Accelerating Diffusion Models in Offline RL via Reward-Aware Consistency Trajectory Distillation

Anonymous Author(s)
Affiliation
Address
`email`

Abstract

Although diffusion models have achieved strong results in decision-making tasks, their slow inference speed remains a key limitation. While the consistency model offers a potential solution, its applications to decision-making often struggle with suboptimal demonstrations or rely on complex concurrent training of multiple networks. In this work, we propose a novel approach to consistency distillation for offline reinforcement learning that directly incorporates reward optimization into the distillation process. Our method enables single-step generation while maintaining higher performance and simpler training. Empirical evaluations on the Gym MuJoCo benchmarks and long horizon planning demonstrate that our approach can achieve an 6.8% improvement over previous state-of-the-art while offering up to $142\times$ speedup over diffusion counterparts in inference time.

1 Introduction

Recent advances in diffusion models have demonstrated their remarkable capabilities across various domains [Song et al., 2020a, Karras et al., 2022, Liu et al., 2023, Chi et al., 2023, Janner et al., 2022], including decision-making tasks in reinforcement learning (RL). These models excel particularly in capturing multi-modal behavior patterns [Janner et al., 2022, Chi et al., 2023, Ajay et al., 2022] and achieving strong out-of-distribution generalization [Duan et al., 2025, Block et al., 2023], making them powerful tools for complex decision-making scenarios. However, their practical deployment faces a significant challenge: the computational overhead of the iterative sampling procedures, which requires numerous denoising steps to generate high-quality outputs.

To address this limitation, various diffusion acceleration techniques have been proposed, including ordinary or stochastic differential equations (ODE or SDE) solvers with flexible step sizes [Song et al., 2020a, Lu et al., 2022, Karras et al., 2022], sampling step distillation [Song et al., 2023, Kim et al., 2023] and improved noise schedules and parametrizations [Salimans and Ho, 2022, Song and Dhariwal, 2024]. In particular, consistency distillation [Song et al., 2023] has emerged as one of the most promising solutions for image generation, in which a many-step diffusion model serves as a teacher to train a student consistency model that achieves comparable performance while enabling faster sampling through a single-step or few-step generation process.

This breakthrough has sparked considerable interest in applying consistency distillation to decision-making tasks. However, current applications either adopt a behavior cloning approach [Lu et al., 2024, Prasad et al., 2024, Wang et al., 2024] or integrate few-step diffusion based samplers in actor-critic frameworks [Chen et al., 2023, Ding and Jin, 2023, Li et al., 2024]. While promising, these approaches face inherent challenges: behavior cloning performs well only with expert demonstrations but struggles with suboptimal data (e.g., median-quality replay buffers), while the actor-critic paradigm

Submitted to the Workshop on Structured Probabilistic Inference and Generative Modeling (SPIGM 2025). Do not distribute.

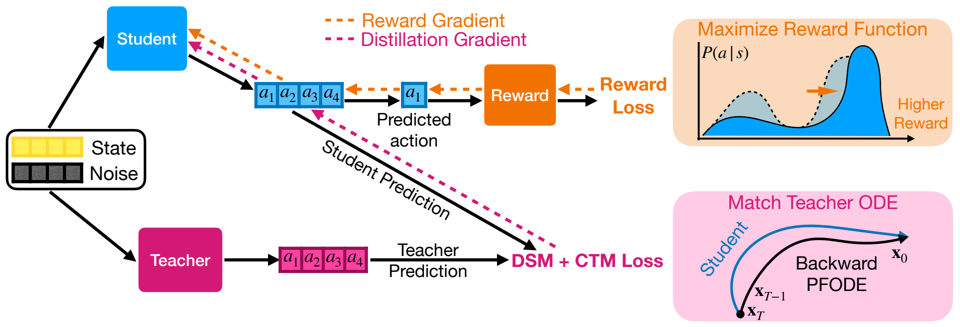

Figure 1: Overview of Reward Aware Consistency Trajectory Distillation (RACTD). We incorporate reward guidance with consistency trajectory distillation to train a student model that can generate actions with high rewards with only one denoising step.

introduces concurrent training of multiple networks and sensitive hyperparameters, exacerbating training complexity, instability, and computational overhead.

This raises an important question: can we develop a more effective approach to consistency distillation specifically tailored for offline RL? We address this challenge by introducing a novel method that directly incorporates reward optimization into the consistency distillation process. Our approach begins with a pre-trained unconditional diffusion policy and augments the standard consistency trajectory distillation [Kim et al., 2023] with an explicit reward objective. The vanilla consistency trajectory distillation helps the student model cover the diverse behavior patterns learned by the teacher. The additional objective encourages the student consistency model to sample actions that yield higher rewards, effectively steering the model toward selecting optimal trajectories from the multi-modal distributions captured by the teacher diffusion model.

Another key advantage of our method lies in its training simplicity. The reward model can be trained independently from the teacher diffusion model and the distillation process, avoiding the complexity of concurrent multi-network training present in actor-critic methods. Our method also eliminates the need for training noise-aware reward models, unlike existing guided diffusion sampling approaches [Janner et al., 2022]. By integrating reward optimization directly into the distillation process, our method achieves superior performance, enables efficient single-step generation, and maintains straightforward training procedures.

We demonstrate the performance and sampling efficiency of our RACTD on the suboptimal heterogenous D4RL Gym-MuJoCo benchmark and challenging long-horizon planning task Maze2d [Fu et al., 2020]. Our method demonstrates both superior performance and substantial sampling efficiency compared to existing approaches, achieving an 6.8% improvement compared to existing state-of-the-art (SOTA) and a 142-fold reduction in sampling time.

Our contributions include: (1) We propose a novel reward-aware consistency distillation method for offline RL that enables single-step generation while achieving superior performance, (2) We demonstrate that our approach enables decoupled training without the complexity of concurrent multi-network optimization or noise aware reward model training, (3) Through comprehensive experiments on multi-modal suboptimal dataset and long-horizon planning tasks, we show that our method achieves 6.8% improvement over prior SOTA while achieving up to $142\times$ speedup. We will publicly release our code upon acceptance of this paper.

