Shortcut Trajectory Planning for Efficient Offline Reinforcement Learning

Anonymous authors
Paper under double-blind review

Abstract

Diffusion-based trajectory planners have shown strong performance in offline reinforcement learning, but their iterative denoising process often incurs high inference cost. Consistency-based planners reduce the number of sampling steps, yet they typically rely on a two-stage teacher–student distillation pipeline that increases training cost and may introduce instability. We propose Shortcut Trajectory Planning (STP), an offline model-based reinforcement learning framework that incorporates shortcut models as efficient trajectory generators. STP trains a conditional shortcut trajectory model in a single stage, supports adjustable one-step and few-step inference through step-size conditioning, and selects candidate plans using a critic augmented with feasibility-aware correction. Across standard D4RL benchmarks, including locomotion, navigation, manipulation, and dexterous control tasks, STP achieves strong performance while simplifying the training pipeline for fast generative planning.

1 Introduction

Diffusion-based trajectory planners have shown strong performance in offline reinforcement learning by modeling behavior trajectories as generative distributions and using planning-time sampling for decision making (Janner et al., 2022; Ajay et al., 2023). However, their iterative denoising process often requires many sampling steps, which makes them computationally expensive for real-time or high-frequency control. Recent consistency-based planners, including Consistency Planning (CP) (Wang et al., 2024a) and Consistency Trajectory Planning (CTP) (Wang et al., 2026), address this limitation by replacing multi-step diffusion sampling with consistency-based generation (Song et al., 2023; Ding & Jin, 2024; Kim et al., 2024b). These methods substantially reduce inference cost while maintaining competitive planning performance.

Despite their fast inference, CP and CTP still rely on a two-stage training pipeline. They first train an Elucidated Diffusion Model (EDM) (Karras et al., 2022) as a teacher and then distill it into a consistency model (Song et al., 2023; Luo et al., 2023). This teacher-student procedure increases the total training cost and introduces an additional source of instability: the final planner depends on both a successfully trained teacher and a successfully distilled student. Such instability is particularly undesirable in offline RL, where learning is already challenging due to multimodal trajectory distributions and distributional shift (Fujimoto et al., 2019; Kumar et al., 2020; Yu et al., 2020; Kidambi et al., 2020).

In this paper, we propose *Shortcut Trajectory Planning* (STP), a shortcut-model-based trajectory planner for offline model-based RL. Shortcut models (Frans et al., 2025) learn step-size-conditioned updates and can therefore support one-step, few-step, and multi-step generation within a single network. Unlike consistency-based planners that require teacher-student distillation, shortcut models can be trained directly in a single stage, simplifying the learning pipeline while preserving the ability to perform fast sampling. This makes shortcut models a natural fit for trajectory planning, where different tasks may require different trade-offs between planning speed and trajectory quality.

We evaluate STP on standard offline RL benchmarks (Fu et al., 2020) and compare it with diffusion-based, consistency-based, and CTM-based planners. Our experiments show that STP achieves competitive planning performance while reducing both inference cost and training complexity. These results suggest that

shortcut models provide a practical alternative to distillation-based generative planners for efficient offline reinforcement learning.

2 Related Work

Shortcut Models. Shortcut models were recently introduced as an efficient generative modeling framework for accelerating diffusion sampling (Frans et al., 2025). Instead of learning only infinitesimal denoising dynamics, shortcut models condition the network on the desired step size, enabling one-step, few-step, and multi-step generation within a single model. Compared with standard diffusion models (Ho et al., 2020; Song et al., 2021; Karras et al., 2022), this formulation reduces the number of required sampling steps. Compared with consistency models (Song et al., 2023) and consistency trajectory models (Kim et al., 2024a), it avoids a multi-stage teacher-student distillation pipeline and can be trained in a single stage. This combination of training simplicity and adjustable inference cost makes shortcut models attractive for decision-making problems where computation and control precision must be balanced.

Shortcut Models in Reinforcement Learning. The use of shortcut models in reinforcement learning remains relatively unexplored. The closest related work is Scalable Offline Reinforcement Learning (SORL), which applies shortcut models as generative policy representations in offline RL and uses a learned Q-function as an inference-time verifier (Espinosa-Dice et al., 2026). Dreamer 4 also adopts a shortcut forcing objective to train an efficient world model for imagination-based reinforcement learning (Hafner et al., 2025). These works show that shortcut-style objectives can support efficient decision making, but they use shortcut models either as policies or world models.

In contrast, this paper studies shortcut models for offline model-based reinforcement learning with trajectory planning. Rather than parameterizing a policy or predicting future latent states, we use shortcut models as trajectory generators inside a planning framework. This setting is closely related to Consistency Planning (CP) (Wang et al., 2024a) and Consistency Trajectory Planning (CTP) (Wang et al., 2026), but replaces their distillation-based consistency pipeline with a single-stage shortcut-based planner. The goal is to retain efficient inference while simplifying training and reducing the instability introduced by teacher-student distillation.

3 Background on Shortcut Models

Shortcut models (Frans et al., 2025) are efficient generative models built on flow matching (Lipman et al., 2023; Liu et al., 2023). They retain the continuous transport view of flow matching, but additionally condition the model on the desired step size, enabling one-step, few-step, and multi-step generation within a single network. This section briefly introduces the flow-matching notation used in this paper and then summarizes the shortcut formulation.

Flow Matching. Flow matching learns a velocity field that transports samples from a simple noise distribution to the data distribution. Let $\mathbf{x}_0 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ be a noise sample and $\mathbf{x}_1 \sim \mathcal{D}$ be a data sample. For $t \in [0, 1]$, define the linear interpolation

$$\mathbf{x}_t = (1-t)\mathbf{x}_0 + t\mathbf{x}_1, \qquad \mathbf{v}_t = \mathbf{x}_1 - \mathbf{x}_0. \tag{1}$$

A flow-matching model $f_\theta(\mathbf{x}_t, t)$ is trained to predict $\mathbf{v}_t$:

$$\mathcal{L}_{\mathrm{FM}}(\theta) = \mathbb{E}_{\mathbf{x}_0, \mathbf{x}_1, t}\left[\|f_\theta(\mathbf{x}_t, t) - (\mathbf{x}_1 - \mathbf{x}_0)\|_2^2\right]. \tag{2}$$

At inference time, samples are generated by numerical integration,

$$\mathbf{x}_{t+d} = \mathbf{x}_t + d f_\theta(\mathbf{x}_t, t), \tag{3}$$

where $d = 1/K$ for K sampling steps. Although effective, standard flow matching may require many small steps; using large steps can introduce substantial discretization error.

