\boxed{\widetilde{r}_t = f_\phi(\hat{\boldsymbol{s}}_t, \widetilde{\boldsymbol{a}}_t),} \tag{8}$$

The implementation and training of f_ψ and f_ϕ will be discussed in Section 4.2.

Therefore, the sub-trajectory for stitching can be represented as $\tau_r = \{(\boldsymbol{s}_T, \widetilde{\boldsymbol{a}_T}, \widetilde{r_T}), (\widetilde{\boldsymbol{s}}_1, \widetilde{\boldsymbol{a}}_1, \widetilde{r}_1), ..., (\widetilde{\boldsymbol{s}_\Delta}, \widetilde{\boldsymbol{a}_\Delta}, \widetilde{r_\Delta}), (\boldsymbol{s}'_1, \boldsymbol{a}'_1, r'_1)\}$. We denote τ_r as the **stitching trajectory**.

4.1.4. Qualification Module

Though the stitching trajectories could be generated with the techniques mentioned previously, their quality often varies. Therefore, we aim to assess the quality of generated trajectories, filtering out low-quality data while retaining high-quality data to ensure that the stitching trajectories align with environmental dynamics.

For the stitching trajectory $\tau_r = \{(\boldsymbol{s}_T, \widetilde{\boldsymbol{a}_T}, \widetilde{r_T}), (\widetilde{\boldsymbol{s}}_1, \widetilde{\boldsymbol{a}}_1, \widetilde{r}_1), ..., (\widetilde{\boldsymbol{s}_\Delta}, \widetilde{\boldsymbol{a}_\Delta}, \widetilde{r_\Delta}), (\boldsymbol{s}'_1, \boldsymbol{a}'_1, r'_1)\}$, we use the dynamics model (Liang et al., 2023) f_ω to assess the consistency of the generated data with the environmental dynamics. Specifically, for each tuple $(\hat{\boldsymbol{s}}_t, \hat{\boldsymbol{a}}_t, \hat{\boldsymbol{s}}_{t+1}) \in \{(\boldsymbol{s}_T, \widetilde{\boldsymbol{a}_T}, \widetilde{\boldsymbol{s}}_1), (\widetilde{\boldsymbol{s}}_1, \widetilde{\boldsymbol{a}}_1, \widetilde{\boldsymbol{s}}_2), ..., (\widetilde{\boldsymbol{s}_\Delta}, \widetilde{\boldsymbol{a}_\Delta}, \boldsymbol{s}'_1)\}$, we predict the next state via:

$$\hat{\boldsymbol{s}}_{t+1}^q = f_\omega(\hat{\boldsymbol{s}}_t, \hat{\boldsymbol{a}}_t). \tag{9}$$

The implementation and training of f_w will be discussed in Section 4.2. We discard stitching trajectory τ_r if there is a predicted state $\hat{\boldsymbol{s}}_{t+1}$ that deviates significantly from the corresponding state in τ_r, i.e.

$$||\hat{\boldsymbol{s}}_{t+1}^q - \hat{\boldsymbol{s}}_{t+1}||^2 \geq \delta, \tag{10}$$

where δ denotes the qualification threshold. This ensures that the stitching trajectory is aligned with the environmental dynamics.

By employing the pipeline above, we can obtain a new augmented trajectory $\tau_{gen} = \{(\boldsymbol{s}_1, \boldsymbol{a}_1, r_1), ..., (\boldsymbol{s}_T, \widetilde{\boldsymbol{a}_T}, \widetilde{r_T}), (\widetilde{\boldsymbol{s}_1}, \widetilde{\boldsymbol{a}_1}, \widetilde{r_1}), ..., (\widetilde{\boldsymbol{s}_\Delta}, \widetilde{\boldsymbol{a}_\Delta}, \widetilde{r_\Delta}), (\boldsymbol{s}'_1, \boldsymbol{a}'_1, r'_1), ..., (\boldsymbol{s}'_{T'}, \boldsymbol{a}'_{T'}, r'_{T'})\}$ to enhance the offline dataset. We put the augmented trajectories together to obtain the augmented dataset $\mathcal{D}_{\text{aug}}$.

4.2. Model Implementation and Training

The models used for stitching in Section 4.1 are generative model $\mathcal{G}_\theta$, inverse dynamic model f_ψ, reward model f_ϕ and dynamic model f_ω, we will discuss their implementation and training in this section.

