DiffuserLite: Towards Real-time Diffusion Planning

Zibin Dong[1*] **Jianye Hao**[1†] **Yifu Yuan**[1] **Fei Ni**[1] **Yitian Wang**[2] **Pengyi Li**[1] **Yan Zheng**[1]
[1]College of Intelligence and Computing, Tianjin University
[2]UC San Diego Jacobs School of Engineering

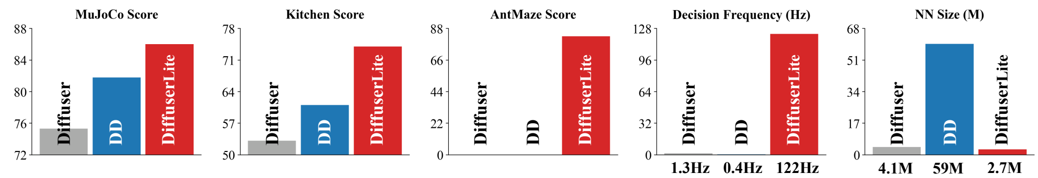

Figure 1: **Performance overview.** We present DiffuserLite, a lightweight framework that utilizes progressive refinement planning to reduce redundant information generation and achieves real-time diffusion planning. DiffuserLite significantly outperforms predominant frameworks, Diffuser and DD, regarding scores, inference time, and model size on three popular D4RL benchmarks. The decision-making frequency of DiffuserLite achieves **122.2Hz**, which is **112.7 times higher** than predominant frameworks.

Abstract

Diffusion planning has been recognized as an effective decision-making paradigm in various domains. The capability of generating high-quality long-horizon trajectories makes it a promising research direction. However, existing diffusion planning methods suffer from low decision-making frequencies due to the expensive iterative sampling cost. To alleviate this, we introduce **DiffuserLite**, a super fast and lightweight diffusion planning framework, which employs a planning refinement process (PRP) to generate coarse-to-fine-grained trajectories, significantly reducing the modeling of redundant information and leading to notable increases in decision-making frequency. Our experimental results demonstrate that DiffuserLite achieves a decision-making frequency of 122.2Hz (112.7x faster than predominant frameworks) and reaches state-of-the-art performance on D4RL, Robomimic, and FinRL benchmarks. In addition, DiffuserLite can also serve as a flexible plugin to increase the decision-making frequency of other diffusion planning algorithms, providing a structural design reference for future works. More details and visualizations are available at project website.

1 Introduction

Diffusion models (DMs) are powerful generative models that demonstrate promising performance across various domains [52, 44, 29, 22]. Motivated by their remarkable capability in complex distribution modeling and conditional generation, researchers have developed a series of works applying diffusion models for decision-making tasks in recent years [55]. DMs can play various roles in decision-making tasks, such as acting as planners to make better decisions from a long-term perspective [20, 1, 8, 36, 10], serving as policies to support complex multimodal-distribution modeling [38, 50, 6], and working as data synthesizers to assist reinforcement learning (RL) training [33, 51, 16], etc. Among these roles, diffusion planning is the most widely applied paradigm [55]. Unlike auto-regressive planning in previous model-based RL approaches [14, 49, 15], diffusion planning avoids severe compounding errors by directly generating the entire trajectory rather than one-step transition [55]. Also, its powerful conditional generation capability allows planning at the

*Contact me at zibindong@outlook.com
†Correspondence to: Jianye Hao (jianye.hao@tju.edu.cn), Yan Zheng (yanzheng@tju.edu.cn)

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

 https://doi.org/10.52202/079017-3895

trajectory level without being limited to step-wise shortsightedness. The diffusion planning paradigm has achieved state-of-the-art (SOTA) performance in various offline RL tasks, including single-agent RL [27], multi-agent RL [54], meta RL [36], and more.