Shortcut Models. Shortcut models address this issue by learning finite-step updates directly. Instead of predicting only an infinitesimal velocity, a shortcut model conditions on the current sample, time, and requested step size:

$$S_\theta : (\mathbf{x}_t, t, d) \mapsto \Delta\mathbf{x}, \qquad \mathbf{x}'_{t+d} = \mathbf{x}_t + d\, S_\theta(\mathbf{x}_t, t, d). \tag{4}$$

When d is small, this recovers the usual flow-matching behavior; when d is large, the model learns to approximate a larger jump along the generative trajectory. The central training signal is a recursive self-consistency constraint: a shortcut of size $2d$ should match the composition of two shortcuts of size d,

$$S(\mathbf{x}_t, t, 2d) = \frac{1}{2} S(\mathbf{x}_t, t, d) + \frac{1}{2} S(\mathbf{x}'_{t+d}, t+d, d). \tag{5}$$

Thus, the same model can be sampled with different inference budgets by choosing $d = 1/K$:

$$\mathbf{x}_{t+d} = \mathbf{x}_t + d\, S_\theta(\mathbf{x}_t, t, d), \qquad t \leftarrow t + d. \tag{6}$$

Training Objective. Shortcut models combine a flow-matching term with a self-consistency term (Frans et al., 2025). The former grounds the model at the finest scale, while the latter propagates this behavior to larger step sizes:

$$\mathcal{L}_{\mathrm{SC}}(\theta) = \mathbb{E}_{\mathbf{x}_0, \mathbf{x}_1, t, d} \left[\| S_\theta(\mathbf{x}_t, t, 0) - (\mathbf{x}_1 - \mathbf{x}_0) \|_2^2 + \| S_\theta(\mathbf{x}_t, t, 2d) - S_{\mathrm{target}} \|_2^2 \right], \tag{7}$$

where

$$S_{\mathrm{target}} = \frac{1}{2} S_\theta(\mathbf{x}_t, t, d) + \frac{1}{2} S_\theta(\mathbf{x}'_{t+d}, t+d, d). \tag{8}$$

Unlike consistency models and consistency trajectory models, which often rely on teacher-student distillation (Song et al., 2023; Kim et al., 2024a), shortcut models can be trained in a single stage with a single network. This makes them appealing for trajectory planning: they preserve flexible low-step inference while reducing training complexity, which is important for offline RL settings where diffusion-based planners can be costly at inference time (Janner et al., 2022; Ajay et al., 2023).

Implications for planning. For the purposes of offline model-based reinforcement learning, the shortcut formulation (Frans et al., 2025) is particularly appealing because it combines two properties that are desirable in trajectory generation. First, it preserves low-latency inference by allowing generation with a small number of function evaluations, including the one-step case. Second, it permits the sampling budget to be adjusted at test time, so that easier control tasks may use fewer steps while more challenging tasks may benefit from additional refinement. Compared with teacher-student consistency-based methods (Song et al., 2023; Kim et al., 2024a), shortcut models also simplify training by learning the generative shortcut within a single end-to-end optimization procedure. These properties are particularly relevant to generative planning in offline RL, where trajectory-level generative models have been shown to provide expressive planning mechanisms but often suffer from high inference cost (Janner et al., 2022; Ajay et al., 2023). They therefore motivate the use of shortcut models as trajectory planners in the proposed method of this paper.

4 Planning with Shortcut Models

We incorporate shortcut models into offline model-based RL by replacing the distillation-based consistency trajectory generator used in CTP with a single-stage shortcut trajectory model (Frans et al., 2025). Figure 1 provides an overview of the proposed Shortcut Trajectory Planning (STP) framework. The planner generates candidate trajectories using the shortcut model, selects the best trajectory with a learned critic and a feasibility penalty, and recovers the executable action through an inverse dynamics model (Janner et al., 2022; Ajay et al., 2023; Lu et al., 2025).

4.1 Training process

Trajectory representation. As in CTP (Wang et al., 2026), we adopt a trajectory representation with jumping-steps for planning. Given a trajectory τ, the planner models a subsequence of future states as

$$\mathbf{x}_{t_i}(\tau) := (s_k, s_{k+M}, \ldots, s_{k+(H-1)M})_{t_i}. \tag{9}$$

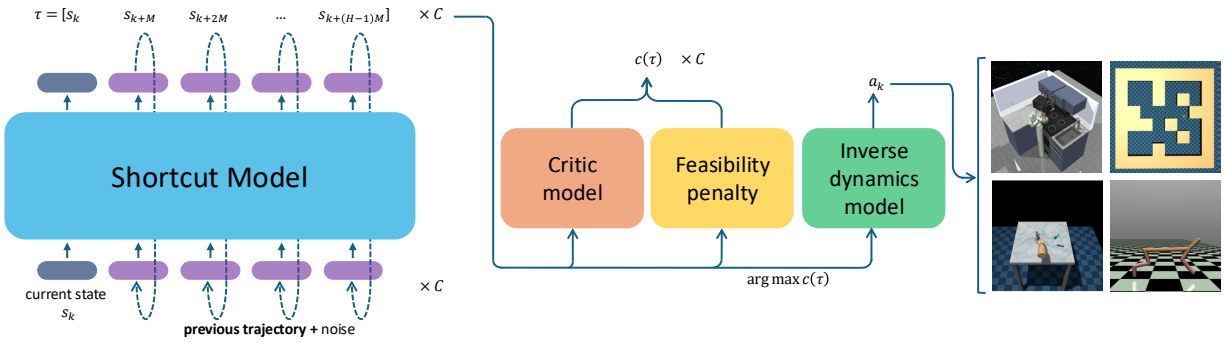


Figure 1: Overview of the proposed Shortcut Trajectory Planning (STP). The shortcut model generates candidate trajectories, which are evaluated by the critic together with a feasibility penalty. The highest-scoring trajectory is converted into the control action using the inverse dynamics model.

Here, k denotes the timestep index within the trajectory τ, H is the planning horizon, and $M \in \mathbb{N}_+$ is the planning stride. Thus, the generated subsequence covers $H \times M$ environment steps in total. This jump-step planning strategy allows the model to plan over a longer effective horizon while keeping the generated trajectory compact (Lu et al., 2025).

Accordingly, $\mathbf{x}_{t_i}(\tau)$ represents a noisy sequence of future states at noise level t_i, where each element corresponds to one stride-separated state in the planned trajectory. During training and inference, the current state s_k is used as the condition, and the shortcut model generates the future state sequence $(s_{k+M}, \ldots, s_{k+(H-1)M})$. Unlike CTP, where $t_i \in [\epsilon, t_N]$ follows the EDM-style noise parameterization (Karras et al., 2022), in this paper we use the flow-matching (Lipman et al., 2023; Frans et al., 2025) notation introduced in Section 3 and represent the noise level by a continuous variable $t \in [0, 1]$.