The generative model $\mathcal{G}_\theta$ is designed to generate the transition states between to-stitch trajectories, and we implemented it with a diffusion model in this paper. Therefore, for noise-masked state sequence $\tau_{s,m}$, $\mathcal{G}_\theta$ reconstruct it with K denoising steps :

$$\tau_{s,m}^{k-1} = \frac{1}{\sqrt{\alpha_k}}\left(\tau_{s,m}^k - \frac{1-\alpha_k}{\sqrt{(1-\bar{\alpha}_k)}}\epsilon_\theta(\tau_{s,m}^k, \tau_{s,m}, k)\right) + \sqrt{1-\alpha_k}\epsilon,$$
$$\epsilon \sim \mathcal{N}(0, \mathbf{I}), \text{for } k = K, .., 1, \tag{11}$$

where α_i is the $cosine$ function of k and $\bar{\alpha}_i = \prod_1^i \alpha_t$. $\mathcal{N}$ denotes the Gaussian distribution. ϵ_θ is a neural network, we use U-Net (Ronneberger et al., 2015) to implement it. $\tau_{s,m}^0$ is the reconstructed states without any noise. From Eq. 11, we can infer that training $\mathcal{G}_\theta$ is essentially identifying the parameters of ϵ_θ.

To identify the parameters, we apply the state sequence of trajectories in offline dataset, and train ϵ_θ in a adding-noise procedure like previous works(Ajay et al., 2022; Janner et al., 2022).

Specifically, for an arbitrary trajectory, we first take out the state sequence of trajectories with a horizon of size H, *i.e.*, $\tau_s = (\mathbf{s}_1, \mathbf{s}_2, ..., \mathbf{s}_H)$, and mask it with $\tau_{s,m} = (\mathbf{s}_i, \texttt{[MASK]}, ..., \texttt{[MASK]}, \mathbf{s}_k, \mathbf{s}_{k+1}.., \texttt{[MASK]})$, where the mask pattern randomly conceals two intervals. We strictly ensure that the initial state of τ_s is unmasked, while the terminal state of τ_s is masked.

condition

Next, we randomly sample a noise $\epsilon \sim \mathcal{N}(0, \mathcal{I})$ and a diffusion timestep $k \sim \mathcal{U}\{1, ..., K\}$, and train $\mathcal{G}_\theta$ with:

$$\mathcal{L}_\theta = \mathbb{E}_{k,\tau_s \in \mathcal{D}}||\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}}\tau_s + \sqrt{1-\bar{\alpha}}\epsilon, \tau_{s,m}, k)||^2. \tag{12}$$

The inverse dynamic model f_ψ, reward model f_ϕ, and dynamic model f_ω are all implemented with MLPs (Multiple Layer Perceptrons). We use states, rewards, and actions in the trajectories of offline datasets to train them. That is, they are trained by:

$$\begin{aligned}\mathcal{L}_\phi &= \mathbb{E}_{(\boldsymbol{s}_t, \boldsymbol{a}_t, r_t, \boldsymbol{s}_{t+1}) \sim \mathcal{D}}\left[\boldsymbol{a}_t - f_\psi(\boldsymbol{s}_t, \boldsymbol{s}_{t+1})\right], \\ \mathcal{L}_\psi &= \mathbb{E}_{(\boldsymbol{s}_t, \boldsymbol{a}_t, r_t, \boldsymbol{s}_{t+1}) \sim \mathcal{D}}\left[r_t - f_\phi(\boldsymbol{s}_t, \boldsymbol{a}_t)\right], \\ \mathcal{L}_\omega &= \mathbb{E}_{(\boldsymbol{s}_t, \boldsymbol{a}_t, r_t, \boldsymbol{s}_{t+1}) \sim \mathcal{D}}\left[\boldsymbol{s}_{t+1} - f_\omega(\boldsymbol{s}_t, \boldsymbol{a}_t)\right],\end{aligned} \tag{13}$$

We summarize the detailed pseudo code for DiffStitch in Appendix A.1. In training offline RL algorithms, we sample data from $\mathcal{D}^* = \mathcal{D} \cup \mathcal{D}_{\text{aug}}$ with a fixed ratio $r =$ (number of original data : number of augmented data) between the original data from $\mathcal{D}$ and the augmented data from $\mathcal{D}_{\text{aug}}$ in each training batch.

effect?

5. Experiments

In this section, we will discuss the experimental settings and the performance of DiffStitch.