One key issue that diffusion planning faces is the expensive iterative sampling cost. As depicted in fig. 1, the decision-making frequencies (number of actions inferred per second) of two predominant diffusion planning frameworks, Diffuser [20] and Decision Diffuser (DD) [1], are recorded as 1.3Hz and 0.4Hz, respectively. Such a low decision frequency fails to meet the requirements of numerous real-world applications, e.g. real-time robot control [48] and game AI [39]. The low decision frequency is primarily attributed to modeling a denoising process for a long-horizon trajectory distribution, which requires a heavy neural network backbone and multiple forward passes. A question may arise whether the generation of complete long-horizon trajectories is necessary for successful planning. Experimental results indicate that it is not, for the detailed trajectory information in long-horizon segments is highly redundant. As shown in fig. 2, a motivation example in Antmaze, the disparities between plans increase as the horizon grows, leading to poor consistency between plans in consecutive steps. Besides, in practice, agents often struggle to reach the planned distant state. These facts argue that while long-horizon planning helps improve foresight, it introduces redundant information in distinct parts. The details in closer parts are more crucial. Ignoring the modeling of these redundant parts in the diffusion planning process will significantly reduce the complexity of the trajectory distribution to be fitted, making it possible to build a fast and lightweight diffusion planning framework.

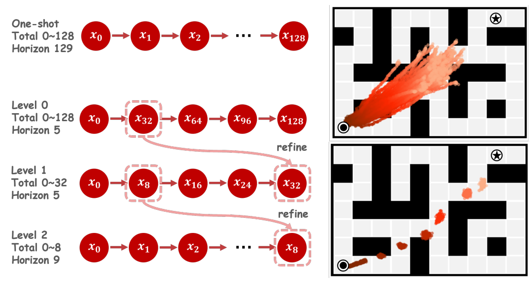

Figure 2: **Comparison of one-shot planning (top) and PRP (down) on Antmaze.** The former directly generates plans with a temporal horizon of 129. The latter consists of three coarse to fine-grained levels with temporal horizons of 0-128, 0-32, and 0-8, and temporal jumps of 32, 8, and 1, respectively. The visualization in the figure illustrates the x-y coordinates of 100 plans. It shows that one-shot planning exhibits a significant amount of redundant information and a large search space. In contrast, PRP demonstrates better plan consistency and a smaller search space.

Motivated by these insights, we propose to build a plan refinement process (PRP) to speed up diffusion planning. First, we perform "rough" planning, where jumpy planning is executed, only considering the states at intervals that are far apart and ignoring other individual states. Then, we refine a small portion of the plan, focusing on the steps closer to the current state. By doing so, we fill in the execution details between two states far apart, gradually refining the plan to the step level. This approach has three advantages: 1) It reduces the length of the sequences generated by the diffusion model, simplifying the complexity of the probability distribution to be fitted. 2) It significantly reduces the search space of plans, making it easier for the planner to find well-performed trajectories. 3) Since only the first action of each step is executed, rough planning of steps further away causes no noticeable performance drop.

Diffusion planning with PRP, which we call DiffuserLite, is simple, fast, and lightweight. Our experiments have demonstrated the effectiveness of PRP, significantly increasing decision-making frequency while achieving SOTA performance. Moreover, it can be easily adapted to other existing diffusion planning methods. In summary, our contributions are as follows:

- We introduce the plan refinement process (PRP) for coarse-to-fine-grained trajectory generation, reducing the modeling of redundant information.
- We introduce DiffuserLite, a lightweight diffusion planning framework, which significantly increases decision-making frequency by employing PRP.
- DiffuserLite is a simple and flexible plugin that can be easily combined with other diffusion planning algorithms.
- DiffuserLite achieves a super high decision-making frequency (122.2Hz, 112.7x faster than predominant frameworks) and SOTA performance on multiple benchmarks in D4RL.