Stride-based inverse dynamics model. As in CTP, we employ a stride-based inverse dynamics model to recover actions from the planned state sequence. Given the planning stride M, the inverse dynamics model $h_{\boldsymbol{\varphi}}$ takes the state pair (s_k, s_{k+M}) as input and predicts the corresponding action a_k (Agrawal et al., 2016; Pathak et al., 2018; Lu et al., 2025). Accordingly, during inference, the action is recovered from the selected trajectory via

$$a_k = h_{\boldsymbol{\varphi}}(s_k, s_{k+M}).$$

The model is trained on state-action-state tuples sampled from the offline dataset $\mathcal{D}$ by minimizing

$$\mathcal{L}(\boldsymbol{\varphi}) := \mathbb{E}_{(s_k, a_k, s_{k+M}) \sim \mathcal{D}} \left[\| a_k - h_{\boldsymbol{\varphi}}(s_k, s_{k+M}) \|_2^2 \right]. \tag{10}$$

Shortcut trajectory model training. Unlike the two-stage training procedure adopted in CP and CTP, the shortcut trajectory planner is trained in a single stage. Specifically, we parameterize the planner as a conditional shortcut model

$$S_{\boldsymbol{\theta}}(\mathbf{x}_t, t, d \mid s_k), \tag{11}$$

where $\mathbf{x}_t$ denotes the noisy future-state sequence at signal level $t \in [0, 1]$, $d \in (0, 1]$ is the prescribed step size, and s_k is the current state used as the condition. Given $\mathbf{x}_t$, the model predicts a finite-step update along the generative trajectory, and the corresponding next state is defined by

$$\mathbf{x}'_{t+d} = \mathbf{x}_t + d\, S_{\boldsymbol{\theta}}(\mathbf{x}_t, t, d \mid s_k). \tag{12}$$

Following the flow-matching formulation introduced in Section 3, we construct noisy training samples by linearly interpolating between a Gaussian noise sample $\mathbf{x}_0^{\text{noise}} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and a clean trajectory sample $\mathbf{x}_1(\tau)$:

$$\mathbf{x}_t(\tau) = (1 - t)\mathbf{x}_0^{\text{noise}} + t\, \mathbf{x}_1(\tau), \tag{13}$$

where

$$\mathbf{x}_1(\tau) := (s_k, s_{k+M}, \ldots, s_{k+(H-1)M}). \tag{14}$$

At the infinitesimal-step limit, the shortcut reduces to the local flow direction. Accordingly, the model is grounded at the finest scale by the flow-matching objective

$$\mathcal{L}_{\mathrm{FM}}(\boldsymbol{\theta}) = \mathbb{E}_{\tau\sim\mathcal{D},\, \mathbf{x}_0^{\mathrm{noise}}\sim\mathcal{N},\, t\sim p(t)} \left[\left\| S_{\boldsymbol{\theta}}(\mathbf{x}_t, t, 0 \mid s_k) - \left(\mathbf{x}_1(\tau) - \mathbf{x}_0^{\mathrm{noise}}\right) \right\|_2^2 \right]. \tag{15}$$

To extend the model from infinitesimal updates to finite-step trajectory generation, a recursive self-consistency relation across step sizes is introduced. In particular, a shortcut associated with step size $2d$ is required to be consistent with the composition of two successive shortcuts associated with step size d. Let

$$\mathbf{x}'_{t+d} = \mathbf{x}_t + d\, S_{\boldsymbol{\theta}}(\mathbf{x}_t, t, d \mid s_k). \tag{16}$$

Then the bootstrap target for the larger step size $2d$ is defined as

$$S_{\mathrm{target}} = \frac{1}{2} S_{\boldsymbol{\theta}}(\mathbf{x}_t, t, d \mid s_k) + \frac{1}{2} S_{\boldsymbol{\theta}}(\mathbf{x}'_{t+d}, t+d, d \mid s_k). \tag{17}$$

The corresponding self-consistency loss is

$$\mathcal{L}_{\mathrm{SC}}(\boldsymbol{\theta}) = \mathbb{E}_{\tau\sim\mathcal{D},\, \mathbf{x}_0^{\mathrm{noise}}\sim\mathcal{N},\, (t,d)\sim p(t,d)} \left[\left\| S_{\boldsymbol{\theta}}(\mathbf{x}_t, t, 2d \mid s_k) - \mathrm{sg}(S_{\mathrm{target}}) \right\|_2^2 \right], \tag{18}$$

where $\mathrm{sg}(\cdot)$ denotes the stop-gradient operator.

The shortcut trajectory planner is trained by jointly minimizing the flow-matching term and the recursive self-consistency term:

$$\mathcal{L}_{\mathrm{STP}}(\boldsymbol{\theta}) := \mathcal{L}_{\mathrm{FM}}(\boldsymbol{\theta}) + \lambda_{\mathrm{SC}}\, \mathcal{L}_{\mathrm{SC}}(\boldsymbol{\theta}), \tag{19}$$

where λ_{SC} controls the contribution of the self-consistency term.

This training objective directly learns a shortcut (Frans et al., 2025) mapping for conditional trajectory generation without relying on a separately trained teacher model or an additional distillation stage. The flow-matching term anchors the planner to the underlying denoising dynamics at the finest scale, while the recursive self-consistency term propagates this behavior to larger step sizes. As a result, the model can generate candidate trajectories under different inference budgets within a single unified framework.

Critic model training. As in CTP, a critic model is used to rank the candidate trajectories generated by the planner according to their expected returns. Let $V_{\boldsymbol{\alpha}}$ denote the critic network with parameters $\boldsymbol{\alpha}$. The critic takes the denoised trajectory $\mathbf{x}_{t_0}(\tau)$ as input and predicts the accumulated discounted return R_k, defined by

$$R_k = \sum_{h=0}^{t_{\mathrm{end}}} \gamma^h r_{k+h}. \tag{20}$$

The critic is trained on trajectories sampled from the offline dataset $\mathcal{D}$ by minimizing the mean squared error

$$\mathcal{L}_{\mathrm{critic}}(\boldsymbol{\alpha}) = \mathbb{E}_{\tau\sim\mathcal{D}} \left[\left(V_{\boldsymbol{\alpha}}(\mathbf{x}_{t_0}(\tau)) - R_k \right)^2 \right]. \tag{21}$$

At inference time, the critic is used to evaluate the candidate trajectories generated by the shortcut planner, and the trajectory with the highest predicted return is selected for action extraction and execution.

4.2 Inference process

Sampling with shortcut models. At inference time, the shortcut trajectory planner generates future state sequences by iteratively applying the learned shortcut model under a prescribed sampling budget (Frans

et al., 2025). Let K denote the number of shortcut updates used during planning, and let $d = 1/K$ be the corresponding step size. Given the current environmental state s_k, sampling aims to produce a trajectory

$$(s_k, s_{k+M}, \ldots, s_{k+(H-1)M}),$$

where the first state is fixed to the current observation and the remaining states are generated by the planner.

For the first environmental timestep, sampling starts from a Gaussian noise sample and progressively refines it into a trajectory plan. Specifically, an initial noisy trajectory $\mathbf{x}_0^{\text{noise}} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ is drawn, and the planner iteratively applies

$$\mathbf{x}_{t+d} \leftarrow \mathbf{x}_t + d\, S_{\boldsymbol{\theta}}(\mathbf{x}_t, t, d \mid s_k), \tag{22}$$

for $t = 0, d, 2d, \ldots, 1-d$, until the final trajectory sample $\mathbf{x}_1$ is obtained.

For subsequent environmental timesteps, instead of restarting from pure noise, we adopt a warm-start strategy (Janner et al., 2022). More precisely, let $\mathbf{x}_1^{(k-1)}$ denote the denoised trajectory generated at the previous environmental timestep. The planner initializes the next trajectory by perturbing $\mathbf{x}_1^{(k-1)}$ with noise, rather than drawing a completely new sample from the Gaussian prior. The resulting noisy initialization is then refined by the same shortcut sampling procedure in equation 22. In this way, the planner reuses information from the previously generated trajectory and updates it into a new trajectory conditioned on the current state.