2 Background

2.1 Problem Setting

In this paper we consider the classic setting of offline reinforcement learning, where the goal is to learn a policy π to generate actions that maximize the expected cumulative discounted reward in a Markov decision process (MDP). A MDP is defined by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, R, \gamma)$, where $\mathcal{S}$ is the set of possible states $\boldsymbol{s} \in \mathcal{S}$, $\mathcal{A}$ is the set of actions $\boldsymbol{a} \in \mathcal{A}$, $\mathcal{P}(\boldsymbol{s}' \mid \boldsymbol{s}, \boldsymbol{a})$ is the transition dynamics, $R(\boldsymbol{s}, \boldsymbol{a})$ is a reward function, and $\gamma \in [0, 1]$ is a discount factor. In offline RL, we further assume that the agent can no longer interact with the environment and is restricted to learning from a size M static dataset $\mathcal{D} = \{\boldsymbol{\tau}_i\}_{i=1}^{M}$, where $\boldsymbol{\tau} = (\boldsymbol{s}_0, \boldsymbol{a}_0, r_0, \boldsymbol{s}_1, \boldsymbol{a}_1, r_1, \ldots, \boldsymbol{s}_H, \boldsymbol{a}_H, r_H)$ represents a rollout of episode horizon H collected by following a behavior policy π_β.

Mathematically, we want to find a policy π^* that

$$\pi^* = \arg\max_{\pi} \mathbb{E}_{\boldsymbol{\tau}\sim\pi}\Big[\sum_{n=0}^{H} \gamma^n \, R\big(\boldsymbol{s}_n, \boldsymbol{a}_n\big)\Big] \tag{1}$$

subject to the constraint that all policy evaluation and improvement must rely on $\mathcal{D}$ alone.

2.2 Diffusion Models

Diffusion models generate data by learning to reverse a gradual noise corruption process applied to training examples. Given a clean data sample $\boldsymbol{x}_0$, we define $\boldsymbol{x}_t$ for $t \in [0, T]$ as increasingly noisy versions of $\boldsymbol{x}_0$. The forward (or noising) process is commonly formulated as an Itô stochastic differential equation (SDE):

$$\mathrm{d}\boldsymbol{x} = f(\boldsymbol{x}, t)\mathrm{d}t + g(t)\mathrm{d}w \tag{2}$$

where w is a standard Wiener process. As t approaches the final timestep T, the distribution of $\boldsymbol{x}_T$ converges to a known prior distribution, typically Gaussian. At inference time, the model reverses this corruption process by following the corresponding reverse-time SDE, which depends on the score function $\nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x})$. In practice, this score function is approximated by a denoiser network D_ϕ, enabling iterative denoising from $\boldsymbol{x}_T$ back to $\boldsymbol{x}_0$. An alternative, deterministic interpretation of the reverse process is given by the probability flow ODE (PFODE):

$$\mathrm{d}\boldsymbol{x} = \Big[f(\boldsymbol{x}, t) - \tfrac{1}{2}g(t)^2 \nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x})\Big]\mathrm{d}t \tag{3}$$

which preserves the same marginal distribution $p_t(\boldsymbol{x})$ as the reverse SDE at each timestep t. This ODE formulation often enables more efficient sampling through larger or adaptive step sizes without significantly compromising the sample quality.

EDM [Karras et al., 2022] refine both the forward and reverse processes through improved noise parameterization and training objectives. Concretely, they reparametrize the denoising score matching (DSM) loss so that the denoiser network learns to predict a scaled version of the clean data:

$$\mathcal{L}_{\text{EDM}} = \mathbb{E}_{t, \boldsymbol{x}_0, \boldsymbol{x}_t|\boldsymbol{x}_0} \left[d(\boldsymbol{x}_0, D_\phi(\boldsymbol{x}_t, t))\right] \tag{4}$$

where d is a distance metric in the clean data space. In this paper, we train an EDM model as the teacher using the pseudo huber loss as d following Prasad et al. [2024]. At inference time, EDM solves the associated PFODE with a 2nd-order Heun solver.

2.3 Consistency Trajectory Distillation

The iterative nature of the diffusion sampling process introduces significant computational overhead. Among various acceleration techniques proposed, consistency distillation [Song et al., 2023] has emerged as a particularly effective approach. The core idea is to train a student model that can emulate the many-step denoising process of a teacher diffusion model in a single step.

Building upon this framework, Kim et al. [2023] introduced Consistency Trajectory Models (CTM). Instead of learning only the end-to-end mapping from noise to clean samples, CTM learns to predict across arbitrary time intervals in the diffusion process. Specifically, given three arbitrary timesteps $0 \le k < u < t \le T$, CTM aims to align two different paths to predict $\boldsymbol{x}_k$: (1) direct prediction from time t to k using the student model, and (2) a two-stage prediction that first uses a numerical solver (e.g., Heun) with the teacher model to predict from time t to u, and then uses the student model to predict from time u to k.

Since the distance metric d is defined on the clean data space and may not be well-defined in the noisy data space, in practice we further map all predictions to time 0 using the student model. Formally, denote the student model as G_θ, the CTM loss is defined as:

$$\mathcal{L}_{\text{CTM}} = \mathbb{E}\left[d\left(G_{sg(\theta)}(\hat{\boldsymbol{x}}_k^{(t)}, k, 0), G_{sg(\theta)}(\boldsymbol{x}_k^{(t,u)}, k, 0)\right)\right] \tag{5}$$

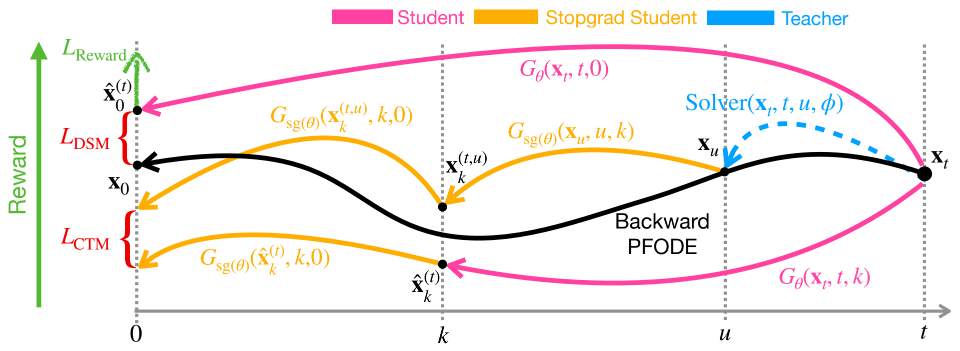
Student
Stopgrad Student
Teacher
L_{Reward}
Reward
$\hat{\mathbf{x}}_0^{(t)}$
L_{DSM}
$\mathbf{x}_0$
L_{CTM}
$G_\theta(\mathbf{x}_t, t, 0)$
Solver$(\mathbf{x}_t, t, u, \phi)$
$G_{\text{sg}(\theta)}(\mathbf{x}_k^{(t,u)}, k, 0)$
$G_{\text{sg}(\theta)}(\mathbf{x}_u, u, k)$
$\mathbf{x}_k^{(t,u)}$
$\mathbf{x}_u$
$\mathbf{x}_t$
Backward PFODE
$G_{\text{sg}(\theta)}(\hat{\mathbf{x}}_k^{(t)}, k, 0)$
$\hat{\mathbf{x}}_k^{(t)}$
$G_\theta(\mathbf{x}_t, t, k)$
0
k
u
t