2 Preliminaries

Problem Setup: Consider a system governed by discrete-time dynamics $o_{t+1} = f(o_t, a_t)$ at state o_t given an action a_t. A trajectory $\boldsymbol{x} = [x_0, \cdots, x_{T-1}]$ can be either a sequence of states $x_t = o_t$ or state-action pairs $x_t = (o_t, a_t)$, where T is the planning horizon. Each trajectory can be mapped to a property $\boldsymbol{c}$. Diffusion planning aims to find a trajectory that exhibits a property closest to the target:

$$\boldsymbol{x}^* = \arg\min_{\boldsymbol{x}} d(\mathcal{C}(\boldsymbol{x}), \boldsymbol{c}_{\text{target}}) \tag{1}$$

where d is a certain distance metric, $\mathcal{C}$ is a critic that maps a trajectory to the property it exhibits, and c_{target} is the target property. The action to be executed a_t is then extracted from the selected trajectory (*state-action sequences*) or predicted by an inverse dynamic model $a_t = h(o_t, o_{t+1})$ (*state-only sequences*). In the context of offline RL, it is a common choice to define the property as the corresponding cumulative reward $\mathcal{C}(\boldsymbol{x}) = \sum_{t=0}^{T-1} r(o_t, a_t)$ in previous works [20, 1].

Diffusion Models assume an unknown trajectory distribution $q_0(\boldsymbol{x}_0)$, DMs define a forward process $\{x_s\}_{s\in[0,S]}$ with $S > 0$. Starting with $\boldsymbol{x}_0$, previous work [23] proved that one can obtain any $\boldsymbol{x}_s$ by solving the following stochastic differential equation (SDE):

$$\mathrm{d}\boldsymbol{x}_s = f(s)\boldsymbol{x}_s\mathrm{d}s + g(s)\mathrm{d}\boldsymbol{w}_s,\ \boldsymbol{x}_0 \sim q_0(\boldsymbol{x}_0) \tag{2}$$

where $\boldsymbol{w}_s$ is the standard Wiener process, and $f(s) = \frac{\mathrm{d}\log\alpha_s}{\mathrm{d}s}$, $g^2(s) = \frac{\mathrm{d}\sigma_s^2}{\mathrm{d}s} - 2\sigma_s^2\frac{\mathrm{d}\log\alpha_s}{\mathrm{d}s}$. Values of $\alpha_s, \sigma_s \in \mathbb{R}^+$ depend on the noise schedule but keep the *signal-to-noise-ratio* (SNR) α_s^2/σ_s^2 strictly decreasing [23]. While this SDE transforms $q_0(\boldsymbol{x}_0)$ into a noise distribution $q_S(\boldsymbol{x}_S) = \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$, one can reconstruct trajectories from the noise by solving the reverse process of eq. (2). Previous work [46] proved that solving its associated *probability flow ODE* can support faster sampling:

$$\frac{\mathrm{d}\boldsymbol{x}_s}{\mathrm{d}s} = f(s)\boldsymbol{x}_s - \frac{1}{2}g^2(s)\nabla_{\boldsymbol{x}}\log q_s(\boldsymbol{x}_s),\ \boldsymbol{x}_S \sim q_S(\boldsymbol{x}_S) \tag{3}$$

in which *score function* $\nabla_{\boldsymbol{x}}\log q_s(\boldsymbol{x}_s)$ is the only unknown term and estimated by a neural network $-\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_s, s)/\sigma_s$ in practice. The parameter θ is optimized by minimizing the following objective:

$$\mathcal{L}(\theta) = \mathbb{E}_{q_0(\boldsymbol{x}_0), q(\boldsymbol{\epsilon}), s}[||\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_s, s) - \boldsymbol{\epsilon}||_2^2] \tag{4}$$

where $\boldsymbol{\epsilon} \sim q(\boldsymbol{\epsilon}) = \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$, $\boldsymbol{x}_s = \alpha_s\boldsymbol{x}_0 + \sigma_s\boldsymbol{\epsilon}$. Various ODE solvers can be employed to solve Equation (3), such as the Euler solver [2], RK45 solver [9], DPM solver [32], etc.