This warm-start strategy serves two purposes. First, it reduces redundant computation by avoiding repeated trajectory generation from pure noise at every environmental step. Second, it improves temporal coherence across consecutive planning steps, since the newly generated trajectory remains connected to the previously selected plan. As a result, the resulting state sequences and recovered actions are typically smoother over time, which helps reduce unnecessary zigzag behavior during execution.

After trajectory sampling, C candidate trajectories are generated, ranked by the critic model, and the highest-valued trajectory is selected for action extraction using the stride-based inverse dynamics model.

Plan selection. At each environmental timestep, the shortcut trajectory planner generates C candidate trajectories conditioned on the current state s_k. Each candidate is evaluated by the critic model $V_{\boldsymbol{\alpha}}$, which estimates the expected return of the generated trajectory (Lu et al., 2025). However, the critic is trained only to predict accumulated discounted returns from trajectories in the offline dataset, and therefore does not explicitly account for whether a generated trajectory is physically feasible in the environment. As a result, trajectories that are inconsistent with environment constraints may still receive spuriously high critic scores.

To address this issue, we augment the value-based plan selection rule with a feasibility correction term. Let $\mathbf{x}_{t_0}^{(c)}$ denote the c-th denoised candidate trajectory, and let $\Psi(\mathbf{x}_{t_0}^{(c)})$ denote a feasibility penalty that measures whether the generated trajectory violates known environment constraints. The corrected score used for plan selection is defined as

$$\tilde{V}\left(\mathbf{x}_{t_0}^{(c)}\right) = V_{\boldsymbol{\alpha}}\left(\mathbf{x}_{t_0}^{(c)}\right) - \lambda_{\text{fea}}\, \Psi\left(\mathbf{x}_{t_0}^{(c)}\right), \tag{23}$$

where $\lambda_{\text{fea}} > 0$ controls the strength of the feasibility correction. The best plan is then selected according to

$$c^* = \arg\max_{c \in \{1,\ldots,C\}} \tilde{V}\left(\mathbf{x}_{t_0}^{(c)}\right). \tag{24}$$

For example, as shown in Figure 2, the maze tasks considered in this paper provide a clear illustration of the feasibility correction. In this setting, the feasibility penalty is used to detect whether the generated trajectory overlaps with obstacle regions. Since the agent cannot pass through blocked cells in the true environment, any trajectory that intersects with a wall is infeasible and should not be selected, even if its predicted return is high. The feasibility correction therefore prevents the critic from favoring generated trajectories with high predicted cumulative rewards but unrealizable under the environment dynamics.

Action extraction. After selecting the best trajectory, the action extraction is performed in the same manner as in CTP. Specifically, the first stride transition (s_k, s_{k+M}) is taken from the selected trajectory,

Figure 2: Examples of generated trajectories in the maze environment. Left: a feasible trajectory whose states remain outside obstacle regions. Right: an infeasible trajectory with intermediate states overlapping obstacle cells. The feasibility-corrected score prevents such unrealizable trajectories from being selected.

and the action is recovered by the stride-based inverse dynamics model (Agrawal et al., 2016; Pathak et al., 2018; Lu et al., 2025):

$$a_k = h_{\boldsymbol{\varphi}}(s_k, s_{k+M}). \tag{25}$$

The resulting action a_k is then executed in the environment, after which the planner replans from the newly observed state. The overall procedure is summarized in Algorithm 1.

Algorithm 1 Shortcut Trajectory Planning

Input: Planning horizon H, dataset $\mathcal{D}$, discount factor γ, candidate number C, planning stride M
Initialize: Shortcut planner $S_{\boldsymbol{\theta}}$, inverse dynamics model $h_{\boldsymbol{\varphi}}$, critic $V_{\boldsymbol{\alpha}}$
Calculate discounted returns $R_k = \sum_{h=0}^{\text{end}} \gamma^h r_{k+h}$ for every step k
function TRAINING
    Sample $s_k, s_{k+M}, \ldots, s_{k+(H-1)M}$, $a_k, a_{k+M}, \ldots, a_{k+(H-1)M}$, R_k from $\mathcal{D}$
    Construct clean trajectory $\mathbf{x}_1(\tau) := (s_k, s_{k+M}, \ldots, s_{k+(H-1)M})$
    Sample Gaussian noise $\mathbf{x}_0^{\text{noise}} \sim \mathcal{N}(\mathbf{0}, I)$
    Sample $t \in [0, 1]$ and step size d
    Construct noisy trajectory $\mathbf{x}_t = (1-t)\mathbf{x}_0^{\text{noise}} + t\,\mathbf{x}_1(\tau)$
    Train shortcut planner $S_{\boldsymbol{\theta}}$ using Equation 19
    Train inverse dynamics model $h_{\boldsymbol{\varphi}}$ using Equation 10
    Train critic $V_{\boldsymbol{\alpha}}$ using Equation 21
end function
function PLANNING(s)
    if first environmental timestep **then**
        Generate C candidate plans from Gaussian noise with the first state fixed as s
    else
        Generate C candidate plans using warm-start sampling with the first state fixed as s
    end if
    Select the best plan using the feasibility-corrected critic score
    Generate action with $h_{\boldsymbol{\varphi}}$ using s and the next state in the selected plan
    Execute the action in the environment
end function

5 Experiment

We evaluate Shortcut Trajectory Planning (STP) on standard D4RL offline RL benchmarks (Fu et al., 2020), including Gym MuJoCo locomotion tasks (Todorov et al., 2012; Brockman et al., 2016), Maze2D, AntMaze, Kitchen, and Adroit. These domains cover continuous control, long-horizon navigation, sparse-reward goal reaching, and high-dimensional manipulation. All methods are evaluated in the offline setting without additional environment interaction during training (Levine et al., 2020; Fu et al., 2020).

5.1 Experimental setting

We keep the experimental setting aligned with CTP unless otherwise specified. STP is compared with representative actor-critic and generative planning methods, including Diffusion-QL (D-QL) (Wang et al.,

Table 1: Locomotion results. Prior results are from Ding & Jin (2024); Ajay et al. (2023); Wang et al. (2024a).

Dataset	**Diffuser**	**DD**	**D-QL**	**C-AC**	**CTP**	**STP**
Halfcheetah-mr	37.5 ± 0.5	39.3 ± 4.1	47.8 ± 0.3	**58.7**± 3.9	43.4 ± 0.4	44.7 ± 0.2
Halfcheetah-m	42.8 ± 0.3	49.1 ± 1.0	51.1 ± 0.5	**69.1**± 0.7	50.4 ± 0.1	51.7 ± 0.1
Hopper-mr	93.6 ± 0.4	100.0 ± 0.7	**101.3**± 0.6	99.7 ± 0.5	90.0 ± 1.0	92.6 ± 0.1
Hopper-m	74.3 ± 1.4	79.3 ± 3.6	**90.5**± 4.6	80.7 ± 10.5	83.6 ± 1.3	**97.2** ± 0.3
Walker2d-mr	70.6 ± 1.6	75.0 ± 4.3	**95.5**± 1.5	79.5 ± 3.6	86.9 ± 0.3	89.5 ± 0.2
Walker2d-m	79.6 ± 0.55	82.5 ± 1.4	**87.0**± 0.9	83.1 ± 0.3	85.7 ± 0.2	83.4 ± 0.2
Average	63.8	70.9	**78.9**	78.5	73.3	73.9

Table 2: Maze2D results. Prior results are from Janner et al. (2022); Wang et al. (2024b); Lu et al. (2025); Duan et al. (2026).

