MITIGATING THE NEED FOR BELLMAN COMPLETENESS: MULTI-FRAGMENT TRAJECTORY STITCHING FOR DECISION TRANSFORMER

Anonymous authors
Paper under double-blind review

ABSTRACT

The Decision Transformer (DT) has achieved promising results in offline reinforcement learning by reframing it as sequence modeling. However, DT fundamentally struggles with trajectory stitching—the ability to combine trajectory fragments into globally superior trajectories. This limitation stems from two intertwined factors: (i) DT is a supervised learning model whose policy space is confined to the support of the training data distribution, and (ii) offline datasets, especially suboptimal ones, often lack coverage of the compositional trajectories that stitch together high-return fragments. In contrast, dynamic programming (DP)-based approaches, such as Q-learning, naturally support trajectory stitching through value iteration. However, they rely on Bellman completeness and accurate function approximation, which can be challenging to satisfy in practice. To bridge this gap, we propose Multi-Fragment Trajectory Stitching (MFTS), a diffusion-based data augmentation framework that constructs synthetic trajectories by stitching high-return fragments from suboptimal data. MFTS stitches high-return fragments together by generating forward and backward state trajectories with a pre-trained conditional diffusion model, then fusing them via a time-aware asymmetric blending strategy. The resulting bridge is completed with actions and rewards through auxiliary models. MFTS thus grants DT the benefits of dynamic programming while mitigating the need for Bellman completeness. Extensive experiments on the D4RL benchmark demonstrate that MFTS achieves superior or competitive performance in most settings, with the largest gains on challenging medium-replay datasets where stitching matters most.

1 INTRODUCTION

Offline reinforcement learning (RL) aims to learn effective policies from static datasets without online interaction. This paradigm becomes fundamentally challenging when the dataset is suboptimal: the agent must stitch locally optimal segments into globally superior trajectories, a capability known as *trajectory stitching* (Fu et al., 2020). The stitching ability represents a fundamental divide between offline RL paradigms. Dynamic programming (DP)-based methods such as CQL (Kumar et al., 2020) and IQL (Kostrikov et al., 2021) support stitching through the Bellman operator, but they rely on Bellman completeness and accurate value functions, which are difficult to satisfy with function approximation (Zhou et al., 2023). See Appendix A.1 for a detailed discussion.

Return-conditioned supervised learning (RCSL), particularly Decision Transformer (DT) (Chen et al., 2021), have reframed offline RL as a conditional sequence generation problem. By leveraging the autoregressive capabilities of Transformer architectures, DT models trajectories as sequences of returns-to-go, states, and actions, enabling policy learning via supervised learning. While DT has shown remarkable success in learning from offline datasets, it fundamentally lacks trajectory stitching ability (Yamagata et al., 2023). Its policy space is confined to the support of the training data distribution, and suboptimal offline datasets rarely cover the compositional trajectories that combine locally optimal fragments. As a result, globally optimal stitched trajectories lie outside the data support and cannot be recovered by supervised learning alone. This limitation is illustrated in Figure 1.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

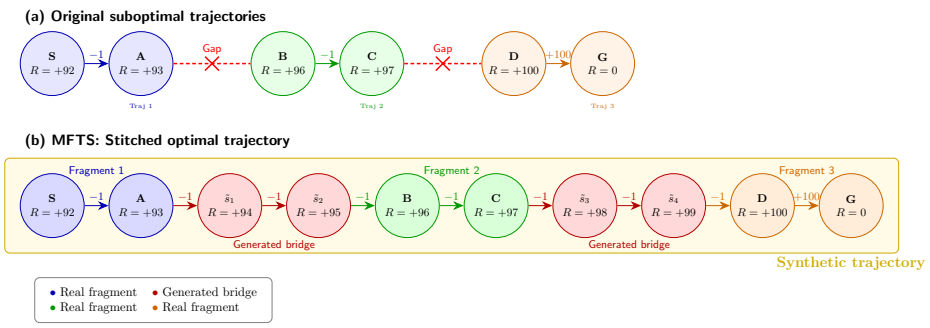

Figure 1: DT fails to reach the goal due to its lack of stitching ability, whereas MFTS succeeds by stitching fragments with a generated bridge. Arrow numbers denote rewards; R denotes RTG.

To endow DT with stitching ability, QDT (Yamagata et al., 2023) relabels the return-to-go (RTG) targets in the offline dataset using conservative Q-value estimates. CGDT (Wang et al., 2023) employs a learned critic to introduce action correction during training to align DT policy with the critic's guidance. However, both still rely on accurate value function estimation and degrade when the Q-function is poorly learned, especially in long-horizon or sparse-reward settings.

Our key insight is that DT's lack of stitching ability stems from two factors: (i) DT is fundamentally a supervised learning model whose policy space is limited to the span of behaviors present in the training data, and (ii) offline datasets, especially suboptimal ones, often lack sufficient coverage of the compositional trajectories that consist of locally optimal fragments. Consequently, globally optimal trajectories constructed by stitching multiple high-return fragments typically lie outside the support of the empirical data distribution, rendering DT incapable of recovering them despite its expressive sequence modeling capacity.

To address this, we introduce Multi-Fragment Trajectory Stitching (MFTS), a diffusion-based data augmentation framework. MFTS stitches together multiple high-return fragments from an offline dataset into synthetic trajectories, generating bridges via time-aware asymmetric fusion of forward and backward state trajectories, followed by action/reward completion and filtering. Unlike MBRCSL (Zhou et al., 2023), which learns an environment dynamics model and a behavior policy, and trains the final policy exclusively on model-generated rollouts, MFTS keeps the source fragments as real transitions and only generates the bridges between them. This design also distinguishes MFTS from prior methods that operate at the transition or full-trajectory level (Lu et al., 2023; Li et al., 2024): by generating bridges alone, MFTS models long-horizon behavior as a structured composition of short primitives. Experiments on MuJoCo show that MFTS significantly enhances DT's performance, with up to 27.7% relative improvement on Hopper-Medium-Replay.

2 Background

Offline RL

Offline RL learns effective decision-making policies from a pre-collected, fixed dataset generated by an unknown behavior policy, without further interaction with the environment. Formally, an RL problem is modeled as a Markov Decision Process (MDP) described by the tuple $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$, where $\mathcal{S}$ and $\mathcal{A}$ denote the state and action spaces respectively, $P(s' \mid s, a)$ is the transition dynamics and $R(s, a)$ is the reward function, and $\gamma \in (0, 1]$ is the discount factor. A trajectory is defined as $\tau = (s_1, a_1, r_1, s_2, a_2, r_2, \ldots, s_T, a_T, r_T)$ where T is the episode time horizon, and offline dataset $\mathcal{D}$ consists of a set of trajectories. RL aims to learn a policy that maximizes the expected return $\mathbb{E}_\pi \left[\sum_{t=1}^{T} \gamma^{t-1} r_t\right]$.