Conditional Sampling helps to generate trajectories exhibiting certain properties in a priori. There are two main approaches: classifier-guidance (CG) [7] and classifier-free-guidance (CFG) [18]. CG requires an additional classifier $\log p_\phi(\boldsymbol{c}|\boldsymbol{x}_s, s)$ to predict the log probability that a noisy trajectory $\boldsymbol{x}_s$ exhibits a given property $\boldsymbol{c}$. The gradients from this classifier are then used to guide the solver:

$$\tilde{\boldsymbol{\epsilon}}_\theta(\boldsymbol{x}_s, s, \boldsymbol{c}) := \boldsymbol{\epsilon}_\theta(\boldsymbol{x}_s, s) - w \cdot \sigma_s\nabla_{\boldsymbol{x}_s}\log p_\phi(\boldsymbol{c}|\boldsymbol{x}_s, s) \tag{5}$$

CFG does not require an additional classifier but uses a conditional noise predictor to guide the solver.

$$\tilde{\boldsymbol{\epsilon}}_\theta(\boldsymbol{x}_s, s, \boldsymbol{c}) := w \cdot \boldsymbol{\epsilon}_\theta(\boldsymbol{x}_s, s, \boldsymbol{c}) + (1-w) \cdot \boldsymbol{\epsilon}_\theta(\boldsymbol{x}_s, s) \tag{6}$$

Increasing the value of guidance strength w leads to more property-aligned generation, but decreases the legality of the generated trajectories [8].

3 Efficient Planning via Refinement

Diffuser [20] and DD [1] are two pioneering frameworks, upon which a vast amount of diffusion planning works has been built [28, 26, 53]. Although the design details differ, they can be unified into one paradigm. In the inference step, multiple candidate trajectories are first conditionally sampled using the diffusion model. Then, a critic is used to select the optimal one that exhibits the closest property to the target. Finally, the action to be executed is extracted. This paradigm relies on multiple forwarding complex neural networks, resulting in extremely low decision-making frequencies (typically 1-10Hz, or even less than 1Hz), severely hindering its real-world deployment.

https://doi.org/10.52202/079017-3895

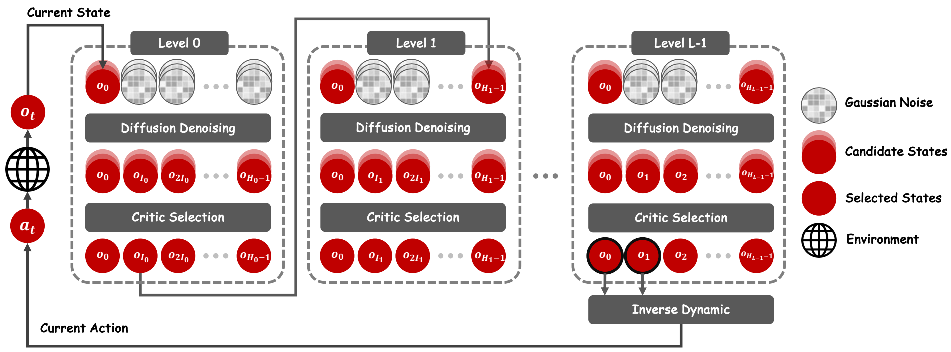

Figure 3: **Overview of DiffuserLite.** Observing the current state o_t, level 0 of DiffuserLite fixes o_t as o_0 and generates multiple candidate trajectories. A critic is then used to select the optimal one, in which o_{I_0} is then passed to the next level as its terminal o_{H_1-1}. The plan refinement process continues iteratively until the last level with a temporal jump of $I_{L-1}=1$. Finally, the action a_t to be executed is extracted using an inverse dynamic model $a_t = h(o_0, o_1)$.