Figure 2: Visualization of CTM loss, DSM loss and reward loss.

where $G_\theta(\boldsymbol{x}_t, t, u)$ represents the student prediction from time t to u given noisy sample $\boldsymbol{x}_t$ at time t, $\text{sg}(\theta)$ represents stop-gradient student parameters and

$$\hat{\boldsymbol{x}}_k^{(t)} = G_\theta(\boldsymbol{x}_t, t, k), \qquad \boldsymbol{x}_k^{(t,u)} = G_{\text{sg}(\theta)}(\text{Solver}(\boldsymbol{x}_t, t, u; \phi), u, k) \tag{6}$$

Here $\text{Solver}(\boldsymbol{x}_t, t, u; \phi)$ is the numerical solver result from time t to u using the teacher model D_ϕ given noisy sample $\boldsymbol{x}_t$ at time t.

In addition to the CTM loss, consistency trajectory distillation also incorporates the DSM loss to enforce its samples to be close to the training data. The DSM loss for the student model is the same as the one for EDM in Equation 4:

$$\mathcal{L}_{\text{DSM}} = \mathbb{E}_{t,\boldsymbol{x}_0,\boldsymbol{x}_t|\boldsymbol{x}_0}\left[d(\boldsymbol{x}_0, G_\theta(\boldsymbol{x}_t, t, 0))\right] \tag{7}$$

In Figure 2 we provide a visualization of these objectives on a PFODE trajectory. After the distillation, the student model can perform "anytime-to-anytime" jumps along the PFODE trajectory. One-step sampling can then be achieved by calculating $\hat{\boldsymbol{x}}_0^{(T)} = G_\theta(\boldsymbol{x}_T, T, 0)$.

3 Method

3.1 Motivation and Intuition

Diffusion models and their consistency-based counterparts have demonstrated promising results in decision-making tasks, particularly in capturing multimodal behavior patterns [Janner et al., 2022, Chi et al., 2023, Ajay et al., 2022]. Common recipes for using these models in decision-making generally fall into one of the three paradigms: (1) training a diffusion or consistency model via behavior cloning and deploying it directly as a policy [Chi et al., 2023, Ajay et al., 2022]; (2) integrating the model into actor-critic frameworks [Wang et al., 2022, Hansen-Estruch et al., 2023, Ding and Jin, 2023]; or (3) using the model as a planner through guided diffusion sampling [Janner et al., 2022].

While behavior cloning pipelines perform well when trained on expert demonstrations, they often struggles with suboptimal datasets (e.g., medium-replay buffers) collected from diverse behavior policies. Such datasets typically exhibit complex multimodal behavior patterns, where only some modes lead to high rewards. Although one could potentially use rejection sampling to filter out low-reward training data, this approach becomes prohibitively sample inefficient, particularly as the quality of the training data deteriorates. On the other hand, to generate high reward actions, actor-critic approaches require concurrent training of multiple neural networks with sensitive hyperparameters. Finally, guided diffusion sampling necessitates training noise-aware reward models and multi-step sampling, which could be detrimental for time-sensitive decision-making tasks like self-driving.

So how can we better leverage potentially suboptimal datasets to design a diffusion-based single-step sampling model with simple training procedure? Our key idea is to utilize the multimodal information captured by the teacher model and encourage the student model to favor the high reward modes. We achieve this by incorporating a reward objective directly in the consistency distillation process. Since our student model can achieve single-step denoising, we can incorporate a reward model trained in the clean sample space and avoid the multi-step reward optimization for diffusion models.

3.2 Modeling Action Sequences

Before introducing our reward-aware consistency trajectory distillation, we would like to first clearly establish the modeling formulation in our method. When applying diffusion models to RL, several modeling choices are available: modeling actions (as a policy), modeling rollouts (as a planner), or modeling state transitions (as a world model). While more comprehensive modeling approaches can offer advantages, particularly in long-horizon tasks, they also introduce additional complexity and computational challenges.

Following Chi et al. [2023], we adopt a balanced approach: modeling a fixed-length sequence of future actions conditioned on a fixed-length sequence of observed states. This formulation ensures consecutive actions form coherent sequences, and reduces the chances of generating idle actions.

Formally, let $\vec{\boldsymbol{s}}_n = \{\boldsymbol{s}_{n-h}, \boldsymbol{s}_{n-h+1}, \dots, \boldsymbol{s}_n\}$ denote a length-h sequence of past states at rollout time n, and $\vec{\boldsymbol{a}}_n = \{\boldsymbol{a}_n, \boldsymbol{a}_{n+1}, \dots, \boldsymbol{a}_{n+c}\}$ represent a length-c sequence of future actions. Both the teacher and the student learn to model the conditional distribution $p(\vec{\boldsymbol{a}}_n \mid \vec{\boldsymbol{s}}_n)$. In the context of diffusion notations, $\boldsymbol{x} = \vec{\boldsymbol{a}}_n \mid \vec{\boldsymbol{s}}_n$. We can easily extend this framework for goal-conditioned RL, where $\vec{\boldsymbol{s}}_n = \{\boldsymbol{s}_n, \boldsymbol{s}_T\}$ are the current and goal state that is used for conditioning.

During execution, we can either execute only the first predicted action $\boldsymbol{a}_n$ in the environment before replanning at the next step, or follow the entire predicted sequence of actions at once.

3.3 Reward-Aware Consistency Trajectory Distillation

Having established the formulation, we now present our approach to integrating reward optimization into the consistency trajectory distillation process.

Let R_ψ be a pre-trained differentiable return-to-go network (i.e. reward model) that takes the state $\boldsymbol{s}_n$ and action $\boldsymbol{a}_n$ at rollout time n as inputs and predicts the future discounted cumulative reward $\hat{r}_n = \sum_{j=0}^{H-n} \gamma^j r_{n+j}$. When the student model generates a prediction $\vec{\boldsymbol{a}}_n \mid \vec{\boldsymbol{s}}_n = \hat{\boldsymbol{x}}_0^{(T)} = G_\theta(\boldsymbol{x}_T, T, 0)$, we extract the action at time n, denoted as $\hat{\boldsymbol{a}}_n$, from the predicted sequence and pass it along with $\boldsymbol{s}_n$ to the frozen reward model R_ψ to estimate $\hat{r}_n$. The goal of our reward-aware training is to maximize the estimated discounted cumulative reward. Mathematically, the reward objective is defined as

$$\mathcal{L}_{\text{Reward}} = -R_\psi(\vec{\boldsymbol{s}}_n, \hat{\boldsymbol{a}}_n) \tag{8}$$

The final loss for reward-aware consistency trajectory distillation (RACTD) combines all three terms:

$$\mathcal{L} = \alpha \mathcal{L}_{\text{CTM}} + \beta \mathcal{L}_{\text{DSM}} + \sigma \mathcal{L}_{\text{Reward}} \tag{9}$$

where α, β, and σ are hyperparameters to balance different loss terms.