Diffusion Probabilistic Models

Diffusion models have achieved remarkable success in generative modeling by learning to reverse a forward noising process (Ho et al., 2020; Ajay et al., 2023; Janner et al., 2022). The data-generating

process is formulated as an iterative denoising procedure $p_\theta(\mathbf{x}_{k-1}|\mathbf{x}_k)$ (Sohl-Dickstein et al., 2015; Ho et al., 2020), which reconstructs the original data from noise. This denoising is the reverse of a forward diffusion process that gradually adds Gaussian noise to the data:

$$q(\mathbf{x}_k|\mathbf{x}_{k-1}) = \mathcal{N}\left(\mathbf{x}_k; \sqrt{1-\beta_k}\mathbf{x}_{k-1}, \beta_k\mathbf{I}\right), \tag{1}$$

or equivalently,

$$\mathbf{x}_k = \sqrt{1-\beta_k}\,\mathbf{x}_{k-1} + \sqrt{\beta_k}\,\epsilon, \quad \epsilon \sim \mathcal{N}(0, \mathbf{I}),$$

where $\beta_k \in (0,1)$ is the noise variance at step k and $\{\beta_k\}_{k=1}^K$ is the variance schedule.

By defining $\alpha_k = 1 - \beta_k$ and $\bar{\alpha}_k = \prod_{i=1}^k \alpha_i$, the distribution of $\mathbf{x}_k$ can be derived directly as:

$$q(\mathbf{x}_k|\mathbf{x}_0) = \mathcal{N}\left(\mathbf{x}_k; \sqrt{\bar{\alpha}_k}\mathbf{x}_0, (1-\bar{\alpha}_k)\mathbf{I}\right). \tag{2}$$

To train diffusion models, a simplified surrogate loss (Ho et al., 2020) is defined as:

$$\mathcal{L}_{denoise}(\theta) = \mathbb{E}_{\mathbf{x}_0\sim\mathcal{D}, k\sim\mathcal{U}(1,K), \epsilon\sim\mathcal{N}(0,\mathbf{I})}\|\epsilon - \epsilon_\theta(\mathbf{x}_k, k)\|^2, \tag{3}$$

where $\mathcal{U}$ represents the discrete uniform distribution and ϵ_θ is a noise prediction network trained to approximate the added noise.

Decision Transformer

Decision Transformer (DT) (Chen et al., 2021) reframes offline RL as a sequence modeling problem. Using a causal Transformer, DT predicts actions autoregressively conditioned on a trajectory context, bypassing explicit policy and value iteration.

Rather than feeding the model with rewards directly, DT computes returns-to-go $R_t = \sum_{t'=t}^T r_{t'}$ and takes the returns-to-go as part of its inputs. Given a target return, past states and actions, DT learns a policy $\pi(a_t \mid \tau_{<t}, R_t, s_t)$ that generates actions conditioned on the desired return.

During training, given a fixed dataset of trajectories, DT samples mini-batches of sequence length K from the dataset and predicts actions $\hat{a}$ based on input tokens. For continuous action spaces, the prediction head uses mean squared error (MSE), while for discrete action spaces, it employs cross-entropy loss. The loss is computed at each timestep and averaged over the sequence.

3 Related Work

Offline RL Methods

Existing offline RL approaches fall into four main categories: value-based methods, policy-constrained methods, model-based approaches, and sequence modeling. Value-based methods, such as Conservative Q-Learning (CQL) (Kumar et al., 2020) and Implicit Q-Learning (IQL) (Kostrikov et al., 2021), leverage dynamic programming to enable trajectory stitching. Policy-constrained methods restrict the learned policy to stay close to the behavior policy, avoiding explicit value bootstrapping. Representative methods include BCQ (Fujimoto et al., 2019), BEAR (Kumar et al., 2019), AWAC (Nair et al., 2021), and TD3+BC (Fujimoto & Gu, 2021). Model-based methods learn environment models from offline data. One-step (Yu et al., 2020) and multi-step (Hafner et al., 2024) dynamics models support reward penalization (Kidambi et al., 2021), uncertainty-aware planning (Hansen et al., 2022), and synthetic experience generation, guiding policy learning within the support of the observed data. Finally, sequence modeling methods reframe policy learning as conditional sequential decision-making. The Decision Transformer (DT) (Chen et al., 2021) employs autoregressive Transformers conditioned on return-to-go targets to predict actions, bypassing value bootstrapping. Later work (Janner et al., 2022) discretizes states, actions, and rewards and uses beam search for planning. However, sequence modeling fundamentally lacks stitching ability, as its supervised learning formulation relies on observed trajectory coverage.

Data Augmentation Methods

Data augmentation in offline reinforcement learning synthesizes plausible interactions to extend the effective support of a static dataset. Early methods operate at the transition level, using dynamics ensembles with uncertainty-aware weighting (Zhang et al., 2023) or diffusion models for

single-step oversampling (Lu et al., 2023). Recent efforts have shifted toward trajectory-level synthesis, including policy-guided rollouts (Jackson et al., 2024; He et al., 2023), curiosity-driven prioritization (Wang et al., 2025), and diffusion-based trajectory modeling. Representative approaches include DiffStitch (Li et al., 2024), which connects low- and high-return trajectories via one-directional diffusion, and BiTrajDiff (Qing et al., 2026), which synthesizes full trajectories by generating forward and backward trajectories anchored at intermediate states. However, these methods lack the flexibility to stitch together multiple high-return fragments from the offline dataset.

Comparison with the Most Related Work

Most related to our MFTS are DiffStitch and BiTrajDiff. However, they differ in two key aspects. First, DiffStitch connects one low-return and one high-return trajectory, while BiTrajDiff generates a full trajectory anchored at a single intermediate state. In contrast, MFTS stitches multiple short, high-return fragments from heterogeneous regions, enabling flexible multi-source composition and broader coverage of the data distribution. Second, DiffStitch generates a single bridging segment via one-directional diffusion, without explicit bidirectional fusion, and BiTrajDiff generates forward and backward trajectories independently, without explicitly resolving conflicts between the two directions. MFTS instead introduces a time-aware asymmetric blending strategy that assigns varying weights to the forward trajectory along the temporal horizon, selectively integrating reliable information from each direction based on proximity to known boundary states. These differences make MFTS more flexible in source composition and more effective in resolving bidirectional generation conflicts.

4 METHOD

We propose Multi-Fragment Trajectory Stitching (MFTS) to endow decision transformers (DTs) with the capability of trajectory stitching by bridging multiple local fragments extracted from an offline dataset. MFTS first constructs an expert buffer, then generates bridges between fragments by fusing forward and backward diffusion-based state trajectories with a time-aware asymmetric blending strategy, and finally completes the actions and rewards with auxiliary models before filtering. The overall pipeline is summarized in Algorithm 1 and illustrated in Figure 2.

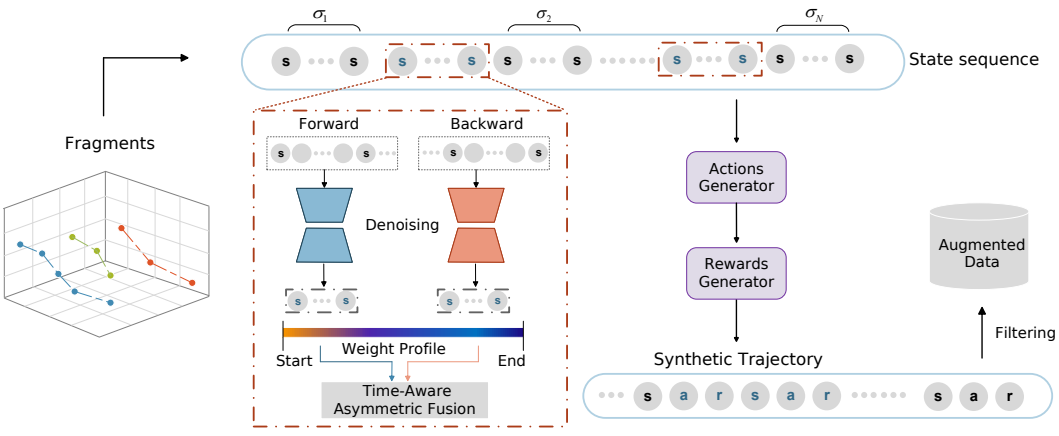

Figure 2: Overview of MFTS. Given an offline dataset, MFTS constructs an expert buffer, samples multiple fragments, generates forward and backward state trajectories at each boundary via a pre-trained diffusion model, and fuses them with a time-aware asymmetric blending strategy. The fused states are completed with actions and rewards, then filtered to form the augmented dataset for DT training.