Dataset	**MPPI**	**CQL**	**IQL**	**Diffuser**	**CP**	**RACTD**	**D-QL**	**CTP**	**STP**
Maze2D U-Maze	33.2	5.7	47.4	113.9	122.7	125.7	140.6	154.1	**159.3** $\pm$ 2.4
Maze2D Medium	10.2	5.0	34.9	121.5	121.4	130.8	152.0	167.1	**176.9** $\pm$ 1.8
Maze2D Large	5.1	12.5	58.6	123.0	119.5	143.8	186.8	**216.7**	**215.1** $\pm$ 2.9
Average	16.2	7.7	47.0	119.5	121.2	133.4	159.8	179.3	**183.8**

2023), Consistency Actor-Critic (C-AC) (Ding & Jin, 2024), Diffuser (Janner et al., 2022), Decision Diffuser (DD) (Ajay et al., 2023), Consistency Planning (CP) (Wang et al., 2024b), Consistency Trajectory Planning (CTP), Reward-Aware Consistency Trajectory Distillation (RACTD) (Duan et al., 2026), and Lower Expectile Q-learning (LEQ) (Park & Lee, 2025), depending on the benchmark. We report normalized D4RL returns averaged over 150 independent planning seeds (Fu et al., 2020).

STP uses two shortcut sampling steps for AntMaze and three steps for all other benchmarks. For baselines, we follow the sampling budgets in their original implementations: Diffusion-QL uses $N = 5$ denoising steps (Wang et al., 2023), Diffuser uses $N = 20$ steps, and Decision Diffuser uses $N = 40$ steps (Janner et al., 2022; Ajay et al., 2023). Consistency-based baselines are evaluated with their corresponding reported settings. STP generates 30 candidate trajectories per planning step in all experiments except AntMaze-Large, whereas CP and CTP use 50 candidates. Warm-start initialization and feasibility-aware plan selection are applied during STP inference, with additional implementation details provided in Appendix A.

5.2 Experimental Results

As shown in Table 1, STP is evaluated on the medium and medium-replay datasets of three standard D4RL locomotion environments. In contrast to the medium-expert setting, where near-optimal trajectories are included in the offline dataset, the medium and medium-replay datasets mainly contain suboptimal behavior data. These settings are therefore more challenging for offline RL and better reveal whether an algorithm can generate improved trajectories from imperfect demonstrations. Across these benchmarks, STP achieves

Table 3: Kitchen results. Prior results are from Ding & Jin (2024); Ajay et al. (2023).

Dataset	**Environment**	**Diffuser**	**DD**	**D-QL**	**C-AC**	**CTP**	**STP**
Mixed	Kitchen	47.5	65 ± 2.8	62.6 ± 5.1	45.8 ± 1.5	74.5 ± 0.3	**74.7** $\pm$ 0.2
Partial	Kitchen	33.8	57 ± 2.5	60.5 ± 6.9	38.2 ± 1.8	91.2 ± 1.0	**92.5** $\pm$ 1.2
Average	-	40.65	61.0	61.55	42.0	82.85	**83.6**

Table 4: AntMaze results. Prior results are from Wang et al. (2023); Park & Lee (2025).

Dataset	**Environment**	**LEQ**	**D-QL**	**CTP**	**STP**
Diverse	Antmaze-Large	60.2 ± 18.3	56.6 ± 7.6	82.0 ± 3.1	$\mathbf{83.3} \pm 3.0$
Play	Antmaze-Large	62.0 ± 9.9	46.4 ± 8.3	$\mathbf{82.0} \pm 3.1$	81.3 ± 3.2
Diverse	Antmaze-Medium	46.2 ± 23.2	78.6 ± 10.3	86.0 ± 2.8	$\mathbf{88.0} \pm 2.7$
Play	Antmaze-Medium	76.3 ± 17.2	76.6 ± 10.8	83.3 ± 3.0	$\mathbf{88.7} \pm 2.6$
Average	-	61.18	64.6	83.33	**85.33**

Table 5: Adroit expert results. Prior results are from He et al. (2024); Lu et al. (2025).

Environment	**BC**	**BCQ**	**CQL**	**IQL**	**AlignIQL**	**D-QL**	**CTP**	**STP**
Door	34.9	99.0	101.5	103.8	104.6	104.3	104.1	$\mathbf{105.3} \pm 0.4$
Hammer	**125.6**	107.2	86.7	116.3	124.7	55.9	110.5	120.9 ± 2.2
Pen	85.1	114.9	107.0	111.7	116.0	60.9	104.1	$\mathbf{122.0} \pm 4.7$
Relocate	101.3	41.6	95.0	102.7	106.0	108.8	108.8	$\mathbf{108.9} \pm 0.6$
Average	86.7	90.7	97.6	108.6	112.8	82.4	106.9	**114.3**

competitive results and improves the average score from 73.3 with CTP to 73.9. More specifically, STP outperforms CTP on five out of six datasets, including both HalfCheetah tasks, both Hopper tasks, and Walker2d-medium-replay. The improvement is particularly clear on Hopper-medium, where STP achieves the best score among all compared methods. These results indicate that the shortcut trajectory formulation can further enhance the planning quality of CTP while retaining the low-step generative inference advantage of consistency-based methods. Although Diffusion-QL and Consistency-AC still achieve higher scores on some tasks due to their explicit value-based policy optimization, STP remains competitive among trajectory-generation-based planning methods under challenging suboptimal-data settings.

Table 2 shows the performance of STP on the Maze2D benchmark. Despite the sparse-reward and long-horizon nature of this domain, STP achieves the best average performance among all compared methods, with an average normalized score of 183.8. It outperforms CTP on Maze2D-Umaze and Maze2D-Medium, and achieves comparable performance on Maze2D-Large. These results indicate that the proposed shortcut-based trajectory planner is effective for long-horizon planning and further improves over CTP on average. This improvement is also consistent with the intended role of the warm-start strategy and the feasibility-aware plan selection used during inference.

Table 3 presents the performance of STP on the Kitchen benchmark, which evaluates multi-stage manipulation with compositional task structure. STP achieves the best performance on both `kitchen-mixed` and `kitchen-partial`, and further improves over CTP on both datasets. This result indicates that the proposed shortcut-based trajectory planner is effective not only for navigation, but also for compositional robotic manipulation tasks.