3.4 Decoupled Training

A key advantage of our method of combining reinforcement learning, diffusion models, and consistency distillation is the ability to support fully decoupled training of all components. Traditional actor-critic frameworks, which are commonly used to incorporate diffusion models into reinforcement learning, require simultaneous training of multiple neural networks. This concurrent optimization presents considerable challenges, often demanding extensive hyperparameter tuning and careful balancing of different learning objectives.

Guided diffusion sampling, as proposed in Janner et al. [2022], offers an alternative approach by taking inspiration from classifier guided diffusion [Dhariwal and Nichol, 2021, Song et al., 2020b]. However, these classifiers (i.e. reward models) require noise-aware training that cannot be separated from the diffusion model. Also, predicting the correct reward from highly corrupted input could be very challenging, which can lead to inaccurate guidance that accumulates during its multi-step sampling process.

Our method, in contrast, fully leverages the advantages of single-step denoising models by operating entirely in the noise-free state-action space. This design choice enables the reward model to provide stable and effective signals without requiring noise-aware training. Importantly, the reward model can be pre-trained completely decoupled from the teacher model and distillation process. This separation not only simplifies the training process but also allows for flexible integration of different reward models using the same teacher model.

3.5 Reward Objective as Mode Selection

In offline RL, models often have to learn from datasets containing behaviors of varying quality. While diffusion models excel at capturing these diverse behavioral modes, they inherently lack the ability to differentiate between actions that lead to high versus low rewards. Our RACTD addresses this limitation by transforming the reward-agnostic teacher diffusion sampling distribution into one that preferentially samples from high-reward modes, We empirically verify this through a comparative analysis using the D4RL hopper-medium-expert dataset Fu et al. [2020], which contains an equal mixture of expert demonstrations and suboptimal rollouts from a partially trained policy.

Figure 3 illustrates the reward distributions of rollouts sampled from three models: the unconditioned teacher, unconditioned student, and RACTD. The dataset (grey) exhibits two distinct modes corresponding to medium-quality and expert rollouts. The unconditioned teacher model (blue) accurately captures this bimodal distribution, and the unconditioned student model (orange) faithfully replicates it. In contrast, our RACTD (green) concentrates its samples on the higher-reward mode, demonstrating that our reward guidance effectively identifies and selects optimal behaviors from the teacher's multi-modal distribution. We also include the discussion between sample diversity and mode selection in Appendix G.

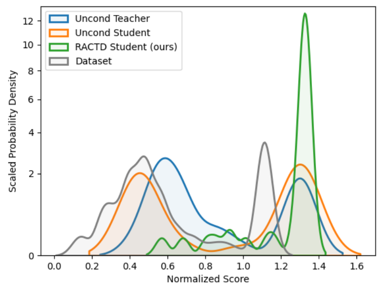

Figure 3: The reward distribution of the D4RL hopper-medium-expert dataset and 100 rollouts from an unconditioned teacher, an unconditioned student, and RACTD.

4 Experiment

In this section, we conduct experiments to demonstrate: (1) the effectness of our RACTD, especially when the teacher model is trained with suboptimal data, (2) the capability of accurately capturing the complex and diverse high dimensional behavior patterns with only one denoising step, and (3) the speed-up achieved over the teacher model and existing policy-based diffusion models.

4.1 Offline RL

Baselines We compare our approach against a comprehensive set of baselines, including vanilla behavior cloning (BC) and Consistency Policy (Consistency BC) [Ding and Jin, 2023]; model-free RL algorithms CQL [Kumar et al., 2020] and IQL [Kostrikov et al., 2021]; model-based algorithms Trajectory Transformer (TT) [Janner et al., 2021], MOPO [Yu et al., 2020], MOReL [Kidambi et al., 2020], MBOP [Argenson and Dulac-Arnold, 2020]; autoregressive model Decision Transformer (DT) Chen et al. [2021]; diffusion-based planner Diffuser Janner et al. [2022]; and diffusion-based actor-critic methods Diffusion QL [Wang et al., 2022] and Consistency AC [Ding and Jin, 2023].

Setup We first evaluate our method and the baselines on D4RL Gym-MuJoCo [Fu et al., 2020], which is a popular offline RL benchmark that contains mixtures of varying quality data.

We include the results for both online and offline models selection following prior works, where we report the performance of the last epoch for offline model selection and the best-performing checkpoint observed during training for online model selection. Results for non-diffusion-based models and Diffuser are taken from Janner et al. [2022], and results for Diffusion QL and Consistency AC/BC are sourced from their respective papers. The results for RACTD are reported as the mean and standard error over 100 planning seeds. We use $h = 1, c = 16$ and closed-loop plannign in all experiments.

Results As shown in Table 1 and Table 7, RACTD achieves the best or second best performance in almost all task in Gym-MujoCo, and outperforms the best baseline in overall average score by a substantial margin. It consistently outperforms the diffusion-based planner Diffuser, and exceeds the performance of consistency-based actor-critic baseline Consistency AC in all hopper and walker2d tasks.

4.2 Long Horizon Planning

Next, we showcase the effectiveness of RACTD in complex high-dimensional long-horizon planning and its flexibility to adapt to goal-conditioned tasks. Previously, Diffuser has shown great potential

Table 1: (Offline RL: **Offline model selection**) Performance of RACTD and a variety of baselines on the D4RL Gym-MujoCo benchmark. The best score is emphasized in bold and the second-best is underlined.