4.1 FRAGMENT EXTRACTION VIA RETURN MAXIMIZATION

Given a suboptimal dataset $\mathcal{D} = \{\tau^{(i)}\}_{i=1}^{M}$, we select the top-$k$ trajectories ranked by discounted return $R^{(i)} = \sum_{t=1}^{T_i} \gamma^{t-1} r_t^{(i)}$ to form an expert buffer $\mathcal{F} \subseteq \mathcal{D}$. This buffer provides high-quality behavioral exemplars while preserving the diversity of the original data.

Algorithm 1 Multi-Fragment Trajectory Stitching (MFTS)

Require: Offline dataset $\mathcal{D}$, window length W, discount factor γ, filtering thresholds δ and $r_{\min}$.
Construct expert buffer $\mathcal{F}$ by selecting top-k trajectories in $\mathcal{D}$.
Sample and sort fragments $\{\sigma_1, \ldots, \sigma_N\}$ from $\mathcal{F}$.
for each pair (σ_k, σ_{k+1}) **do**
  Set $s_{\text{src}} \leftarrow$ terminal state of σ_k, $s_{\text{tgt}} \leftarrow$ initial state of σ_{k+1}.
  Denoise masked sequence and determine L via equation 5.
  Generate forward and backward state trajectories via equation 6.
  Extract forward $\mathbf{f}$ and backward $\mathbf{b}$.
  Fuse $\mathbf{f}$ and $\mathbf{b}$ via equation 7.
end for
Complete actions and rewards via f_ψ and f_ϕ to form τ_{bridge}.
Filter τ_{bridge} via dynamics and reward consistency checks.
Concatenate source fragments with τ_{bridge} to form τ_{syn}; add to $\mathcal{D}_{\text{aug}}$.

4.2 Trajectory Generation via Multi-Fragment Stitching

We sample $N \geq 2$ fragments $\{\sigma_1, \ldots, \sigma_N\}$ from $\mathcal{F}$, each of length W, and sort them by local return $R(\sigma_k) = \sum_{t=1}^{W} \gamma^{t-1} r_t^{(k)}$ in ascending order to form a skill-progression sequence. Naively concatenating fragments creates discontinuities at boundaries (e.g., from $s_W^{(k)}$ to $s_1^{(k+1)}$). To bridge them, we use a pre-trained conditional diffusion model over state sequences, trained on $\mathcal{D}$ to capture environment dynamics. Unlike the unconditional denoising objective in Eq. equation 3, the model here is conditioned on the boundary states and fragment context $\mathbf{c}$, and predicts the noise from a masked state sequence $\mathbf{x}_k$:

$$\mathcal{L}_{\text{bridge}}(\theta) = \mathbb{E}_{\mathbf{x}_0, \mathbf{c}, k, \epsilon} \left[\|\epsilon - \epsilon_\theta(\mathbf{x}_k, k, \mathbf{c})\|^2 \right]. \tag{4}$$

The construction of $\mathbf{c}$, the masking strategy, and the inference procedure are detailed in Appendix A.2.

For each adjacent pair (σ_k, σ_{k+1}), let $s_{\text{src}} = s_W^{(k)}$ and $s_{\text{tgt}} = s_1^{(k+1)}$. We construct a masked sequence of length H starting from s_{src} and denoise it to obtain a full trajectory, from which we select the stitching terminus as the state with the highest cosine similarity to s_{tgt}:

$$L = \arg \max_{i \in \{1, \ldots, H\}} \frac{\langle \hat{s}_i, s_{\text{tgt}} \rangle}{\|\hat{s}_i\| \, \|s_{\text{tgt}}\|}. \tag{5}$$

The model generates forward and backward trajectories:

$$\begin{aligned} \tau_{fd} &= (s_{\text{src}}, \hat{s}_1, \ldots, \hat{s}_L, s_{\text{tgt}}, s_2^{(k+1)}, \ldots, s_{H-L-1}^{(k+1)}), \\ \tau_{bk} &= (s_{W-H+L+2}^{(k)}, \ldots, s_{W-1}^{(k)}, s_{\text{src}}, \hat{s}'_1, \ldots, \hat{s}'_L, s_{\text{tgt}}). \end{aligned} \tag{6}$$

From τ_{fd} and τ_{bk}, we extract the intermediate state sequences $\mathbf{f} = (\hat{s}_1, \ldots, \hat{s}_L)$ and $\mathbf{b} = (\hat{s}'_1, \ldots, \hat{s}'_L)$. The forward sequence $\mathbf{f}$ is conditioned on s_{src} and the subsequent states of σ_{k+1}, while the backward sequence $\mathbf{b}$ is conditioned on s_{tgt} and the historical states of σ_k. Sharing the same boundary states but different contexts, they jointly capture bidirectional information. The final bridge is a convex combination:

$$\mathbf{Fused}_t = w_t \mathbf{f}_t + (1 - w_t) \mathbf{b}_t, \quad t = 1, \ldots, L, \tag{7}$$

where $w_t = 0.5 + 0.5 \cdot \frac{t-1}{L-1}$ increases monotonically from 0.5 to 1, assigning increasingly higher weight to the forward trajectory near s_{tgt}. This asymmetric fusion captures bidirectional context while suppressing conflicting signals.

4.3 Action and Reward Completion

The fused states lack actions and rewards. We train two MLPs on $\mathcal{D}$: an inverse dynamics model f_ψ predicting $a_t = f_\psi(s_t, s_{t+1})$, and a reward predictor f_ϕ estimating $r_t = f_\phi(s_t, a_t)$. Applying them to the bridge segments yields $\tau_{\text{bridge}} = \{(s_t, a_t, r_t)\}$, which are then concatenated with the source fragments to form the complete synthetic trajectory τ_{syn}. Both models are trained with MSE loss (see Appendix A.3 for details).

4.4 TRAJECTORY FILTER

To ensure physical plausibility, we filter the generated bridge segments τ_{bridge} with two checks. The source fragments are not filtered, as they are real transitions from $\mathcal{D}$. First, a dynamics model f_ω verifies $\|\hat{s}_{t+1} - s_{t+1}\|^2 \leq \delta$ for all t. Second, a reward discriminator $r_D(s, a, s') \in (0, 1)$ from $\mathcal{D}$ (Luo et al., 2023) ensures $r_D(s_t, a_t, s_{t+1}) > r_{\min}$ for all t. Trajectories failing either check are discarded, preventing compounding errors and ensuring the augmented data aligns with high-value behavior. Training details of f_ω and r_D are provided in Appendix A.3.