The fundamental reason is the requirement for highly complex neural networks to model the complex long-horizon trajectory distributions [55]. Although some works have explored using advanced ODE solvers to reduce the sampling steps to around 5 [8], the time consumption of network forwarding is still unacceptable. However, we notice that *ignoring some redundant distant parts of the generated plan can be a possible solution* to address the issue. As shown in fig. 2, the disparities between plans increase as the horizon grows, leading to poor consistency between the plans selected in consecutive steps. Besides, agents often struggle to reach the distant states given by a plan in practice. Both facts indicate that terms in distant parts of a plan become increasingly redundant, whereas the closer parts are more crucial.

Regarding these findings, we aim to develop a progressive refinement planning (PRP) process. This process initially plans a rough trajectory consisting of only key points spaced at equal intervals and then progressively refines the first interval by generative interpolating. It is worth noting that this is orthogonal to methods of identifying key points based on task semantic information [27]. Specifically, our proposed PRP consists of L planning levels. At each level $l \in \{0, 1, \cdots, L-1\}$, starting from the known first term x_0, DMs plan rough trajectories $\boldsymbol{x}_{0:H_l:I_l}$ with temporal horizon H_l and temporal jump I_l. Then, the planned first key point x_{I_l} is passed to the next level as its terminal:

$$\begin{aligned} \boldsymbol{x}_{0:H_l:I_l} &:= [x_0, x_{I_l}, x_{2I_l} \cdots, x_{H_l-1}] \\ \boldsymbol{x}_{0:H_{l+1}:I_{l+1}} &:= \left[x_0, x_{I_{l+1}}, x_{2I_{l+1}}, \cdots, x_{H_{l+1}-1}\right] \\ x_{I_l} &= x_{H_{l+1}-1} \end{aligned} \tag{7}$$

By this design, only the first planned intervals are refined in the next level, and the other redundant details are all ignored, resulting in a coarse-to-fine generation process. The progressive refinement continues until the last level to extract an action. To support conditional sampling for each level, we define the property of a rough trajectory $\boldsymbol{x}_{0:H_l:I_l}$ as the property expectation over the distribution of all its completed trajectories $\mathcal{X}(\boldsymbol{x}_{0:H_l:I_l})$:

$$\mathcal{C}(\boldsymbol{x}_{0:H_l:I_l}) := \mathbb{E}_{\boldsymbol{x}\sim\mathcal{X}(\boldsymbol{x}_{0:H_l:I_l})}[\mathcal{C}(\boldsymbol{x})] \tag{8}$$

PRP ensures that long-term planning maintains foresight while alleviating the burden of modeling redundant information. As a result, it greatly contributes to reducing model size and improving planning efficiency:

Simplifying the fitted distribution of DMs. The absence of redundant details in PRP allows for a significant reduction in the complexity of the fitted distribution at each level. This reduction in complexity enables us to utilize a lighter neural network backbone, shorter network input sequence lengths, and a reduced number of denoising steps.

Reducing the plan-search space. Key points generated at former levels often sufficiently reflect the quality of the entire trajectory, which allows the planner to focus more on finding distant key points and planning actions for the immediate steps, reducing search space and complexity.

https://doi.org/10.52202/079017-3895

4 A Lite Architecture for Real-time Diffusion Planning

Employing PRP results in a new lightweight architecture for diffusion planning, which we refer to as DiffuserLite. DiffuserLite can reduce the complexity of the fit distribution and significantly increase the decision-making frequency, achieving 122Hz on average for the need of real-time control. We present the architecture overview in fig. 3, provide pseudocode for both training and inference in algorithm 1 and algorithm 2, and discuss detailed design choices in this section.