Method	Medium Expert			Medium			Medium Replay			Avg	NFE
	HalfCheetah	Hopper	Walker2d	HalfCheetah	Hopper	Walker2d	HalfCheetah	Hopper	Walker2d		
BC	55.2	52.5	107.5	42.6	52.9	75.3	36.6	18.1	26.0	51.9	-
CQL	91.6	105.4	108.8	44.0	58.5	72.5	45.5	95.0	77.2	77.6	-
IQL	86.7	91.5	109.6	47.4	66.3	78.3	44.2	94.7	73.9	77.0	-
DT	86.8	107.6	108.1	42.6	67.6	74.0	36.6	82.7	66.6	74.7	-
TT	95.0	110.0	101.9	46.9	61.1	79.0	41.9	91.5	82.6	78.9	-
MOPO	63.3	23.7	44.6	42.3	28.0	17.8	$\underline{53.1}$	67.5	39.0	42.1	-
MOReL	53.3	108.7	95.6	42.1	**95.4**	77.8	40.2	93.6	49.8	72.9	-
MBOP	**105.9**	55.1	70.2	44.6	48.8	41.0	42.3	12.4	9.7	47.8	-
Diffusion QL	$\underline{96.8}$ ±0.3	$\underline{111.1}$ ±1.3	110.1 ±0.3	51.1 ±0.5	$\underline{90.5}$ ±4.6	$\underline{87.0}$ ±0.9	47.8 ±0.3	**101.3** ±0.6	$\underline{95.5}$ ±1.5	87.9	5
Consistency AC	84.3 ±4.1	100.4 ±3.5	$\underline{110.4}$ ±0.7	**69.1** ±0.7	80.7 ±10.5	83.1 ±0.3	**58.7** ±3.9	99.7±0.5	79.5 ±3.6	$\underline{85.1}$	2
Consistency BC	32.7 ±1.2	90.6 ±9.3	$\underline{110.4}$ ±0.7	31.0 ±0.4	71.7 ±8.0	83.1 ±0.3	34.4 ±5.3	99.7 ±0.5	73.3 ±5.7	69.7	2
Diffuser	88.9 ±0.3	103.3 ±1.3	106.9 ±0.2	42.8 ±0.3	74.3 ±1.4	79.6 ±0.55	37.7 ±0.5	93.6 ±0.4	70.6 ±1.6	77.5	20
RACTD(Ours)	88.5 ±2.1	**120.2** ±2.6	**122.3** ±0.3	$\underline{56.6}$ ±0.6	87.2 ±1.4	**112.8** ±1.3	$\underline{51.4}$ ±0.2	$\underline{101.1}$ ±2.2	**105.2** ±1.8	**93.9**	**1**

Table 2: (Long-horizon planning) The performance of RACTD, Diffuser, and prior model-free algorithms in the Maze2D environment. The best score is in bold and the second-best is underlined.

Method	U-Maze			Medium			Large			Avg Score
	Score	NFE	Time (s)	Score	NFE	Time (s)	Score	NFE	Time (s)	
MPPI	33.2	-	-	10.2	-	-	5.1	-	-	16.2
CQL	5.7	-	-	5	-	-	12.5	-	-	7.7
IQL	47.4	-	-	34.9	-	-	58.6	-	-	47.0
Diffuser	113.9 ±3.1	64	1.664	$\underline{121.5}$ ±2.7	256	4.312	123.0 ±6.4	256	5.568	119.5
CTD	$\underline{123.4}$ ±1.0	**1**	**0.029**	119.8 ±4.1	**1**	**0.047**	$\underline{127.1}$ ±6.4	**1**	**0.049**	$\underline{123.4}$
RACTD (Ours)	**125.7** ±0.6	**1**	**0.029**	**130.8** ±1.8	**1**	**0.047**	**143.8** ±0.0	**1**	**0.049**	**133.4**

in open-loop long-horizon planning, but requires a significantly larger number of denoising steps compared to closed-loop planning like MuJoCo. We demonstrate that our model can achieve superior performance with a single-step denoising process under the same problem formulation.

Setup We test this ability on D4RL Maze2d [Fu et al., 2020], which is a sparse reward long-horizon planning task where an agent may take hundreds of steps to reach the goal in static environments. Following the setup in Janner et al. [2022], we use a planning horizon $128, 256, 384$ for U-Maze, Medium and Large respectively. We perform open-loop planning by generating the entire state sequence followed by a reverse dynamics model to infer all the actions from the predicted state sequence. The reward model returns 1 if the current state reaches the goal and 0 otherwise. The baseline results are reported from Janner et al. [2022] and RACTD results are reported as the mean and standard error of 100 planning seeds.

Results As shown in Table 13, both Diffuser and RACTD outperform previous model-free RL algorithms CQL and IQL, and MPPI which usese ground-truth dynamics. Our approach surpasses the diffusion baseline in almost all settings, highlighting its ability to effectively capture complex behavioral patterns and high-dimensional information in the training dataset. Notably, the planning dimension for this task (384 for the Large Maze) is substantially higher than that of MuJoCo tasks (16). As a result, Diffuser requires significantly more denoising steps (256 for the Large Maze) compared to MuJoCo (20 steps). On the contrary, despite the increased task complexity, RACTD still only requires a single denoising step to achieve $11.6\times$ performance boost.

4.3 Inference Time Comparison

Beyond performance improvements, a key contribution of our work is significantly accelerating diffusion-based models for decision-making tasks. The primary computational bottleneck in diffusion models arises from the multiple function evaluations (NFEs) required by the iterative denoising process. By reducing the number of denoising steps to a single NFE, our approach achieves a speed-up roughly proportional to the number of denoising originally required.

Setup To evaluate sampling efficiency, we compare RACTD with different samplers, including DDPM [Ho et al., 2020], DDIM [Song et al., 2020a], and EDM [Karras et al., 2022] using the same network architecture as our teacher model. Additionally, we report the efficiency of Diffuser. Note that since Diffuser employs a different model architecture and generates future state-action pairs, its sampling time may also be influenced by these factors. Table 9 and Table 13 present the wall clock sampling time and NFE for MuJoCo (hopper-medium-replay) and Maze2d. All experiments are conducted on one NVIDIA Tesla V100-SXM2-32GB.

Results In hopper-medium-replay, RACTD achieved $20\times$ reduction in NFEs and a $43\times$ speed-up compared to Diffuser. Additionally, our student model requires $80\times$ fewer NFEs and samples $142\times$ faster than the teacher model. In Maze2d, RACTD significantly accelerates computation compared to Diffuser, achieving approximately $57\times$, $92\times$, and $114\times$ speed-ups on Umaze, Medium and Large mazes, by reducing NFEs by a factor of 256 for the Medium and Large mazes. Furthermore, we observe that more denoising steps lead to better model performance. This highlights the advantage of distilling from a slow but high-performing teacher model, which enables better performance compared to training a diffusion model directly with fewer denoising steps.

5 Related Work

Diffusion Models in Reinforcement Learning Diffusion models have emerged as a powerful approach for decision-making tasks in reinforcement learning Janner et al. [2022], Ajay et al. [2022], Wang et al. [2022], Hansen-Estruch et al. [2023], Chi et al. [2023]. The integration of diffusion models into RL frameworks typically follows two main paradigms: actor-critic approaches, where diffusion models serve as policy or value networks [Wang et al., 2022, Hansen-Estruch et al., 2023], and policy-based approaches, where diffusion models directly generate action trajectories Janner et al. [2022], Ajay et al. [2022], Chi et al. [2023]. While these methods have demonstrated impressive performance on standard RL benchmarks, their practical deployment is hindered by the slow sampling time inherent to vanilla diffusion policies based on DDPM Ho et al. [2020].