5 EXPERIMENTS

5.1 EXPERIMENTAL SETUP

We evaluate MFTS on the standard D4RL benchmark (Fu et al., 2020), comparing against sequence-modeling baselines: RvS (Emmons et al., 2022), Decision Transformer (DT) (Chen et al., 2021), QDT (Yamagata et al., 2023), and DiffStitch+DT (Li et al., 2024). These baselines are either transformer-based or use a sequential decision-making formulation, enabling a focused comparison with MFTS. We report results on 9 continuous Gym tasks across three environments (HalfCheetah, Hopper, Walker2d) and three dataset levels (medium, medium-replay, medium-expert). Hyperparameters and network architectures are listed in Appendix A.5.

Table 1: Normalized scores of different methods. Average and standard deviation scores are reported over 3 seeds. The best and the second-best results of each setting are marked in **bold** and underlined, respectively. *Baseline results are taken from original papers. Standard deviation was not reported for RvS in the original paper.

Dataset	Environment	RvS*	DT	QDT*	DiffStitch+DT	Ours
Medium-Expert	HalfCheetah	**92.2**	84.9 ± 0.6	-	86.7 ± 0.7	$\underline{87.5} \pm 1.4$
Medium-Expert	Hopper	101.7	$\underline{104.2} \pm 3.6$	-	97.9 ± 5.4	**104.8** ± 2.4
Medium-Expert	Walker2d	106.0	107.1 ± 1.4	-	$\underline{108.1} \pm 0.5$	**108.7** ± 0.1
Medium	HalfCheetah	41.6	42.0 ± 0.6	42.3 ± 0.4	$\underline{42.4} \pm 0.2$	**42.5** ± 0.2
Medium	Hopper	60.2	64.4 ± 2.6	**66.5** ± 6.3	61.3 ± 1.0	$\underline{65.1} \pm 2.1$
Medium	Walker2d	$\underline{71.7}$	71.0 ± 1.0	67.1 ± 3.2	58.4 ± 12.1	**73.8** ± 1.1
Medium-Replay	HalfCheetah	$\underline{38.0}$	34.7 ± 2.7	35.6 ± 0.5	37.0 ± 2.1	**39.4** ± 0.5
Medium-Replay	Hopper	$\underline{73.5}$	63.8 ± 7.0	52.1 ± 20.3	40.9 ± 31.0	**81.5** ± 4.1
Medium-Replay	Walker2d	60.6	63.6 ± 4.0	58.2 ± 5.1	$\underline{63.6} \pm 1.9$	**72.4** ± 2.1
Average		$\underline{71.7}$	70.6	-	66.3	**75.1**

5.2 MAIN RESULTS

Table 1 summarizes the results. MFTS achieves the highest average normalized score (75.1) among all methods, outperforming DT by 6.4% and DiffStitch+DT by 13.3% in relative terms. The gains are most pronounced on medium-replay datasets, where MFTS improves upon DT by 13.5%, 27.7%, and 13.8% on HalfCheetah, Hopper, and Walker2d, respectively. This is expected: medium-replay datasets are highly suboptimal, and thus benefit most from stitching, directly supporting our core hypothesis. On medium datasets, MFTS surpasses DT by up to 3.9% on Walker2d. Notably, DiffStitch+DT exhibits severe instability on Hopper-medium-replay, whereas MFTS maintains high performance with low variance (81.5 ± 4.1). The high variance of DiffStitch+DT likely stems from its one-directional generation, which lacks sufficient conditioning when the two trajectories are far apart, yielding dynamics-inconsistent transitions. Overall, these results demonstrate that MFTS effectively stitches high-quality decision sequences from noisy, suboptimal experience.

5.3 ABLATION ANALYSIS

Fusion Mechanism. To validate our time-aware asymmetric fusion, we compare it against three alternatives on three Medium-Replay tasks (HalfCheetah, Hopper, Walker2d): **Forward Only** ($w_t = 1$), **Backward Only** ($w_t = 0$), and **Averaged** ($w_t = 0.5$). As reported in Table 2, our method

Table 2: Ablation study on fusion strategies across Medium-Replay tasks. We report the mean and standard deviation over 3 seeds. The best result for each environment is highlighted in bold.

Environment	Forward Only	Backward Only	Averaged	Ours (Time-Aware)
HalfCheetah-Medium-Replay	35.8 ± 1.2	36.2 ± 1.0	37.5 ± 0.8	$\mathbf{39.4 \pm 0.5}$
Hopper-Medium-Replay	70.3 ± 6.7	68.1 ± 5.9	65.5 ± 3.8	$\mathbf{81.5 \pm 4.1}$
Walker2d-Medium-Replay	69.2 ± 2.8	68.5 ± 3.0	70.1 ± 2.2	$\mathbf{72.4 \pm 2.1}$
Average	58.4	57.6	57.7	**64.4**

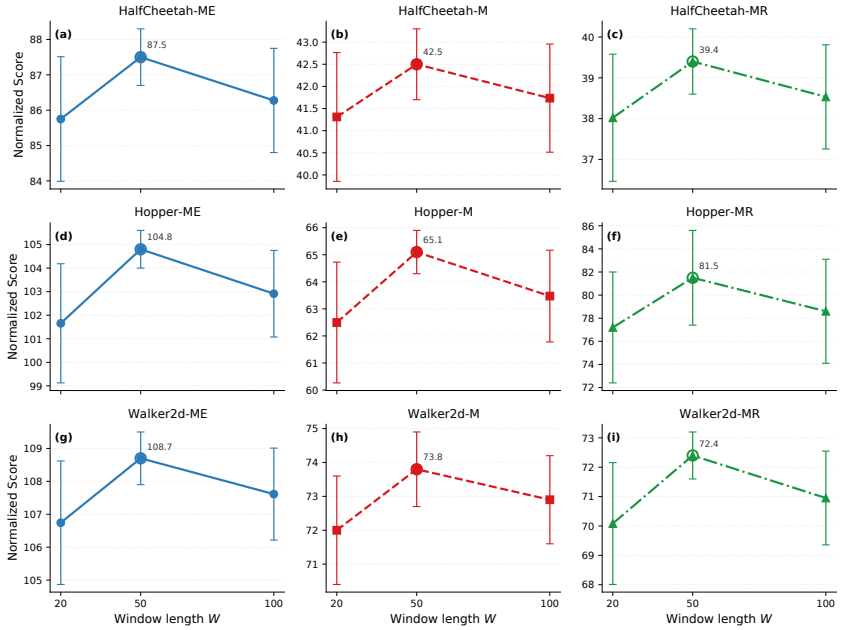

Figure 3: Sensitivity analysis of window length W across nine D4RL settings. Rows correspond to HalfCheetah, Hopper, and Walker2d; columns correspond to Medium-Expert (ME), Medium (M), and Medium-Replay (MR). Markers denote mean normalized scores over 3 seeds (± 1 std.).

consistently outperforms all alternatives, with the largest margin on Hopper-Medium-Replay (81.5 vs. 70.3 for Forward Only, 68.1 for Backward Only, and 65.5 for Averaged). The single-direction baselines degrade most on Hopper, indicating that relying on only forward or backward generation fails to capture the bidirectional context needed for stable transitions. Notably, Averaged performs even worse than Forward Only (65.5 vs. 70.3), suggesting that naive symmetric combination introduces conflicting signals. In contrast, our asymmetric weighting, which monotonically increases the forward weight along the horizon, selectively integrates reliable information from each direction based on proximity to boundary states, preserving temporal coherence. This confirms that *asymmetric* weighting, rather than simple averaging, is key to successful bidirectional fusion.

Effect of Window Length W. The fragment window length W controls the granularity of behavioral primitives used for stitching. To examine its sensitivity, we vary $W \in \{20, 50, 100\}$ across all nine D4RL settings, keeping the remaining hyperparameters identical to the main experiments. Results are summarized in Figure 3.