Table 4 reports the performance of STP on the AntMaze benchmark, which evaluates goal-conditioned control under sparse rewards, long-horizon planning, and high-dimensional actuation. As shown in Table 4, STP achieves the best average performance among all compared methods. Compared with CTP, STP improves the average score from 83.33 to 85.33. These results indicate that the proposed shortcut-based trajectory planner remains effective in challenging goal-conditioned control tasks that require both long-horizon planning and high-dimensional continuous control.

Table 6: Ablation study on the warm-start strategy in the Maze2D environment. All other hyperparameters are kept unchanged.

Dataset	**Without warm-start**	**With warm-start**
Maze2D U-Maze	119.4 ± 4.4	$\mathbf{159.3} \pm 2.4$
Maze2D Medium	145.7 ± 3.1	$\mathbf{176.9} \pm 1.8$
Maze2D Large	187.2 ± 4.7	$\mathbf{215.1} \pm 2.9$
Average	150.8	**183.8**

We further evaluate STP on the expert dataset from the Adroit benchmark, which consists of dexterous manipulation tasks requiring precise control in high-dimensional action spaces. Quantitative results are reported in Table 5. While STP does not outperform all prior methods on every task, it achieves the best average performance overall. These results suggest that STP is effective in goal-conditioned manipulation scenarios that require precise spatial reasoning and fine-grained control.

Overall, the experimental results across D4RL locomotion, Maze2D, Kitchen, AntMaze, and Adroit consistently demonstrate the effectiveness of STP over diverse offline RL benchmarks. These environments cover substantially different task structures, including long-horizon maze navigation, sparse-reward quadrupedal control, compositional manipulation, and high-dimensional dexterous manipulation. The consistent performance across these heterogeneous domains and dataset types suggests that the effectiveness of STP is not merely due to overfitting to a specific environment, dataset, or task structure. Instead, the results indicate that the shortcut-based trajectory generation mechanism provides a generally effective planning component for offline model-based reinforcement learning.

Compared with CTP, the proposed shortcut-based trajectory planner preserves strong planning performance while achieving improved average results on most domains, indicating that replacing the teacher-student consistency trajectory generator with a single-stage shortcut model does not compromise control quality. These findings suggest that shortcut-based trajectory generation provides an effective and robust alternative for offline model-based reinforcement learning. In the next subsection, we further examine the contribution of individual design choices in STP through ablation studies.

5.3 Ablation study

This section presents ablation studies to analyze the contributions of two key design choices in STP: the warm-start strategy and the feasibility penalty. The former is designed to improve planning efficiency and temporal consistency during online execution, while the latter aims to suppress infeasible trajectories in long-horizon Maze2D planning. For each component, we conduct controlled comparisons by modifying only the target component while keeping the remaining hyperparameters unchanged.

Effect of the warm-start strategy. We first examine the effect of the warm-start strategy introduced in Section 4.2. Warm-start is designed to reuse the trajectory generated at the previous environmental time step, rather than restarting the planning process from pure noise at every step. This design is expected to improve temporal consistency across consecutive planning steps and allow the planner to refine previously generated trajectories instead of regenerating them from scratch.

Tables 6 and 7 report the results on Maze2D and AntMaze, respectively. On Maze2D, warm-start improves the average normalized score from 150.8 to 183.8. On AntMaze, the average score increases from 68.0 to 85.3. The improvement is consistent across all tasks in both domains, indicating that warm-start initialization plays an important role in improving planning performance.

By reusing previous trajectories, warm-start makes the generated plans and extracted actions smoother over time, which reduces unnecessary oscillatory behavior and improves long-horizon execution. The consistent gains observed in both Maze2D and AntMaze suggest that this effect is not limited to relatively simple

Table 7: Ablation study on the warm-start strategy in the AntMaze environment. All other hyperparameters are kept unchanged.

Dataset	**Without warm-start**	**With warm-start**
AntMaze Medium-Diverse	64.7 ± 3.9	$\mathbf{88.0} \pm 2.7$
AntMaze Medium-Play	62.0 ± 4.0	$\mathbf{88.7} \pm 2.6$
AntMaze Large-Diverse	78.0 ± 3.4	$\mathbf{83.3} \pm 3.0$
AntMaze Large-Play	67.3 ± 3.8	$\mathbf{81.3} \pm 3.2$
Average	68.0	**85.3**

Table 8: Ablation study on the feasibility penalty in the Maze2D environment. All other hyperparameters are kept unchanged.

Dataset	**Without feasibility penalty**	**With feasibility penalty**
Maze2D U-Maze	158.5 ± 2.3	$\mathbf{159.3 \pm 2.4}$
Maze2D Medium	175.4 ± 1.6	$\mathbf{176.9 \pm 1.8}$
Maze2D Large	181.9 ± 4.7	$\mathbf{215.1 \pm 2.9}$
Average	171.9	**183.8**

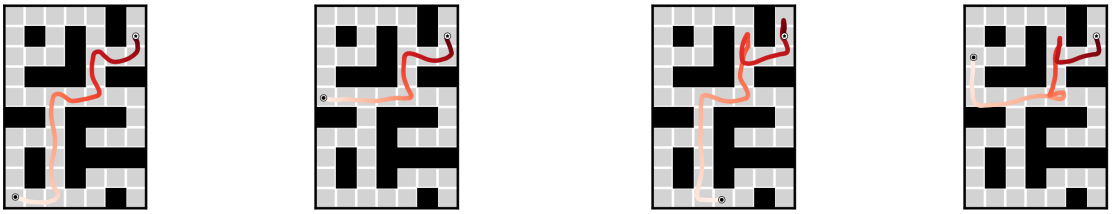
navigation domains, but also extends to more challenging environments with sparse rewards and high-dimensional actuation.

Effect of the feasibility penalty. We then evaluate the effect of the feasibility-aware correction introduced in Section 4.2. The comparison is conducted on the Maze2D benchmark with and without the feasibility penalty, so that the only difference between the compared variants lies in whether infeasible trajectories are explicitly penalized during plan selection.

The results, shown in Table 8, indicate that the feasibility penalty has limited effect on Maze2D U-Maze and Maze2D Medium, but yields a substantial improvement on Maze2D Large. In particular, the score on Maze2D Large increases from 181.9 ± 4.7 to 215.1 ± 2.9, whereas the gains on U-Maze and Medium are comparatively small. These results suggest that the feasibility-aware correction becomes increasingly important as the planning problem becomes more complex.

In relatively simple tasks such as U-Maze and Medium, generated candidate trajectories are less likely to violate environment constraints, and therefore value-based plan selection alone is often sufficient. In contrast, in the more challenging Maze2D Large setting, generated trajectories are more likely to contain infeasible segments that overlap with obstacle regions. Since the critic is trained to predict expected return rather than physical feasibility, such unrealizable trajectories may still receive high scores and be mistakenly selected. By explicitly penalizing trajectories that violate environment constraints, the feasibility penalty improves the quality of plan selection and leads to a clear performance gain on the more complex task.