5.1. Experiment Settings

Evaluation Environments We evaluate DiffStitch on a wide range of domains in the D4RL benchmark (Fu et al., 2020), including MuJoCo tasks and Adroit tasks. Specifically, we evaluate the performance of DiffStitch on three environments in MuJoCo tasks: HalfCheetah, Hopper, and Walker2d. Each environment includes three types of datasets: Medium-expert, Medium, and Medium-replay, in which the Medium-expert is composed of both expert demonstrations and suboptimal data, Medium-replay and Medium are collected when an unconverged policy (Haarnoja et al., 2018) interacted with the environment. We evaluate the performance of DiffStitch with the dataset of Human and Cloned in Adroit-pen tasks, which aims to control a robot to twirl a pen. The Human dataset is composed of human demonstrations and the Cloned dataset is collected by an imitation policy on the demonstrations. Compared to MuJoCo tasks, Adroit-pen has a larger observation space (45 dimensions) and action space (24 dimensions), making it more challenging.

Baselines Since we aim to augment the offline dataset to improve the performance of offline RL algorithms, we compare our method with two recent data augmentation algorithms in offline RL, SER(Lu et al., 2023) and TATU(Zhang et al., 2023), which have demonstrated outstanding performance. SER is designed to utilize the diffusion model for learning the distribution of transitions in the offline dataset and generating new transitions. However, since these transitions are generated without any conditions, SER cannot generate complete trajectories. On the other hand, TATU employs a forward dynamic model to simulate transitions based on the offline dataset. Nevertheless, TATU is specifically designed to roll out forward transitions in very limited steps to ensure that the generated transitions remain within

Table 1. The average normalized score of different methods. Here $\pm$ denoting the variance. The mean and standard deviation are computed over 3 random seeds. The best and the second-best results of each setting are marked as **bold** and <u>underline</u>, respectively.

IL RL

Dataset	**Environment**	IQL				TD3+BC				DT	
		Original	SER	TATU	**DStitch**	Original	SER	TATU	**DStitch**	Original	**DStitch**
Med-Expert	HalfCheetah	92.7±2.5	88.9±2.1	**94.4**±0.6	**94.4**±1.4	<u>94.3</u>±0.9	86.5±8.8	89.3±3.9	**96**±0.5	<u>90.8</u>±1.4	**92.6**±0.1
Med-Expert	Hopper	98.7±10	<u>110.4</u>±1.6	93.4±17.8	**110.9**±2.9	94.8±13	<u>104</u>±10.8	99±14.9	**107.1**±7	**109.9**±1.9	<u>109.4</u>±1.9
Med-Expert	Walker2d	**112.4**±0.8	<u>111.7</u>±1.1	110.7±0.6	111.6±0.1	109.9±0.8	<u>110.5</u>±0.3	**110.7**±0.7	110.2±0.3	<u>108.1</u>±0.3	**108.6**±0.4
Medium	HalfCheetah	48.5±0.3	<u>49.3</u>±0.1	48.2±0.1	**49.4**±0.14	<u>48.4</u>±0.1	<u>48.4</u>±0.4	48.1±0.2	**50.4**±0.5	<u>40.4</u>±2.2	**44.2**±0.3
Medium	Hopper	65.9±5.7	<u>66.6</u>±2.4	60.3±3.6	**71**±4.2	58±2.8	56.4±1.3	<u>58.3</u>±4.8	**60.3**±4.9	**61.5**±2.2	<u>60.5</u>±4.3
Medium	Walker2d	81.1±1.8	**85.9**±1.6	76.6±10.7	<u>83.2</u>±2.2	81.4±2.3	**84.9**±0.3	75.8±3.5	<u>83.4</u>±1.7	<u>70.5</u>±1.6	**72**±4.9
Med-Replay	HalfCheetah	44.1±0.5	**46.6**±0.1	44.2±0.1	<u>44.7</u>±0.1	44.5±0.2	**45.2**±0.1	44.5±0.3	<u>44.7</u>±0.3	<u>39.8</u>±1.6	**41**±0.4
Med-Replay	Hopper	91.4±8.1	**102.4**±0.5	79.6±7.6	<u>102.1</u>±0.4	<u>64.8</u>±19.2	56.8±9.9	64.1±10.5	**79.6**±13.5	<u>83.6</u>±3.9	**96.1**±2
Med-Replay	Walker2d	80.7±7	<u>85.7</u>±3.6	75±12.1	**86.6**±2.8	82.4±5	<u>89.1</u>±0.5	62.1±10.4	**89.7**±4.2	<u>53.3</u>±11.2	**60.2**±1.8
Locomotion Average		79.5	<u>83.1</u>	75.8	**83.8**	75.4	<u>75.8</u>	72.4	**80.2**	73.1	**76.1**
human	pen	79.1±28.5	<u>88.9</u>±22.6	**96.8**±8.6	87.4±8.6	-3.3±0.4	-2.8±1.5	<u>7.3</u>±14.1	**29.8**±6.9	<u>62.8</u>±2.1	**70**±6.8
cloned	pen	45.8±29.9	<u>52.5</u>±27.9	45.3±23.4	**64**±29.6	-3.1±0.4	<u>-2.8</u>±0.1	-3±0.3	**11.3**±6.2	<u>57.7</u>±3.3	**59.7**±9.6
Pen Average		62.5	70.7	<u>71.1</u>	**75.7**	-3.2	-2.8	<u>2.2</u>	**20.6**	60.3	**64.9**

the distribution, thereby making it challenging to generate long trajectories.