As shown in Figure 3, $W = 50$ consistently achieves the best performance. When $W = 20$, each fragment carries weaker local semantics and more boundaries must be bridged, which increases the burden on generation and filtering. When $W = 100$, fragments become closer to long contiguous trajectory segments, reducing the flexibility of multi-source composition and diminishing the stitching benefit. The inverted-U trend appears across all three environments and dataset types, indicating

Table 3: Effect of synthetic-to-original data ratio on normalized scores across Medium tasks. We report the mean and standard deviation over 3 seeds. The best result for each environment is highlighted in bold.

Environment	**No Aug (DT)**	8:1 ($\rho = 0.125$)	4:1 ($\rho = 0.25$)	2:1 ($\rho = 0.5$)	1:1 ($\rho = 1.0$)
HalfCheetah-Medium	42.0 ± 0.6	41.1 ± 0.5	$\mathbf{42.5 \pm 0.2}$	41.9 ± 0.4	41.3 ± 0.5
Hopper-Medium	64.4 ± 2.6	60.9 ± 2.0	$\mathbf{65.1 \pm 2.1}$	57.3 ± 4.5	61.3 ± 4.2
Walker2d-Medium	71.0 ± 1.0	71.6 ± 1.2	$\mathbf{73.8 \pm 1.1}$	72.0 ± 1.5	71.5 ± 1.8
Average	59.1	57.9	**60.5**	57.1	58.0

that a moderate window length is robust across tasks. Accordingly, we adopt $W = 50$ as the default in all main experiments.

Effect of Data Ratio. To investigate the sensitivity of MFTS to the synthetic-to-original data ratio, we conduct an ablation on three Medium tasks (HalfCheetah, Hopper, Walker2d) with $\rho \in \{0.125, 0.25, 0.5, 1.0\}$ (8:1, 4:1, 2:1, 1:1) and report vanilla DT (No Aug) as reference. As shown in Table 3, the 4:1 ratio ($\rho = 0.25$) consistently achieves the highest average normalized score (60.5), outperforming both weaker and heavier augmentation. This trend is most pronounced on Hopper-Medium, where 4:1 (65.1) exceeds 1:1 (61.3), while 2:1 collapses to 57.3, even below vanilla DT (64.4). We attribute this to two competing factors: (i) *stitching benefit*, which grows with more synthetic data, and (ii) *compounding model errors* from the auxiliary inverse dynamics and reward models, which accumulate as synthetic data increases. On the unstable Hopper task, the latter dominates beyond the 4:1 point, whereas HalfCheetah and Walker2d are more tolerant. Overall, a modest augmentation ratio (4:1) suffices to unlock the stitching capability of DT without introducing excessive estimation errors, and we adopt it as the default.

Trajectory-Level Stitching Gain Analysis. We introduce a trajectory-level metric, *Stitching Gain*, defined as the difference in discounted cumulative return between a synthesized trajectory and its best source fragment. We randomly generate $M = 500$ synthetic trajectories by stitching $N = 2$ fragments sampled from the expert buffer $\mathcal{F}$, and evaluate all returns over a fixed horizon H in the ground-truth MuJoCo simulator. The gain is calculated as:

$$\text{Gain}(\tau_{syn}) = R(\tau_{syn}) - \max\big(R(\sigma_1), R(\sigma_2)\big). \tag{8}$$

As a naive baseline, we replace the diffusion bridge with linear interpolation in the raw state space (denoted **Linear-Interp**), reusing the same auxiliary models and evaluation pipeline. Specifically, given boundary states s_{src} and s_{tgt}, it generates intermediate states via $\tilde{s}_t = (1 - \frac{t}{L+1})s_{\text{src}} + \frac{t}{L+1}s_{\text{tgt}}$ for $t = 1, \ldots, L$. Table 4 reports the results. Across all evaluated settings, MFTS achieves consistently positive gains and outperforms Linear-Interp. Linear-Interp yields negative gains overall, because linear interpolation in Euclidean space produces physically invalid states that violate the non-linear MuJoCo dynamics, leading to early termination. The success ratio exceeds 70% on all Medium-Replay settings, demonstrating the robustness of MFTS and providing direct, trajectory-level evidence that it composes partial successes into globally superior trajectories.

Figure 4 visualizes one concrete stitch on Hopper-medium-replay, comparing MFTS with Linear-Interp under identical boundary states. We plot instantaneous reward, forward velocity, and torso height, with shaded regions marking the two source fragments and the generated bridge.

For MFTS (Figure 4(a)–(b)), forward velocity increases gradually across the bridge, torso height stays within the stable range, and the trajectory completes with a return exceeding both source fragments. For Linear-Interp (Figure 4(c)–(d)), torso height drops below the stability threshold shortly after entering the interpolated segment, and the episode terminates early. This illustrates why the two methods differ: the diffusion bridge maintains physically consistent intermediate states, whereas linear interpolation does not. Overall, the case confirms that MFTS produces executable, value-improving bridges, and that the filtering mechanism discards unreliable synthetic trajectories.

Table 4: Trajectory-level stitching gain over 9 D4RL settings. Average gain (std) of 500 synthesized trajectories over their best source fragment.

Environment	Dataset	**Linear-Interp**	**MFTS (Ours)**	
		Avg. Gain (↑)	Avg. Gain (↑)	Success Ratio
HalfCheetah	Medium-Replay	-4.2 ± 3.1	12.0 ± 2.5	76.2%
	Medium	-2.0 ± 2.8	8.5 ± 3.0	70.5%
	Medium-Expert	-0.5 ± 1.5	2.5 ± 1.8	55.3%
Hopper	Medium-Replay	-12.5 ± 5.2	22.4 ± 4.1	82.1%
	Medium	-8.0 ± 4.5	10.5 ± 3.5	73.4%
	Medium-Expert	-15.0 ± 6.0	1.2 ± 2.1	51.2%
Walker2d	Medium-Replay	-6.8 ± 3.8	12.8 ± 3.2	74.6%
	Medium	-6.0 ± 3.2	9.0 ± 2.8	68.9%
	Medium-Expert	-4.0 ± 2.5	3.0 ± 2.0	58.1%

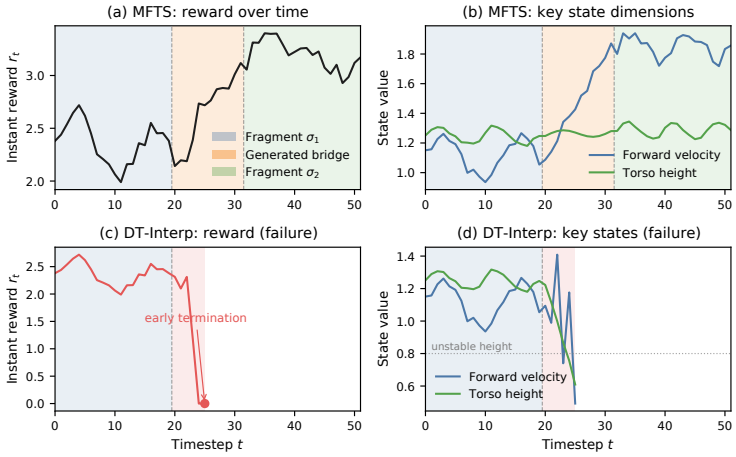

Figure 4: Case study on Hopper-medium-replay. (a)–(b): successful MFTS sample. (c)–(d): Linear-Interp failure under identical boundary states. Shaded regions denote source fragments σ_1 and σ_2, and generated bridge (failed bridge in red).