Accelerating Diffusion Model Sampling Various approaches have been proposed to accelerate the sampling process in diffusion models. One prominent direction leverages advanced ODE solvers to reduce the number of required denoising steps [Song et al., 2020a, Karras et al., 2022, Lu et al., 2022]. Another line of work explores knowledge distillation techniques Luhman and Luhman [2021], Salimans and Ho [2022], Berthelot et al. [2023], Kim et al. [2023], Song et al. [2023], where student models learn to take larger steps along the ODE trajectory. Consistency trajectory models [Kim et al., 2023] enable one-step sampling by learning anytime-to-anytime jumps along the PFODE trajectory.

Consistency Models in Decision Making Consistency models have emerged as a promising policy class for behavior cloning from expert demonstrations in robotics [Lu et al., 2024, Prasad et al., 2024, Wang et al., 2024]. In RL, several works have enhanced actor-critic methods by replacing traditional diffusion-based value/policy networks with consistency models, showing faster inference and training speed [Chen et al., 2023, Ding and Jin, 2023, Li et al., 2024]. These approaches directly incorporate consistency loss [Song et al., 2023] into the value/policy network training, rather than distilling a separate student model. While Wang et al. made some initial attempts to apply consistency distillation in policy-based RL through classifier-free guidance and reverse dynamics, their approach requires two NFEs and under-performs the state-of-the-art. In contrast, RACTD is a straightforward approach of using a separate reward model and incorporating reward objective during student distillation, achieving superior performance with only one NFE.

6 Conclusion

In this work, we address the challenge of diffusion policy acceleration by introducing reward-aware consistency trajectory distillation (RACTD), which predicts high-reward actions in a single denoising step. RACTD uses a pre-trained diffusion teacher model and a separately trained reward model, leveraging the teacher's ability to capture multi-modal distributions while prioritizing higher-reward modes to generate high-quality samples from suboptimal training data. Its decoupled training approach avoids the complex concurrent optimization of multiple networks and enables the use of a standalone, noise-free reward model. RACTD outperforms previous state-of-the-art by 6.8% while accelerating its diffusion counterparts up to a factor of 142.

References

Anurag Ajay, Yilun Du, Abhi Gupta, Joshua Tenenbaum, Tommi Jaakkola, and Pulkit Agrawal. Is conditional generative modeling all you need for decision-making? *arXiv preprint arXiv:2211.15657*, 2022.

Arthur Argenson and Gabriel Dulac-Arnold. Model-based offline planning. *arXiv preprint arXiv:2008.05556*, 2020.

David Berthelot, Arnaud Autef, Jierui Lin, Dian Ang Yap, Shuangfei Zhai, Siyuan Hu, Daniel Zheng, Walter Talbott, and Eric Gu. Tract: Denoising diffusion models with transitive closure time-distillation. *arXiv preprint arXiv:2303.04248*, 2023.

Adam Block, Ali Jadbabaie, Daniel Pfrommer, Max Simchowitz, and Russ Tedrake. Provable guarantees for generative behavior cloning: Bridging low-level stability and high-level behavior. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL `https://openreview.net/forum?id=PhFVF0gwid`.

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. *Advances in neural information processing systems*, 34:15084–15097, 2021.

Yuhui Chen, Haoran Li, and Dongbin Zhao. Boosting continuous control with consistency policy. *arXiv preprint arXiv:2310.06343*, 2023.

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, page 02783649241273668, 2023.

Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021.

Zihan Ding and Chi Jin. Consistency models as a rich and efficient policy class for reinforcement learning. *arXiv preprint arXiv:2309.16984*, 2023.

Xintong Duan, Yutong He, Fahim Tajwar, Wen-Tse Chen, Ruslan Salakhutdinov, and Jeff Schneider. State combinatorial generalization in decision making with conditional diffusion models. *arXiv preprint arXiv:2501.13241*, 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.

Philippe Hansen-Estruch, Ilya Kostrikov, Michael Janner, Jakub Grudzien Kuba, and Sergey Levine. Idql: Implicit q-learning as an actor-critic method with diffusion policies. *arXiv preprint arXiv:2304.10573*, 2023.

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

Audrey Huang, Adam Block, Dylan J Foster, Dhruv Rohatgi, Cyril Zhang, Max Simchowitz, Jordan T Ash, and Akshay Krishnamurthy. Self-improvement in language models: The sharpening mechanism. *arXiv preprint arXiv:2412.01951*, 2024.

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

Michael Janner, Yilun Du, Joshua B Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. *arXiv preprint arXiv:2205.09991*, 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. *arXiv preprint arXiv:2310.02279*, 2023.

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

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.

Haoran Li, Zhennan Jiang, Yuhui Chen, and Dongbin Zhao. Generalizing consistency policy to visual rl with prioritized proximal experience regularization. *arXiv preprint arXiv:2410.00051*, 2024.

Haohe Liu, Zehua Chen, Yi Yuan, Xinhao Mei, Xubo Liu, Danilo Mandic, Wenwu Wang, and Mark D Plumbley. Audioldm: Text-to-audio generation with latent diffusion models. *arXiv preprint arXiv:2301.12503*, 2023.

Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps. *Advances in Neural Information Processing Systems*, 35:5775–5787, 2022.

Guanxing Lu, Zifeng Gao, Tianxing Chen, Wenxun Dai, Ziwei Wang, and Yansong Tang. Manicm: Real-time 3d diffusion policy via consistency model for robotic manipulation. *arXiv preprint arXiv:2406.01586*, 2024.

Eric Luhman and Troy Luhman. Knowledge distillation in iterative generative models for improved sampling speed. *arXiv preprint arXiv:2101.02388*, 2021.

Diganta Misra. Mish: A self regularized non-monotonic activation function. *arXiv preprint arXiv:1908.08681*, 2019.

Aaditya Prasad, Kevin Lin, Jimmy Wu, Linqi Zhou, and Jeannette Bohg. Consistency policy: Accelerated visuomotor policies via consistency distillation. *arXiv preprint arXiv:2405.07503*, 2024.

Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=TIdIXIpzhoI`.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020a.

Yang Song and Prafulla Dhariwal. Improved techniques for training consistency models. In *The Twelfth International Conference on Learning Representations*, 2024. URL `https://openreview.net/forum?id=WNzy9bRDvG`.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020b.

Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. *arXiv preprint arXiv:2303.01469*, 2023.

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 2024*.

Zhendong Wang, Jonathan J Hunt, and Mingyuan Zhou. Diffusion policies as an expressive policy class for offline reinforcement learning. *arXiv preprint arXiv:2208.06193*, 2022.