Evaluation Algorithms To assess the quality of the augmented data, we utilize the augmented dataset to train three groups of offline RL algorithms: TD3+BC (Fujimoto & Gu, 2021), IQL (Kostrikov et al., 2021), and DT (Chen et al., 2021). All of these methods have been widely employed in previous studies due to their popularity (Prudencio et al., 2023). Here IQL is a one-step method, which first learns the value function and then derives the policy based on the learned value function. TD3+BC is a imitation learning method, which jointly learns the policy and value function. DT is a trajectory optimization method, which takes trajectory data as input and applies Transformer (Vaswani et al., 2017) to model the distribution of trajectories in the offline dataset.

Implementaion Details For the generative model, the horizon of generation is set to 100 for MuJoCo tasks, and 55 for Adroit-pen tasks. The denoising step K is set to 100. We implement the inverse dynamics model f_ψ with a 2-layer MLP, and the dynamic model f_ω, reward model f_ϕ with a 4-layer MLP. For the stitching, we set the qualification threshold used for data selection δ ranging from $[1, 16]$ depending on the dataset. Our model is trained on a device with A5000 GPUs(24GB GPU memory, 27.8 TFLOPS computing capabilities, AMD EPYC 7371 16-Core Processor, optimized by Adam (Kingma & Ba, 2014) optimizer.

5.2. Main Results

To evaluate the performance of DiffStitch and compare its performance with the baselines, we train TD3+BC and IQL with the augmented data generated by DiffStitch, SER, and TATU, as well as the original offline data. This allows us to compare the effectiveness of the augmented data. Additionally, we use the data generated by DiffStitch and the original offline data to train DT in order to assess the effectiveness of DiffStitch. We did not employ the data generated by SER and TATU to train DT because either they were unable to generate trajectories or could only produce very short trajectories, which makes the data generated by them inapplicable to augment decision transformer(DT). The results are presented in Table 1. Each result is obtained by conducting 10 trials with different seeds, and reported with the normalized average returns (Fu et al., 2020) and standard deviation.

From Table 1, we can observe: (1) Our DiffStitch achieves the best performance in most cases, and obtains the best av-

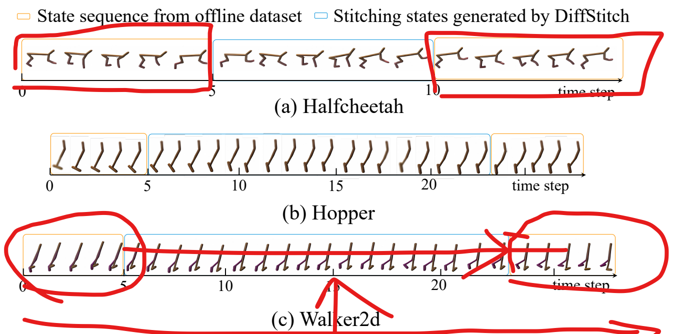
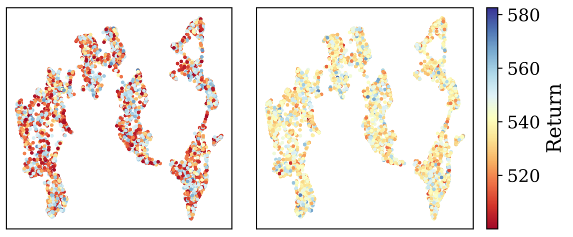


Figure 3. The stitching trajectory.