6 Conclusion

We introduce Multi-Fragment Trajectory Stitching (MFTS), a data augmentation framework that endows Decision Transformer with trajectory stitching ability without relying on Bellman completeness. MFTS combines multiple high-return fragments by bridging them with diffusion-generated forward and backward state trajectories, fused through a time-aware asymmetric blending strategy and filtered for dynamics consistency. On the D4RL benchmark, MFTS achieves the best average score with particularly large gains on medium-replay datasets. Ablation studies and trajectory-level stitching gain analysis further confirm that the asymmetric fusion, moderate augmentation ratio, and diffusion bridge are all essential.

These results demonstrate that trajectory stitching can be recast as a data augmentation problem, providing a practical path to DP-style behavior in offline RL without value bootstrapping. Two limitations remain. First, MFTS depends on the quality of the pre-trained diffusion model and auxiliary models, whose errors can accumulate as the synthetic-to-original ratio grows. Second, our evaluation is restricted to MuJoCo locomotion tasks. Future work includes adaptive fragment selection, learning fusion weights, and extending MFTS to higher-dimensional and real-world robotic domains.

AI USE STATEMENT

Generative AI tools were used to aid or polish the writing of this paper. Specifically, we employed generative AI tools to improve the language, grammar, readability and clarity of the manuscript. All AI-generated or AI-assisted outputs were critically reviewed, verified, and revised by the authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

We adhere to the ICLR Code of Ethics. This work does not involve human or animal subjects, and no identifiable personal information was used in developing this paper. Therefore, there are no privacy or security concerns. We are committed to maintaining integrity and transparency throughout the research process.

REPRODUCIBILITY STATEMENT

We have made every effort to ensure that the methodology presented in this work is reproducible. The code is provided in the supplementary material as a ZIP archive and will be publicly released upon publication. Full descriptions of the methods and experimental setup are provided in both the main text and the appendix to facilitate transparent evaluation and replication of our results.

REFERENCES

Anurag Ajay, Yilun Du, Abhi Gupta, Joshua Tenenbaum, Tommi Jaakkola, and Pulkit Agrawal. Is conditional generative modeling all you need for decision-making?, 2023. URL https://arxiv.org/abs/2211.15657.

Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 15084–15097. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/7f489f642a0ddb10272b5c31057f0663-Paper.pdf.

Scott Emmons, Benjamin Eysenbach, Ilya Kostrikov, and Sergey Levine. Rvs: What is essential for offline rl via supervised learning?, 2022. URL https://arxiv.org/abs/2112.10751.

Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4RL: datasets for deep data-driven reinforcement learning. *CoRR*, abs/2004.07219, 2020. URL https://arxiv.org/abs/2004.07219.

Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning, 2021. URL https://arxiv.org/abs/2106.06860.

Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration, 2019. URL https://arxiv.org/abs/1812.02900.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor, 2018. URL https://arxiv.org/abs/1801.01290.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models, 2024. URL https://arxiv.org/abs/2301.04104.

Nicklas Hansen, Xiaolong Wang, and Hao Su. Temporal difference learning for model predictive control, 2022. URL https://arxiv.org/abs/2203.04955.

Haoran He, Chenjia Bai, Kang Xu, Zhuoran Yang, Weinan Zhang, Dong Wang, Bin Zhao, and Xuelong Li. Diffusion model is an effective planner and data synthesizer for multi-task reinforcement learning, 2023. URL https://arxiv.org/abs/2305.18459.

Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. In *Advances in Neural Information Processing Systems 29, vol. 7: 30th annual conference on Neural Information Processing Systems 2016, 5-10 December 2016, Barcelona, Spain*, pp. 4572–4580, Barcelona, 2016.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 6840–6851. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/4c5bcfec8584af0d967f1ab10179ca4b-Paper.pdf.

Matthew Thomas Jackson, Michael Tryfan Matthews, Cong Lu, Benjamin Ellis, Shimon Whiteson, and Jakob Foerster. Policy-guided diffusion, 2024. URL https://arxiv.org/abs/2404.06356.

Michael Janner, Yilun Du, Joshua B. Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis, 2022. URL https://arxiv.org/abs/2205.09991.

Rahul Kidambi, Aravind Rajeswaran, Praneeth Netrapalli, and Thorsten Joachims. Morel : Model-based offline reinforcement learning, 2021. URL https://arxiv.org/abs/2005.05951.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning, 2021. URL https://arxiv.org/abs/2110.06169.

Aviral Kumar, Justin Fu, George Tucker, and Sergey Levine. Stabilizing off-policy q-learning via bootstrapping error reduction, 2019. URL https://arxiv.org/abs/1906.00949.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning, 2020. URL https://arxiv.org/abs/2006.04779.

Guanghe Li, Yixiang Shan, Zhengbang Zhu, Ting Long, and Weinan Zhang. Diffstitch: Boosting offline reinforcement learning with diffusion-based trajectory stitching, 2024. URL https://arxiv.org/abs/2402.02439.

Cong Lu, Philip J. Ball, Yee Whye Teh, and Jack Parker-Holder. Synthetic experience replay. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA, 2023. Curran Associates Inc.

Fan-Ming Luo, Tian Xu, Xingchen Cao, and Yang Yu. Reward-consistent dynamics models are strongly generalizable for offline reinforcement learning, 2023. URL https://arxiv.org/abs/2310.05422.

Rémi Munos. Error bounds for approximate value iteration. *AAAI Conference on Artificial Intelligence*, 2005.

Ashvin Nair, Abhishek Gupta, Murtaza Dalal, and Sergey Levine. Awac: Accelerating online reinforcement learning with offline datasets, 2021. URL https://arxiv.org/abs/2006.09359.

Yunpeng Qing, Yixiao Chi, Shuo Chen, Shunyu Liu, Kelu Yao, Sixu Lin, Litao Liu, and Changqing Zou. Bitrajdiff: Bidirectional trajectory generation with diffusion models for offline reinforcement learning, 2026. URL https://arxiv.org/abs/2506.05762.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In Francis Bach and David Blei (eds.), *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 2256–2265, Lille, France, 07–09 Jul 2015. PMLR. URL https://proceedings.mlr.press/v37/sohl-dickstein15.html.

Renhao Wang, Kevin Frans, Pieter Abbeel, Sergey Levine, and Alexei A. Efros. Prioritized generative replay, 2025. URL https://arxiv.org/abs/2410.18082.

Yuanfu Wang, Chao Yang, Ying Wen, Yu Liu, and Yu Qiao. Critic-guided decision transformer for offline reinforcement learning, 2023. URL https://arxiv.org/abs/2312.13716.

Taku Yamagata, Ahmed Khalil, and Raul Santos-Rodriguez. Q-learning decision transformer: Leveraging dynamic programming for conditional sequence modelling in offline rl, 2023. URL https://arxiv.org/abs/2209.03993.

Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. Mopo: Model-based offline policy optimization, 2020. URL https://arxiv.org/abs/2005.13239.

Junjie Zhang, Jiafei Lyu, Xiaoteng Ma, Jiangpeng Yan, Jun Yang, Le Wan, and Xiu Li. Uncertainty-driven trajectory truncation for data augmentation in offline reinforcement learning, 2023. URL https://arxiv.org/abs/2304.04660.