Zhendong Wang, Zhaoshuo Li, Ajay Mandlekar, Zhenjia Xu, Jiaojiao Fan, Yashraj Narang, Linxi Fan, Yuke Zhu, Yogesh Balaji, Mingyuan Zhou, et al. One-step diffusion policy: Fast visuomotor policies via diffusion distillation. *arXiv preprint arXiv:2410.21257*, 2024.

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 Model Architecture

We follow the model architecture used in Chi et al. [2023] and Prasad et al. [2024] and continue to use the 1D temporal CNN layer for our Unet and FiLM layers to process the conditioning information.

A.1 Model Sizes for Maze2d

Model parameters for teacher models of Umaze, Medium, and Large Maze are shown below in Table 3. The student model has the same architecture as the teacher model except it also takes one extra variable of denoising timestep as conditioning.

Parameter	Umaze	Medium	Large
diffusion_step_embed_dim	256	256	256
down dims	[256, 512, 1024]	[512, 1024, 2048]	[256, 512, 1024, 2048]
horizon	128	256	384
kernel size	5	5	5

Table 3: Model parameters for Unet in Maze2d.

A.2 Model Sizes for Gym-MuJoCo

Unet parameters for teacher and student models in the MuJoCo task are shown below in Table 4. Model sizes are fixed through 9 different environments.

Parameter	MuJoCo
diffusion_step_embed_dim	128
down dims	[512, 1024, 2048]
horizon	16
kernel size	5

Table 4: Model parameters for Unet in MuJoCo.

We follow the setup in Janner et al. [2022], where we use Linear layers and Mish layers Misra [2019] for the reward model. Reward model architecture remains the same across all MuJoCo benchmarks. Model parameters are shown below in Table 5.

B Training Details

Our models are trained on D4RL dataset [Fu et al., 2020], which was released under Apache-2.0 license.

B.1 Noise Scheduler

We follow the setup in Prasad et al. [2024] and use EDM noise scheduler [Karras et al., 2022] for the teacher model. Particularly, discretization bins are chosen to be 80.

Student model used CTM scheduler [Kim et al., 2023] also with discretization bins of 80.

B.2 Weight of Different Losses

The weights for CTM, DSM, and Reward loss we used in the experiment are shown below in Table 6. Generally, if the training dataset includes more expert samples, the weight for reward guidance is smaller. A reward weight of 0.0 resembles behavior cloning with consistency trajectory distillation. We found that as long as the loss weights are chosen to keep the individual loss terms within the same order of magnitude, the model will achieve reasonable performance.

Parameter	MuJoCo
layer dimensions	[32, 64, 128, 256]

Table 5: Reward model parameters in MuJoCo.

Parameter	CTM	DSM	Reward
hopper-medium-replay	1.0	1.0	1.0
hopper-medium	1.0	1.0	3.0
hopper-medium-expert	1.0	1.0	0.0
walker2d-medium-replay	1.0	1.0	1.0
walker2d-medium	1.0	1.0	0.4
walker2d-medium-expert	1.0	1.0	0.2
halfcheetah-medium-replay	1.0	1.0	1.0
halfcheetah-medium	1.0	1.0	0.5
halfcheetah-medium-expert	1.0	1.0	0.0

Table 6: Weights for CTM, DSM, and Reward loss used in MuJoCo benchmark.

C Results for Online model selection in Gym-MuJoCo

We include the performance of online model selection for diffusion-based methods below in Table 7.

Table 7: (Offline RL: **Online model selection**) Performance of RACTD and diffusion based baselines on the D4RL Gym-MujoCo benchmark. The best score is emphasized in bold and the second-best is underlined.

Method	Medium Expert			Medium			Medium Replay			Avg	NFE
	HalfCheetah	Hopper	Walker2d	HalfCheetah	Hopper	Walker2d	HalfCheetah	Hopper	Walker2d		
Diffusion QL	97.2 ±0.4	112.3 ±0.8	111.2 ±0.9	51.5 ±0.3	96.6 ±3.4	87.3 ±0.5	48.3 ±0.2	102.0 ±0.4	98.0 ±0.5	89.3	5
Consistency AC	89.2 ±3.3	106.0 ±1.3	111.6 ±0.7	**71.9** ±0.8	99.7 ±2.3	84.1 ±0.3	**62.7** ±0.6	100.4±0.6	83.0 ±1.5	89.8	2
Consistency BC	39.6 ±3.4	96.8 ±4.6	111.6 ±0.7	46.2 ±0.4	78.3 ±2.6	84.1 ±0.3	45.4 ±0.7	100.4 ±0.6	80.8 ±3.4	75.9	2
RACTD(Ours)	95.9 ±1.5	**129.0** ±1.3	**122.3** ±0.3	59.3 ±0.2	**121.0** ±0.5	**118.8** ±0.3	57.9 ±1.0	**104.9** ±2.1	**105.2** ±1.8	**101.6**	**1**

D Results for Kitchen

We also include the performance for Kitchen in Table 8.

Table 8: (Offline RL) Performance of RACTD and diffusion-based baselines on the D4RL Kitchen benchmark. The best score is emphasized in bold and the second-best is underlined. Each cell has two values: one for offline model selection and another (in brackets) for online model selection.

Method	Kitchen			Avg	NFE
	complete	partial	mixed		
Diffusion QL	**84.0** ±7.4 (84.5 ±6.1)	**60.5** ±6.9 (63.7 ±5.2)	**62.6** ±5.1 (66.6 ±3.3)	**69.0**(71.6)	5
Consistency AC	51.9 ±6.0 (67.6 ±2.7)	38.2 ±1.8 (39.8 ±1.6)	45.8 ±1.5 (46.7 ±0.9)	45.3(51.4)	2
Consistency BC	45.2 ±5.0 (50.9 ±3.6)	22.6 ±3.8 (23.8 ±2.8)	23.5 ±1.8 (24.3 ±1.3)	30.4(33.0)	2
RACTD(Ours)	56.3 ±8.2 (58.1 ±8.3)	59.0 ±14.9 (63.1 ±0.5)	60.9 ±6.1 (61.9 ±1.6)	58.7(61.0)	**1**

E Ablation Study

We ablate over (1) the impact of the reward objective in both student and teacher model training, (2) the effect of different reward loss weights on model performance and training stability, and (3) the impact of increasing denoising steps with our student model.

Table 9: Wall clock time and NFEs per action for different samplers and Diffuser on MuJoCo hopper-medium-replay.