(a) Before stitching (b) After stitching

Figure 4. Return of states before and after stitching

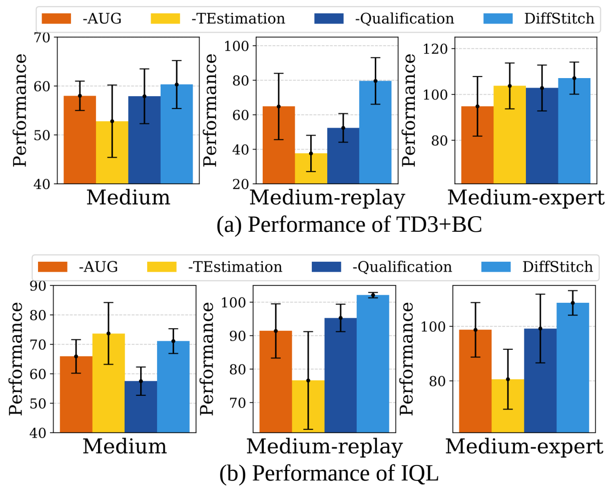
erage performance in both MuJoCo tasks and Adroit tasks. This demonstrates the data generated by our method has higher quality than the data generated by SER and TATU in most cases; (2) Our DiffStitch demonstrates superior performance on challenging tasks. As Table 1 illustrated, It is easier for DiffStitch to achieve significant improvement in the cases where the original score is relatively low, like the cases on pen, the case of TD3+BC on Med-Replay Hopper. The low score in these cases demonstrates that offline RL algorithms have difficulty exploring high-reward regions. With the trajectory stitching, our DiffStitch potentially offers high-reward training samples for them. (3) Both our DiffStitch and SER utilize the diffusion model for data augmentation. However, our DiffStitch achieves better performance than SER in 14 out of 22 cases, indicating that augmenting data from the perspective of trajectory stitching is more effective in most scenarios.(4)Offline RL methods trained on augmented data have better performance than the one trained on original data, which demonstrates enhancing the performance of offline RL via data augmentation is effective.

5.3. Further Analysis

As observed in Section 5.2, DiffStitch outperforms baseline augmentation methods in most cases. We assume two features mainly contribute to the advancements: (1) The generated stitching sub-trajectories are consistent with the environment dynamics, and (2) DiffStitch successfully transforms low-reward trajectories to high-reward trajectories. Both features ensure the provision of high-quality data for offline RL algorithms to learn from, resulting in improved performance. To investigate whether the stitching sub-trajectories conform to environment dynamics, we randomly select three augmented trajectories, one for each of HalfCheetah, Walker2d, and Hopper. We visualize the robot states at each time step, and the results are presented in Figure 3. We can observe the stitching states are quite natural and roughly align with the locomotion patterns of these robots. To check whether the generated samples transform low-reward trajectories to high-reward trajectories, we randomly take the states from the left side of stitching trajectories and compare their returns before and after stitching in Figure 4. We can observe that the returns of the majority of states have improved after stitching. As the returns are the accumulation of discounted future rewards, the improvement in return indicates that the originally lower-reward trajectories starting from those states are transformed into higher-reward trajectories, serving as more valuable data in policy learning.

5.4. Ablation Study

To investigate the impact of the modules in DiffStitch, we apply IQL and TD3+BC to conduct ablation studies on the environment of Hopper. Specifically, we have the variants:



(a) Performance of TD3+BC

(b) Performance of IQL

Figure 5. Ablation study on the hopper

- **-AUG** remove the augmented data, and directly train the offline RL algorithms with the original data.
- **-TEstimation** remove the step estimation module from DiffStitch. In this case, the number of stitching steps is set to a constant value 40.
- **-Qualification** remove the qualification module from DiffStitch. In this case, all the data generated by DiffStitch are directly used to train the offline RL algorithms.

The results are illustrated in Figure 5. We can observe from Figure 5: (1) the performance of DiffStitch is better than -TEstimation in most cases. This demonstrates that estimating the number of stitching steps is important. Conversely, utilizing a fixed number of states to stitch trajectories might be detrimental, leading to a degradation in the original performance; (2) the performance of DiffStitch is better than -Qualification in all cases. That demonstrates the crucial role of data selection; (3) DiffStitch outperforms -AUG in all the cases, which highlights the benefits of data augmentation in offline RL tasks.

5.5. Parameter Study

To further investigate the factors that influence the performance of DiffStitch, we investigate the impact of data ratio and qualification threshold δ.

The impact of data ratio r DiffStitch enhances the effectiveness of offline RL algorithms by generating augmented data. In the training of offline RL algorithms, both the augmented dataset and original offline dataset are used. Consequently, the ratio of the original data to the augmented

Table 2. The impact of data ratio r.

ratio	TD3+BC	IQL	DT
0:1	0.1±0.9	-0.5±1.0	1.2±0.3
1:2	91.7±2.3	91.5±3.7	**92.6**±0.1
1:1	94.1±1.2	91.7±3.7	92.5±0.6
2:1	93.2±2.8	93.9±1.0	91.5±1.0
4:1	**96**±0.5	**94.4**±1.4	92.4±0.5
1:0	94.3±0.9	92.7±2.5	90.8±1.4

Table 3. The impact of qualification threshold δ.