Zhaoyi Zhou, Chuning Zhu, Runlong Zhou, Qiwen Cui, Abhishek Gupta, and Simon Shaolei Du. Free from bellman completeness: Trajectory stitching via model-based return-conditioned supervised learning, 2023. URL https://arxiv.org/abs/2310.19308.

A Appendix

A.1 Bellman Completeness and MFTS

We provide a non-formal discussion of Bellman completeness and clarify how MFTS alleviates the need for it, following the analysis of Zhou et al. (2023) and Munos (2005).

Bellman completeness. Let $\mathcal{T}$ denote the Bellman optimality operator,

$$(\mathcal{T}Q)(s,a) = R(s,a) + \gamma\, \mathbb{E}_{s' \sim P(\cdot|s,a)} \left[\max_{a'} Q(s',a') \right].$$

A function class $\mathcal{F}$ is said to satisfy *Bellman completeness* if

$$\max_{Q \in \mathcal{F}} \min_{Q' \in \mathcal{F}} \|Q' - \mathcal{T}Q\| = 0,$$

i.e., applying the Bellman operator to any function in the class keeps the result inside the same class (or arbitrarily close to it). This condition is stronger than realizability, which only requires $Q^* \in \mathcal{F}$, and it is a standard sufficient condition for the convergence of value-based methods with function approximation (Munos, 2005; Zhou et al., 2023).

Why value-based methods rely on it. Dynamic programming (DP)-based methods such as CQL (Kumar et al., 2020) and IQL (Kostrikov et al., 2021) iteratively apply $\mathcal{T}$ to a learned Q-function. If $\mathcal{T}Q \notin \mathcal{F}$, the bootstrapped target falls outside the representable function class, and the approximation error at each iteration can compound. Bellman completeness rules out this source of error and is therefore a sufficient condition for stable value iteration. As shown by Zhou et al. (2023), the absence of Bellman completeness may not be fixable by simply increasing model size; it is a property of the function class relative to the MDP.

Why DT neither satisfies nor requires it. Decision Transformer (DT) (Chen et al., 2021) does not maintain a Q-function and does not apply $\mathcal{T}$. It is trained with a supervised sequence-modeling objective whose policy space is confined to the support of the offline data distribution. Consequently, DT never needs Bellman completeness, but it also cannot perform trajectory stitching: any globally optimal trajectory that combines locally optimal segments lies outside the data support and cannot be recovered by supervised learning alone (Zhou et al., 2023).

How MFTS alleviates the requirement. MFTS does not introduce a Q-function or a Bellman backup. Instead, it expands the effective support of the offline dataset by synthesizing new trajectories that stitch together high-return fragments, and then trains DT on the augmented dataset with the same supervised objective. Because no value iteration is performed, the algorithm does not rely on Bellman completeness. At the same time, the augmented trajectories provide DT with compositional examples that would otherwise be absent from the data, effectively granting DT the stitching behavior of DP without requiring the assumptions that DP depends on.

A.2 State Diffusion Model

MFTS uses a conditional DDPM for state sequence generation. The general forward process and the unconditional denoising objective follow the standard formulation in Section 2, and the conditional training objective is given in Eq. equation 4. Here we detail the MFTS-specific components.

Let $\mathbf{x} = (s_1, \ldots, s_H) \in \mathbb{R}^{H \times d_s}$ denote a state sub-trajectory of length H. The denoising network $\epsilon_{\phi_s}(\mathbf{x}_k, k, \mathbf{c})$ is conditioned on

$$\mathbf{c} = \big(s_{\text{src}},\ s_{\text{tgt}},\ \text{ctx}(\sigma_k),\ \text{ctx}(\sigma_{k+1})\big),$$

where s_{src} and s_{tgt} are the boundary states, and $\text{ctx}(\cdot)$ denotes the local context extracted from the two adjacent fragments. At training time, we sample real state segments from $\mathcal{D}$, construct the condition $\mathbf{c}$, and apply random masks to the input to simulate partial observations, so that the model learns to reconstruct masked states from boundary and context information.

The network architecture, noise schedule, denoising steps, generation horizon, and other diffusion-related hyperparameters are summarized in Table 6.

A.3 Auxiliary Models

Inverse dynamics and reward predictors. The inverse dynamics model f_ψ and reward predictor f_ϕ are trained with MSE losses:

$$\mathcal{L}_\psi = \mathbb{E}_{(s_t, s_{t+1}) \sim \mathcal{D}} \left[\| f_\psi(s_t, s_{t+1}) - a_t \|^2 \right], \tag{9}$$

$$\mathcal{L}_\phi = \mathbb{E}_{(s_t, a_t) \sim \mathcal{D}} \left[\| f_\phi(s_t, a_t) - r_t \|^2 \right]. \tag{10}$$

Both are implemented as three-layer MLPs with ReLU activations (see Table 6).

One-step dynamics model. The one-step dynamics model f_ω is used for the dynamics consistency check. It is an MLP trained on $\mathcal{D}$ to fit one-step transitions with an ℓ_2 loss:

$$\mathcal{L}_{f_\omega} = \mathbb{E}_{(s_t, a_t, s_{t+1}) \sim \mathcal{D}} \big\| s_{t+1} - f_\omega(s_t, a_t) \big\|^2. \tag{11}$$

After training, f_ω is frozen and only used to compute the prediction error for hard-threshold filtering during stitching.

Dynamics reward model. The dynamics reward model provides a second, complementary filter that assesses whether a transition is consistent with the true environment dynamics. Following Luo et al. (2023) and inspired by GAIL (Ho & Ermon, 2016), we jointly learn an environment dynamics model $P_\theta(s_{t+1}|s_t, a_t)$ and a discriminator $D_\omega(s_t, a_t, s_{t+1})$. The former generates fake next states $s'_{\text{fake}} \sim P_\theta(\cdot|s, a)$, while the latter distinguishes real from generated transitions. The discriminator is trained to maximize

$$\begin{aligned} \ell_D(\omega) =& \mathbb{E}_{(s,a,s') \sim \mathcal{D}} \big[\log D_\omega(s, a, s') \big] \\ &+ \mathbb{E}_{(s,a) \sim \mathcal{D},\, s' \sim P_\theta(\cdot|s,a)} \big[\log \big(1 - D_\omega(s, a, s')\big) \big], \end{aligned} \tag{12}$$

and in practice we minimize the gradient-penalized loss

$$\mathcal{L}_D(\omega) = -\ell_D(\omega) + \lambda \cdot \|\nabla_{D_\omega} \ell_D(\omega)\|^2, \tag{13}$$

where $\lambda > 0$ is a penalty coefficient. The dynamics model P_θ is updated via PPO using $-\log(1 - D_\omega(s, a, s'))$ as the immediate reward, which is equivalent to minimizing

$$\mathbb{E}_{(s,a) \sim \mathcal{D}} \Big[\mathbb{E}_{s' \sim P_\theta(\cdot|s,a)} \big[\log \big(1 - D_\omega(s, a, s')\big) \big] \Big]. \tag{14}$$

To reduce training variance, we periodically store discriminator parameter snapshots into an ensemble $\{D_{\omega_i}\}_{i=1}^L$, replacing the earliest snapshot in a FIFO manner when the capacity is exceeded. The final dynamics reward is the ensemble-aggregated signal

$$r_D(s, a, s') = -\log \left(1 - \frac{1}{L} \sum_{i=1}^{L} D_{\omega_i}(s, a, s') \right). \tag{15}$$

This signal is not a replacement for the task reward; it serves only as a transition credibility score for filtering.