Figure 3 further provides a qualitative comparison of the executed trajectories on Maze2D-large with and without the feasibility penalty. As shown in the figure, when the feasibility penalty is applied, the agent follows a relatively smooth route from the start position to the goal, with fewer unnecessary detours and fewer interactions with obstacle boundaries. In contrast, without the feasibility penalty, the agent can still eventually reach the goal, but the executed trajectory becomes more tortuous and includes segments that collide with or slide along the maze walls. This suggests that value-based plan selection alone may be sufficient for reaching the goal in some cases, but it does not necessarily guarantee physically feasible or smooth execution. By explicitly accounting for environment constraints, the feasibility penalty helps filter out unrealistic candidate trajectories and improves the executability of the selected plans in complex maze layouts.



(a) With feasibility penalty (b) Without feasibility penalty

Figure 3: Qualitative comparison of executed trajectories on Maze2D-large with and without the feasibility penalty. The feasibility penalty encourages smoother routes with fewer obstacle interactions, while value-based selection alone may produce more tortuous paths and wall collisions.

Overall, the ablation results indicate that the two inference-time designs in STP contribute in complementary ways. The warm-start strategy mainly improves temporal consistency across consecutive planning steps, leading to smoother trajectory refinement and more stable long-horizon execution. In contrast, the feasibility penalty mainly improves plan selection in more complex environments by suppressing unrealizable trajectories that may otherwise be overestimated by the critic. Together, these results support the effectiveness of the proposed inference design in improving the robustness and performance of STP.

6 Discussion

This paper introduced Shortcut Trajectory Planning (STP), a shortcut-model-based trajectory planner for offline reinforcement learning. By replacing the teacher-student distillation pipeline used in consistency-based planners with a single-stage shortcut model, STP simplifies training while preserving efficient few-step trajectory generation and an adjustable inference budget.

Across standard offline RL benchmarks, STP achieves strong performance on sparse-reward navigation, compositional robotic manipulation, high-dimensional goal-conditioned control, and dexterous manipulation. The results show that shortcut-based trajectory generation can match or improve over CTP on average, suggesting that removing distillation does not weaken planning capability. The ablations further indicate that warm-start sampling and feasibility-aware plan selection are complementary: warm-start improves temporal consistency across planning steps, while feasibility correction suppresses unrealizable trajectories during candidate selection.

Overall, STP shows that shortcut models are a practical alternative to distillation-based consistency models for fast generative planning. Its combination of single-stage training, efficient trajectory generation, and robust inference-time selection makes shortcut-based planning a promising direction for offline model-based reinforcement learning.

References

Pulkit Agrawal, Ashvin V Nair, Pieter Abbeel, Jitendra Malik, and Sergey Levine. Learning to poke by poking: Experiential learning of intuitive physics. *Advances in neural information processing systems*, 29, 2016.

Anurag Ajay, Yilun Du, Abhi Gupta, Joshua B. Tenenbaum, Tommi S. Jaakkola, and Pulkit Agrawal. Is conditional generative modeling all you need for decision making? In *The Eleventh International Conference on Learning Representations*, 2023. URL `https://openreview.net/forum?id=sP1fo2K9DFG`.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

Zihan Ding and Chi Jin. Consistency models as a rich and efficient policy class for reinforcement learning. In *The Twelfth International Conference on Learning Representations*, 2024. URL `https://openreview.net/forum?id=v8jdwkUNXb`.

Xintong Duan, Yutong He, Fahim Tajwar, Ruslan Salakhutdinov, J Zico Kolter, and Jeff Schneider. Accelerating diffusion planners in offline RL via reward-aware consistency trajectory distillation. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=hRuTBS07C7.

Nicolas Espinosa-Dice, Yiyi Zhang, Yiding Chen, Bradley Guo, Owen Oertell, Gokul Swamy, Kiante Brantley, and Wen Sun. Scaling offline rl via efficient and expressive shortcut models. *Advances in Neural Information Processing Systems*, 38:2376–2414, 2026.

Kevin Frans, Danijar Hafner, Sergey Levine, and Pieter Abbeel. One step diffusion via shortcut models. In *International Conference on Learning Representations*, volume 2025, pp. 34668–34684, 2025.

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

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

Danijar Hafner, Wilson Yan, and Timothy Lillicrap. Training agents inside of scalable world models. *arXiv preprint arXiv:2509.24527*, 2025.

Longxiang He, Li Shen, Junbo Tan, and Xueqian Wang. AlignIQL: Policy Alignment in Implicit Q-Learning through Constrained Optimization. *CoRR*, abs/2405.18187, 2024. URL https://doi.org/10.48550/arXiv.2405.18187.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Michael Janner, Yilun Du, Joshua Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. In *International Conference on Machine Learning*, pp. 9902–9915. PMLR, 2022.

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. *Advances in Neural Information Processing Systems*, 35:26565–26577, 2022.

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

Dongjun Kim, Chieh-Hsin Lai, Wei-Hsiang Liao, Naoki Murata, Yuhta Takida, Toshimitsu Uesaka, Yutong He, Yuki Mitsufuji, and Stefano Ermon. Consistency trajectory models: Learning probability flow ODE trajectory of diffusion. In *Proceedings of The Twelfth International Conference on Learning Representations*, 2024a. URL https://openreview.net/forum?id=ymjI8feDTD.

Dongjun Kim, Chieh-Hsin Lai, WeiHsiang Liao, Naoki Murata, Yuhta Takida, Toshimitsu Uesaka, Yutong He, Yuki Mitsufuji, and Stefano Ermon. Consistency trajectory models: Learning probability flow ode trajectory of diffusion. In *International Conference on Learning Representations*, volume 2024, pp. 44493–44525, 2024b.

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

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

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=PqvMRDCJT9t.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *The Eleventh International Conference on Learning Representations (ICLR)*, 2023.

Haofei Lu, Dongqi Han, Yifei Shen, and Dongsheng Li. What makes a good diffusion planner for decision making? In *Proceedings of The Thirteenth International Conference on Learning Representations*, 2025.

Simian Luo, Yiqin Tan, Longbo Huang, Jian Li, and Hang Zhao. Latent consistency models: Synthesizing high-resolution images with few-step inference. *arXiv preprint arXiv:2310.04378*, 2023.

Kwanyoung Park and Youngwoon Lee. Model-based Offline Reinforcement Learning with Lower Expectile Q-Learning. In *Proceedings of The Thirteenth International Conference on Learning Representations*, 2025.

Deepak Pathak, Parsa Mahmoudieh, Guanghao Luo, Pulkit Agrawal, Dian Chen, Yide Shentu, Evan Shelhamer, Jitendra Malik, Alexei A Efros, and Trevor Darrell. Zero-shot visual imitation. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pp. 2050–2053, 2018.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 4195–4205, 2023.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=PxTIG12RRHS`.

Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. In *Proceedings of the 40th International Conference on Machine Learning*, pp. 32211–32252, 2023.

Emanuel Todorov, Tom Erez, and Yuval Tassa. MuJoCo: A physics engine for model-based control. In *Proceedings of IEEE/RSJ international conference on intelligent robots and systems*, pp. 5026–5033. IEEE, 2012.

Guanquan Wang, Takuya Hiraoka, and Yoshimasa Tsuruoka. Planning with consistency models for model-based offline reinforcement learning. In *Deployable RL: From Research to Practice @ Reinforcement Learning Conference*, 2024a. URL `https://openreview.net/forum?id=vxnMxWJRSM`.