δ	TD3+BC	IQL	DT
1.0	93.8±2.0	92±1.7	93.1±0.2
2.0	**96**±0.5	94.4±1.4	92.6±0.1
3.0	91.7±3.2	**95**±0.5	**92.7**±0.6
4.0	94.4±0.8	91.1±2.7	92.6± 1

data might impact the performance of DiffStitch. To investigate that, we conduct experiments on Halfcheetah-Medium-expert with different ratios. Specifically, we set the ratio to $\{0:1, 1:1, 2:1, 4:1, 9:1, 1:0\}$, and other settings remain the same to test the performance. The results are illustrated in Table 2. Except for the effectiveness of the augmentation, we can also observe in Table 2 that both an excessively large ratio and an excessively small ratio have detrimental effects on the performance of DiffStitch. We assume the reasons are as follows: When ratio is excessively large(1:0), the benefit brought by augmentation is abandoned. When data ratio is excessively small(0:1), too much noise are introduced and agents cannot effectively learn a policy.

The impact of qualification threshold δ. The qualification threshold δ determines the decision to exclude generated transitions that deviate from environmental dynamics, guaranteeing that the generated transitions remain within in-distribution (ID) regions. To investigate the impact of δ, we conduct experiments on Halfcheetah-Medium-Expert, and test the performance of our method under TD3+BC, IQL and DT. The experiment results are presented in Table 3. We can observe from Table 3 that a too small δ or a too large δ decrease the performance of all the evaluation algorithms (TD3+BC, IQL and DT). We assume that a smaller δ provides stronger assurance of adherence to the distribution but may be overly conservative, whereas a larger δ may permit transitions that deviate from environmental dynamics, falling into OOD regions.

5.6. Additional Study on Small Samples

To investigate the performance of DiffStitch on the dataset of limited samples, we randomly select 20% of the samples from the original dataset, and augment the dataset with DiffStitch. Then, we train IQL, TD3+BC, and DT with the augmented dataset and compare the performance with the baselines. The experiments are conducted on HalfCheetah-Medium-Expert and Hopper-Medium-Replay, which covers different environments and different dataset qualities. The results are illustrated in Table 4. We can observe from Table 4: (1) in the easier task (HalfCheetah-Medium-Expert), DiffStitch leverages 20% data to achieve a performance close to using 100% of the data in Table 1, which demonstrates the effectiveness of Diffstitch; (2) In the challenging tasks (such as Hopper-Medium-Replay), Diffstitch demonstrates a significant advantage over the baselines. This highlights the superiority of Diffstitch in tackling difficult tasks. By stitching together low-reward trajectories with high-reward trajectories, Diffstitch effectively compensates for the limitations of the dataset.

Table 4. The performance on small samples. ± denoting the variance. The mean and standard deviation are computed over 3 random seeds. The best and the second-best results of each setting are marked as **bold** and underline, respectively.

	Dataset	Environment	Original	SER	TATU	**DiffStitch**
IQL	Med-Expert	HalfCheetah	89.3±4.3	62.5±2.5	**91.5**±2.6	**91.5**±1.5
	Med-Replay	Hopper	51±3.9	13±1.1	49.6±13.2	**67.8**±13.5
	Average		70.2	37.8	70.6	**79.7**
TD3+BC	Med-Expert	HalfCheetah	81.2±8.9	65.1±7.5	80.2±6.3	**91.6**±3.4
	Med-Replay	Hopper	52±10.1	13.5±1.3	6.7±3.7	**64**±25
	Average		66.6	39.3	43.5	**77.8**
DT	Med-Expert	HalfCheetah	88.7±2.4	-	-	**92.6**±0.16
	Med-Replay	Hopper	49.3±8.3	-	-	**75.1**±11.6
	Average		69	-	-	**83.9**

6. Conclusion and Future Work

In this paper, we introduce DiffStitch, which enhances the learning of offline RL algorithms by stitching together low-reward trajectories with high-reward trajectories. Diverging from conventional approaches that primarily support one-step methods and imitation learning methods, our Diffstitch extends its utility to facilitate the learning of trajectory optimization methods such as Decision Transformer(DT). Empirical evaluations conducted on the D4RL benchmarks demonstrate a significant performance boost for base offline RL algorithms.

For future work, exploring how to choose and concatenate two trajectories will be an interesting direction. While our current approach involves concatenating low-reward trajectories with high-reward trajectories, there undoubtedly exist superior concatenation strategies waiting to be explored.