Dual filtering. For each candidate synthetic trajectory τ_{bridge}, we compute $r_D(s_t, a_t, s_{t+1})$ for every transition. If any transition violates

$$r_D(s_t, a_t, s_{t+1}) \geq r_{\min}, \tag{16}$$

the trajectory is discarded, where $r_{\min}$ is a preset threshold. Combined with the one-step consistency check, this forms a dual filter: the former verifies one-step prediction consistency, while the latter verifies high dynamics credibility.

Algorithm 2 Training dynamics reward with a discriminator ensemble

Require: Offline dataset $\mathcal{D}$, dynamics model P_θ, discriminator D_ω, ensemble capacity L, penalty coefficient λ

1: Initialize snapshot queue $\mathcal{E} \leftarrow \emptyset$
2: **for** training iteration $t = 1, 2, \ldots$ **do**
3: Sample real transition (s, a, s') from $\mathcal{D}$
4: Sample $s'_{\text{fake}} \sim P_\theta(\cdot|s, a)$ to form a fake transition
5: Update D_ω via Eq. equation 13
6: Update P_θ with PPO using $-\log(1 - D_\omega(s, a, s'))$ as reward
7: Push current D_ω snapshot into $\mathcal{E}$; if $|\mathcal{E}| > L$, pop the earliest
8: **end for**
9: Define r_D via Eq. equation 15
10: **return** r_D

A.4 Experimental Details

Datasets. We evaluate on the D4RL benchmark (Fu et al., 2020). For each environment (HalfCheetah, Hopper, Walker2d), we use three dataset levels: *medium*, *medium-replay*, and *medium-expert*. The dataset statistics follow the standard D4RL protocol, and the number of transitions for each dataset is listed in Table 5. The three dataset types are constructed as follows:

- **Medium**: 1M timesteps generated by a medium policy trained with Soft Actor-Critic (SAC) (Haarnoja et al., 2018) and early-stopped at roughly one-third of expert performance.
- **Medium-Replay**: the replay buffer recorded during the training of the medium policy, containing all transitions observed before it reaches the medium level of performance.
- **Medium-Expert**: 1M timesteps from the medium policy concatenated with 1M timesteps from an expert policy.

Table 5: Dataset statistics for the Gym-MuJoCo tasks used in our experiments. The number of transitions follows the standard D4RL benchmark (Fu et al., 2020).

Environment	Dataset	# Transitions
Hopper	hopper-medium	10^6
	hopper-medium-replay	200,920
	hopper-medium-expert	2×10^6
HalfCheetah	halfcheetah-medium	10^6
	halfcheetah-medium-replay	101,000
	halfcheetah-medium-expert	2×10^6
Walker2d	walker2d-medium	10^6
	walker2d-medium-replay	100,930
	walker2d-medium-expert	2×10^6

Evaluation protocol. We follow the standard D4RL evaluation protocol. Each method is evaluated over 3 seeds, and for each seed we report the normalized score, defined as

$$\text{Normalized Score} = 100 \times \frac{\text{Score} - \text{Random Score}}{\text{Expert Score} - \text{Random Score}},$$

where the *Random Score* is the average return of an agent taking actions uniformly at random over 100 episodes, and the *Expert Score* is the average return of a domain-specific expert policy. For the Gym-MuJoCo tasks, the expert score corresponds to a SAC agent. Following the D4RL convention, a normalized score of 0 corresponds to a random policy, and 100 corresponds to the expert policy.

For each environment, we report the mean and standard deviation over 3 seeds. We evaluate the policy at the end of training, using the same evaluation protocol across all methods to ensure a fair comparison. When reproducing the DT and DiffStitch+DT baselines, we keep the training and evaluation setup identical to our method, with the only difference being the data augmentation strategy.

A.5 Hyperparameters and Network Architectures

We keep the Decision Transformer (DT) architecture and training hyperparameters identical to the baseline, including network width, context length, optimizer, and learning rate, so that performance differences can be attributed to the proposed data augmentation. The main shared hyperparameters of MFTS are listed in Table 6, and task-specific thresholds and buffer sizes are reported in Table 7.

The state diffusion model is implemented as a Temporal U-Net. The inverse dynamics model f_ψ is a three-layer MLP. The dynamics model f_ω and reward predictor f_ϕ share the same ensemble network, which simultaneously predicts the next state and reward.

Table 6: Main shared hyperparameters of MFTS.

Module	Hyperparameter	Value
Fragment Extraction	Fragment window length W	50
	Number of fragments N	$\{2, 3, 4, 5\}$
	Discount factor γ	0.99
	Maximum trajectory length	1000
Diffusion Model	Denoising network	Temporal U-Net
	Channel width / multipliers	128 / $(1, 4, 8)$
	Generation horizon H / denoising steps	100 / 100
	Noise prediction target	ϵ
	Noise schedule	Cosine, small offset $s = 0.008$
	Bridge length search range $[L_{\min}, L_{\max}]$	$[10, 40]$
	Condition dropout / guidance weight	0.25 / 1.2
MLP Models	f_ψ (inverse dynamics)	3-layer MLP, hidden 256, ReLU
	f_ω, f_ϕ (dynamics and reward)	hidden $[200]^4$, ensemble size 7, num_elite 5
	Activation for f_ω, f_ϕ	Swish
Filtering	Dynamics consistency threshold δ	3
	Dynamics reward similarity threshold	Default 0.70 (Hopper: 0.60, see Table 7)
Training	Batch size	32 (effective 64 with grad-accum=2)
	Learning rate	$32 / 2 \times 10^{-4}$
	Training steps	1×10^6
	Original:Augmented	4:1

A.6 Decision Transformer Configuration

We keep the Decision Transformer architecture and training hyperparameters identical to the original DT paper (Chen et al., 2021). The key hyperparameters for OpenAI Gym experiments are listed in Table 8.

Table 7: Task-specific hyperparameters. r_{test} is the target return percentile for conditional generation. n_{opt} and k_{top} control the expert buffer size and candidate fragment pool size, respectively. The dynamics reward similarity threshold is 0.60 on Hopper and 0.70 on other environments.

Environment	Dataset	r_{test}	n_{opt}	k_{top}	Sim. thr.
HalfCheetah	Medium-Expert	0.92	4×10^4	6×10^5	0.70
	Medium-Replay	0.35	4×10^3	6×10^4	0.70
	Medium	0.50	2×10^4	3×10^5	0.70
Hopper	Medium-Expert	0.90	4×10^4	6×10^5	0.60
	Medium-Replay	0.85	8×10^3	1.2×10^5	0.60
	Medium	0.90	2×10^4	3×10^5	0.60
Walker2d	Medium-Expert	0.80	4×10^4	6×10^5	0.70
	Medium-Replay	0.60	4×10^3	6×10^4	0.70
	Medium	0.92	2×10^4	3×10^5	0.70

Table 8: Decision Transformer hyperparameters for OpenAI Gym experiments, following Chen et al. (2021).

Hyperparameter	Value
Number of layers	3
Number of attention heads	1
Embedding dimension	128
Nonlinearity	ReLU
Batch size	64
Context length K	20 (HalfCheetah, Hopper, Walker2d)
Return-to-go conditioning	6000 (HalfCheetah), 3600 (Hopper), 5000 (Walker2d)
Dropout	0.1
Learning rate	10^{-4}
Grad norm clip	0.25
Weight decay	10^{-4}
Learning rate decay	Linear warmup for first 10^4 steps
Optimizer	AdamW
Training steps	10^5 gradient steps