Method	**Time (s)**	**NFE**	**Score**
Diffuser	0.644	20	93.6
DDPM	0.236	15	24.2
DDIM	0.208	15	60.6
EDM (Teacher)	2.134	80	**114.2**
RACTD (Ours, Student)	**0.015**	**1**	104.9

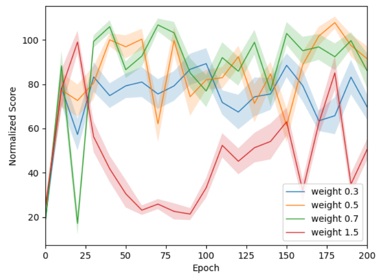

Figure 4: Ablation on reward objective weight.

Table 10: We compare incorporating the reward model in different stages of training on MuJoCo hopper-medium-replay. Results are presented as the mean and standard error across 100 seeds.

Hopper Medium-Replay	Unconditioned teacher	Reward-Aware teacher
Unconditioned student	50.8 ± 0.3	96.2 ± 0.2
Reward-Aware student	**104.9** ± 2.1	96.0 ± 0.3

Table 11: Inference time, NFE, and score comparison for student model multi-step sampling on MuJoCo hopper-medium-replay.

NFE	**Time(s)**	**Score**
1	**0.0147**	104.9 ± 2.1
2	0.0241	**109.8** ± 0.9
3	0.0377	108.7 ± 0.1
4	0.0517	107.9 ± 1.6

E.1 Impact of Reward Objective

To understand the unique advantages of incorporating reward objectives during distillation, we conduct a systematic comparison across four model configurations: the baseline combination of an unconditioned teacher and student, a reward-aware teacher paired with an unconditioned student, a fully reward-aware teacher-student pair, and our proposed RACTD, which combines an unconditioned teacher with a reward-aware student. The results for hopper-medium-replay and walker-medium is shown in Table 10 and Table 12.

Our analysis reveals that while incorporating reward objectives at any stage yields substantial improvements, optimal performance is achieved through our RACTD framework, which combines an unconditioned teacher with reward-aware student distillation. Our method allows the teacher to capture a comprehensive range of behavioral patterns, while enabling the student to selectively distill the most effective strategies. Although incorporating reward objectives in the teacher model also enhances performance, this approach risks discarding suboptimal behaviors that may be valuable in novel testing scenarios, potentially limiting the model's generalization capabilities.

E.2 Effect of Reward Objective Weights

The reward loss weight is a crucial hyperparameter in our pipeline that impacts both training stability and performance. We plot the reward curve achieved over 200 epochs of student training with reward loss weights ranging from $[0.3, 0.5, 0.7, 1.5]$ on MuJoCo hopper-medium-replay in Figure 4. The mean and standard error are reported across 20 rollouts from intermediate checkpoints.

With lower weights, increasing the weight leads to higher performance and relatively stable training. However, when the weight is too high (e.g. 1.5 in this plot), evaluation initially increases but fluctuates as training progresses. This occurs when the reward loss dominates DSM and CTM losses, resulting in unstable training.

E.3 Number of Sampling Steps

Since our student model is trained for anytime-to-anytime jumps, it naturally extends to multi-step denoising without additional training. Following the approach in Song et al. [2023], given intermediate denoising timesteps $0 < t_1 < t_2 < T$, we first denoise from T to 0 as usual. We then add noise again to t_1 and denoise it back to 0, and repeat this process for t_2. This iterative refinement can enhance generation quality. We evaluate the student using 2, 3, and 4 denoising steps as reported in Table 11. While multi-step sampling improves performance, we observe that gains do not scale linearly with the number of denoising steps.

F More Ablations

F.1 walker-medium

Table 12: We compare incorporating the reward model in different stages of training on MuJoCo walker-medium. Results are presented as the mean and standard error across 100 seeds.

Walker Medium	Unconditioned teacher	Reward-Aware teacher
Unconditioned student	93.3 ±1.8	97.0 ±1.0
Reward-Aware student	**98.9** ±1.6	94.5 ±2.6

F.2 Comparing with fast sampling algorithms for Maze2d

Table 13: We compare fast sampling algorithms DDIM and CTD, along with our method RACTD on Maze2d environment. DDIM performs fast sampling based on a DDPM model, while CTD and RACTD (ours) distill an EDM teacher. The number of function evaluations (NFE) reflects the sampling speed of each algorithm. Results are reported as the mean and standard error over 100 random seeds.

Method	**NFE**	**U-Maze**	**Medium**	**Large**	**Average Score**
		Score	**Score**	**Score**	
DDPM	100	**126.3**±0.7	<u>126.8</u> ±3.0	<u>144.8</u> ±4.9	<u>132.6</u>
DDIM	<u>10</u>	121.2 ±1.1	126.2 ±2.8	143.1 ±4.9	130.2
DDIM	**1**	3.5 ±4.7	−2.6 ±12.5	−1.5 ±0.7	−0.2
EDM	80	125.4 ±0.6	120.1 ±4.2	**149.0** ±0.5	131.5
CTD	**1**	123.4 ±1.0	119.8 ±4.1	127.1 ±6.4	123.4
RACTD (ours)	**1**	<u>125.7</u> ±0.6	**130.8** ±1.8	143.8 ±0.0	**133.4**

G The Trade-off between Mode Selection and Sample Diversity

In this section, we include a discussion about the trade-off between mode selection induced by our reward-aware training and the sample diversity of the student. Naturally, favoring selected modes can led to generation with limited sample diversity as summarized in Table 14. This trade-off between sample diversity and sample optimality observed in RACTD is similar to what has been seen in other generative domains (e.g., language model RLHF [Huang et al., 2024], classifier-guided diffusion, conditional image generation), where preference alignment also often reduces sample diversity. In our case, the reward model acts similarly to a classifier or an alignment reward model, guiding the model toward desirable behaviors and sacrificing some of the sample diversity by design.

Importantly, our decoupled framework allows the use of a single, unconditioned teacher with strong generalization capabilities across tasks. For multi-task or unseen-task settings, different reward models can be trained per task, and corresponding student models can be distilled from the same teacher using different reward models.

Table 14: A summarization of the trade off between sample diversity and model performance.

	Sample diversity	Performance	Sample time
Reward agnostic diffusion	High	Low	Slow
Reward aware diffusion	Low	High	Slow
Reward agnostic consistency distillation	High	Low	Fast
Reward aware consistency distillation	Low	High	Fast

H Limitations and Future work

One limitation of our approach is the need to train three separate networks: the teacher, student, and reward model. Training the teacher can be time-consuming, as achieving strong performance often requires a higher number of denoising steps. Additionally, consistency trajectory distillation is prone to loss fluctuations, and incorporating a reward model into the distillation process may further amplify this instability. Future work will focus on developing a more stable and efficient training procedure, as well as exploring methods to integrate non-differentiable reward models into the framework.