Border Impact

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Agarwal, R., Schuurmans, D., and Norouzi, M. An optimistic perspective on offline reinforcement learning. In International Conference on Machine Learning, pp. 104–114. PMLR, 2020.

Ajay, A., Du, Y., Gupta, A., Tenenbaum, J., Jaakkola, T., and Agrawal, P. Is conditional generative modeling all you need for decision-making? arXiv preprint arXiv:2211.15657, 2022.

Brandfonbrener, D., Whitney, W., Ranganath, R., and Bruna, J. Offline rl without off-policy evaluation. Advances in neural information processing systems, 34:4933–4946, 2021.

Chen, L., Lu, K., Rajeswaran, A., Lee, K., Grover, A., Laskin, M., Abbeel, P., Srinivas, A., and Mordatch, I. Decision transformer: Reinforcement learning via sequence modeling. Advances in neural information processing systems, 34:15084–15097, 2021.

Chen, X., Zhou, Z., Wang, Z., Wang, C., Wu, Y., and Ross, K. Bail: Best-action imitation learning for batch deep reinforcement learning. Advances in Neural Information Processing Systems, 33:18353–18363, 2020.

Fatemi, M., Wu, M., Petch, J., Nelson, W., Connolly, S. J., Benz, A., Carnicelli, A., and Ghassemi, M. Semi-markov offline reinforcement learning for healthcare. In Conference on Health, Inference, and Learning, pp. 119–137. PMLR, 2022.

Fu, J., Kumar, A., Nachum, O., Tucker, G., and Levine, S. D4rl: Datasets for deep data-driven reinforcement learning. arXiv preprint arXiv:2004.07219, 2020.

Fujimoto, S. and Gu, S. S. A minimalist approach to offline reinforcement learning. Advances in neural information processing systems, 34:20132–20145, 2021.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In International conference on machine learning, pp. 1861–1870. PMLR, 2018.

Janner, M., Li, Q., and Levine, S. Offline reinforcement learning as one big sequence modeling problem. Advances in neural information processing systems, 34: 1273–1286, 2021.

Janner, M., Du, Y., Tenenbaum, J. B., and Levine, S. Planning with diffusion for flexible behavior synthesis. arXiv preprint arXiv:2205.09991, 2022.

Jaques, N., Shen, J. H., Ghandeharioun, A., Ferguson, C., Lapedriza, A., Jones, N., Gu, S. S., and Picard, R. Human-centric dialog training via offline reinforcement learning. arXiv preprint arXiv:2010.05848, 2020.

Kidambi, R., Rajeswaran, A., Netrapalli, P., and Joachims, T. Morel: Model-based offline reinforcement learning. Advances in neural information processing systems, 33: 21810–21823, 2020.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980, 2014.

Kostrikov, I., Nair, A., and Levine, S. Offline reinforcement learning with implicit q-learning. arXiv preprint arXiv:2110.06169, 2021.

Levine, S., Kumar, A., Tucker, G., and Fu, J. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. arXiv preprint arXiv:2005.01643, 2020.

Liang, Z., Mu, Y., Ding, M., Ni, F., Tomizuka, M., and Luo, P. Adaptdiffuser: Diffusion models as adaptive self-evolving planners. arXiv preprint arXiv:2302.01877, 2023.

Lu, C., Ball, P. J., and Parker-Holder, J. Synthetic experience replay. arXiv preprint arXiv:2303.06614, 2023.

Lyu, J., Li, X., and Lu, Z. Double check your state before trusting it: Confidence-aware bidirectional offline model-based imagination. Advances in Neural Information Processing Systems, 35:38218–38231, 2022.

Matsushima, T., Furuta, H., Matsuo, Y., Nachum, O., and Gu, S. Deployment-efficient reinforcement learning via model-based offline optimization. arXiv preprint arXiv:2006.03647, 2020.

Prudencio, R. F., Maximo, M. R., and Colombini, E. L. A survey on offline reinforcement learning: Taxonomy, review, and open problems. IEEE Transactions on Neural Networks and Learning Systems, 2023.

Ronneberger, O., Fischer, P., and Brox, T. U-net: Convolutional networks for biomedical image segmentation. In Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18, pp. 234–241. Springer, 2015.

Shi, T., Chen, D., Chen, K., and Li, Z. Offline reinforcement learning for autonomous driving with safety and exploration enhancement. arXiv preprint arXiv:2110.07067, 2021.

Siegel, N. Y., Springenberg, J. T., Berkenkamp, F., Abdolmaleki, A., Neunert, M., Lampe, T., Hafner, R., Heess, N., and Riedmiller, M. Keep doing what worked: Behavioral modelling priors for offline reinforcement learning. arXiv preprint arXiv:2002.08396, 2020.