Guanquan Wang, Takuya Hiraoka, and Yoshimasa Tsuruoka. Planning with consistency models for model-based offline reinforcement learning. *Transactions on Machine Learning Research*, 2024b.

Guanquan Wang, Takuya Hiraoka, and Yoshimasa Tsuruoka. Consistency trajectory planning: High-quality and efficient trajectory optimization for offline model-based reinforcement learning. *Transactions on Machine Learning Research*, 2026. ISSN 2835-8856. URL `https://openreview.net/forum?id=RVGkT9ISVf`.

Zhendong Wang, Jonathan J Hunt, and Mingyuan Zhou. Diffusion policies as an expressive policy class for offline reinforcement learning. In *The Eleventh International Conference on Learning Representations*, 2023. URL `https://openreview.net/forum?id=AHvFDPi-FA`.

Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Y Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. Mopo: Model-based offline policy optimization. *Advances in Neural Information Processing Systems*, 33:14129–14142, 2020.

A Additional Details on Shortcut Trajectory Planning

This appendix provides additional experimental and implementation details for Shortcut Trajectory Planning (STP), including the sources of baseline results, model architecture, training hyperparameters, planning settings, inference settings, and ablation configurations.

A.1 Baseline Results and Sources

For the benchmark results reported in the main paper, we use previously published scores when the corresponding methods have already been evaluated on the same standardized D4RL tasks.

For the Gym MuJoCo locomotion tasks, the results of Diffuser and Decision Diffuser (DD) are taken from Table 1 of (Ajay et al., 2023). The results of Diffusion-QL (D-QL) and Consistency-Actor-Critic (C-AC) are taken from Table 2 of (Ding & Jin, 2024). The results of Consistency Planning (CP) and Consistency Trajectory Planning (CTP) are taken from their corresponding experimental settings and evaluation protocols.

For the Maze2D tasks, the results of MPPI, CQL, IQL, and Diffuser are taken from Table 1 of (Janner et al., 2022). The results of Diffusion-QL are taken from Table 1 of (Lu et al., 2025), and the results of RACTD are taken from Table 2 of (Duan et al., 2026). The results of CP and CTP are taken from their corresponding experiments under the same benchmark setting.

For the AntMaze, Kitchen, and Adroit tasks, baseline results are reported using the same sources and evaluation protocol as the CTP baseline. When a baseline result is reproduced using an existing implementation rather than quoted from a published table, we explicitly follow the implementation and evaluation settings described in the corresponding experiment section.

A.2 Implementation Details

Model architecture. We use Diffusion Transformer blocks with the adaLN-Zero architecture (Peebles & Xie, 2023) as the network backbone for the shortcut trajectory model. The critic and inverse dynamics models follow the same architectural design as the CTP baseline, so that the comparison between CTP and STP focuses on the trajectory generator rather than auxiliary model differences.

Shortcut trajectory model training. The shortcut trajectory model is trained in a single stage using the joint objective described in Section 4.1. Unlike CP and CTP, no separately trained diffusion or EDM teacher model is required. The training objective consists of a flow-matching term and a recursive self-consistency term across step sizes. The model is optimized using Adam.

Critic and inverse dynamics training. The critic model and the stride-based inverse dynamics model are trained using the same procedure as the CTP baseline. The critic is trained to predict the accumulated discounted return of a denoised candidate trajectory, and the inverse dynamics model is trained to recover the executable action from the first stride transition of the selected trajectory.

Inference settings. Unless otherwise specified, STP uses a small number of shortcut sampling steps during inference. We use $N = 2$ shortcut sampling steps for the AntMaze tasks and $N = 3$ steps for all other tasks, including the D4RL locomotion, Maze2D, Kitchen, and Adroit tasks. These choices are made to ensure a consistent and fair comparison within each benchmark group. In preliminary experiments, we found that some relatively simple environments can already achieve near-saturated performance with fewer shortcut sampling steps; for example, Walker2d and Hopper in the D4RL locomotion benchmark can perform well with $N = 2$, and AntMaze-medium can achieve near-saturated scores with $N = 1$. Nevertheless, we use a unified sampling budget within the corresponding experiment groups to avoid introducing additional task-specific tuning when reporting results in the same table.

Planning horizon and stride. The planning horizon H and stride M are set as follows:

- **MuJoCo locomotion tasks:** $H = 4$, $M = 1$.

- **Kitchen tasks:** $H = 32$, $M = 4$.
- **Maze2D tasks:** $H = 32$, $M = 15$.
- **AntMaze tasks:** $H = 40$, $M = 25$.
- **Adroit tasks:** $H = 32$, $M = 2$.

These settings are kept consistent with the CTP baseline to enable a controlled comparison between CTP and STP. In particular, the same horizon and stride are used so that the main difference lies in the generative trajectory model: CTP uses a distilled consistency trajectory model, whereas STP uses a single-stage shortcut trajectory model.

A.3 Ablation Study Details

Warm-start strategy. For the warm-start ablation, we compare STP with and without reusing the previously generated denoised trajectory as the initialization for the next environmental time step. In the variant without warm-start, each planning step starts from an independently sampled Gaussian noise trajectory. In the variant with warm-start, the first planning step starts from a pure Gaussian noise trajectory $\mathbf{x}_0 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. For subsequent environmental time steps, the denoised trajectory from the previous step is perturbed by Gaussian noise and reused as the initialization. Specifically, we initialize the shortcut sampling process from an intermediate noise level $t = 0.3$ by mixing the previous denoised trajectory with Gaussian noise, i.e.,

$$\mathbf{x}_{0.3}^{(k)} = 0.7\,\mathbf{x}_0^{\text{noise}} + 0.3\,\mathbf{x}_1^{(k-1)}, \qquad \mathbf{x}_0^{\text{noise}} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}),$$

where $\mathbf{x}_1^{(k-1)}$ denotes the denoised trajectory generated at the previous environmental time step. Shortcut sampling is then applied from $t = 0.3$ to $t = 1$ conditioned on the current state. All other hyperparameters, including the number of candidate trajectories, sampling steps, planning horizon, stride, critic model, and inverse dynamics model, are kept unchanged.

Feasibility penalty. For the feasibility-penalty ablation, we compare value-based plan selection with and without the feasibility correction term introduced in Section 4.2. In the variant without the feasibility penalty, candidate trajectories are selected only according to the critic score. In the variant with the feasibility penalty, the corrected score subtracts a penalty for trajectories that violate known environment constraints. We set the feasibility-penalty coefficient to $\lambda_{\text{fea}} = 5$ in all experiments using this correction. This comparison is mainly conducted on Maze2D, where obstacle regions provide explicit feasibility constraints and infeasible generated trajectories can be directly detected.

Evaluation protocol. All ablation experiments use the same evaluation protocol as the main experiments. For each task, the reported score is the average normalized return over multiple evaluation episodes. Unless otherwise specified, the model checkpoints, critic, inverse dynamics model, planning horizon, stride, and sampling budget are kept fixed across the compared variants.