Sutton, R. S. and Barto, A. G. Reinforcement learning: An introduction. MIT press, 2018.

Tarasov, D., Nikulin, A., Akimov, D., Kurenkov, V., and Kolesnikov, S. Corl: Research-oriented deep offline reinforcement learning library. arXiv preprint arXiv:2210.07105, 2022.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. Advances in neural information processing systems, 30, 2017.

Wang, J., Li, W., Jiang, H., Zhu, G., Li, S., and Zhang, C. Offline reinforcement learning with reverse model-based imagination. Advances in Neural Information Processing Systems, 34:29420–29432, 2021.

Wang, K., Zhao, H., Luo, X., Ren, K., Zhang, W., and Li, D. Bootstrapped transformer for offline reinforcement learning. Advances in Neural Information Processing Systems, 35:34748–34761, 2022.

Wang, Z., Novikov, A., Zolna, K., Merel, J. S., Springenberg, J. T., Reed, S. E., Shahriari, B., Siegel, N., Gulcehre, C., Heess, N., et al. Critic regularized regression. Advances in Neural Information Processing Systems, 33: 7768–7778, 2020.

Yu, T., Thomas, G., Yu, L., Ermon, S., Zou, J. Y., Levine, S., Finn, C., and Ma, T. Mopo: Model-based offline policy optimization. Advances in Neural Information Processing Systems, 33:14129–14142, 2020.

Zhang, J., Lyu, J., Ma, X., Yan, J., Yang, J., Wan, L., and Li, X. Uncertainty-driven trajectory truncation for data augmentation in offline reinforcement learning. In ECAI 2023, pp. 3018–3025. IOS Press, 2023.

A. Experimental Details

A.1. Pseudocode of DiffStitch

Algorithm 1 DiffStitch

```
1: Input: offline dataset $\mathcal{D}$, iterations $N$, Discriminator threshold $d$
2: Initialize $\mathcal{D}_{\text{aug}} \leftarrow \emptyset$.
3: Train generative model $\mathcal{G}_\theta$ on $\mathcal{D}$ using Eq.(12)
4: Train Dynamic model $f_\omega$, inverse dynamic model $f_\psi$, and reward model $f_\phi$ on $\mathcal{D}$ using Eq.(13)
5: for $i = 1$ to $N$ do
6:     Sample trajectories $\tau_s$, $\tau'_s$ from $\mathcal{D}$
7:     Imagine $\tau^m_s = (s_T, \boldsymbol{s}^m_1, ..., \boldsymbol{s}^m_{H-1})$ starting from $\boldsymbol{s}_T$
8:     Set timestep $\Delta = \arg\max_i \text{sim}(\boldsymbol{s}^m_i, \boldsymbol{s}'_1)$
9:     Create $\tau_{s,m} = (s_T \underbrace{, \ldots,}_{\Delta \text{ Masks}} , s'_1, ...)$
10:    Generate state sequence $\tau^s_s = \mathcal{G}_\theta(\tau_{s,m})$
11:    Wrap-up states Eq.7 and Eq.8, obtain $\tau_{gen}$
12:    For each $(\boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{s}_{t+1}) \in \tau_{gen}$, sample $\hat{\boldsymbol{s}}^q_{t+1} \sim f_\omega(\boldsymbol{s}_t, \boldsymbol{a}_t)$
13:    if $\max\{||\hat{\boldsymbol{s}}^q_{t+1} - \boldsymbol{s}_{t+1}||^2 > d\}$ then
14:        continue
15:    end if
16:    $\mathcal{D}_{\text{aug}} \leftarrow \mathcal{D}_{\text{aug}} \cup \tau_{gen}$
17: end for
18: obtain $\mathcal{D}^* \leftarrow \mathcal{D} \cup \mathcal{D}_{\text{aug}}$
```

A.2. Experimental Details

Validation Offline RL algorithms. For the validation offline RL algorithms IQL(Kostrikov et al., 2021) and Decision Transformer(Chen et al., 2021), we use the 'CORL: Research-oriented Deep Offline Reinforcement Learning Library'(Tarasov et al., 2022) codebase. For TD3+BC(Fujimoto & Gu, 2021), we use the author-provided code from GitHub. For IQL and TD3+BC, we train them for 10^6 gradient steps across all tasks. For Decision Transformer(DT), we train it for 5×10^4 gradient steps on MuJoCo-Gym tasks and 10^5 gradient steps on Adroit-pen tasks.