Diffusion model for each level: We train L diffusion models for all levels to generate state-only sequences. We employ DiT [40] as the noise predictor backbone, instead of the more commonly used UNet [43] due to the significantly reduced length of the generated sequences in each level (typically around 5). This eliminates the need for 1D convolution to extract local temporal features. To adapt the DiT backbone for temporal generation, we make minimal structural adjustments following [8]. For conditional sampling, we utilize CFG instead of CG, as the slow gradient computation process of CG reduces the frequency of decision-making. During the training phase, at each gradient step, we sample a batch of H_0-length trajectories, slice each of them into L sub-trajectories $[\boldsymbol{x}_{0:H_0:1}, \cdots, \boldsymbol{x}_{0:H_{L-1}:1}]$, evaluate their properties $\mathcal{C}(\boldsymbol{x}_{0:H_l:1})$ as an estimation of $\mathcal{C}(\boldsymbol{x}_{0:H_l:I_l})$ for condition at each level, and then slice the sub-trajectories into evenly spaced training samples $\boldsymbol{x}_{0:H_l:I_l}$ for training the diffusion models. During the inference phase, as shown in fig. 3, diffusion models generate multiple candidate plans level by level from 0 to $L-1$, and the optimal one is selected by the critic $\mathcal{C}$.

Critic design: The Critic $\mathcal{C}$ in DiffuserLite plays two important roles: providing generation conditions during the diffusion training process and selecting the optimal plan from the candidates generated by the diffusion model during inference. In the context of Offline RL, both Diffuser and DD adopt the cumulative reward of a trajectory as the condition:

$$\mathcal{C}(\boldsymbol{x}) = \sum_{t=0}^{H-1} r(o_t, a_t), \tag{9}$$

where H is the temporal horizon. This design allows rewards from the offline RL dataset to be utilized as a ground-truth critic for acquiring generation conditions during training. During inference, an additional trained reward function is required to serve as the critic. The critic then helps select the plan that maximizes the cumulative reward, as depicted in the lower part of fig. 3. However, this design poses challenges in tasks with sparse rewards, as it can confuse diffusion models when distinguishing better-performing trajectories, especially for short-horizon plans. To address this challenge, we introduce an option to use the sum of discounted rewards and the value of the last state as an additional property design:

$$\mathcal{C}(\boldsymbol{x}) = \sum_{t=0}^{H-2} \gamma^t r(o_t, a_t) + \gamma^{H-1} V(o_{H-1}), \tag{10}$$

where $V(o_t) = \max \mathbb{E}_\pi[\sum_{\tau=t}^{\infty} \gamma^{\tau-t} r_\tau]$ represents the optimal value function [47] and can be estimated by a neural network through various offline RL methods. In the context of other domains, properties can be flexibly designed as needed, as long as the critic $\mathcal{C}$ can evaluate trajectories of variable lengths. It is worth noting that previous diffusion planning algorithms widely support this flexibility. In addition, it is even possible to skip the critic selection during inference, which is equivalent to using a uniform critic, as used in DD.

Action extraction: After obtaining the optimal trajectory from the last level through critic selection, we utilize an additional inverse dynamic model $a_t = h(o_t, o_{t+1})$ to extract the action to be executed. This approach is suggested in [1].

Further speedup with rectified flow: DiffuserLite aims to achieve real-time diffusion planning to support its application in real-world scenarios. Therefore, we introduce *Rectified flow* [30] for further increasing the decision-making frequency. Rectified flow, an ODE on the time interval $[0, 1]$, causalizes the paths of linear interpolation between two distributions. If we define the two distributions as trajectory distribution and standard Gaussian, we can directly replace the Diffusion ODE with rectified flow to achieve the same functionality. The most significant difference is that rectified flow learns a straight-line flow and can continuously straighten the ODE through *reflow*. This straightness property allows for consistent and stable gradients throughout the flow, enabling the generation of trajectories with very few sampling steps (in our experiments, we found that one-step

https://doi.org/10.52202/079017-3895

Table 1: **Time consumption per step and frequency in D4RL.** All results are obtained over 5 random seeds. DiffuserLite achieves an average decision frequency of about 122Hz on the R2 backbone and 81Hz averaging over three variations, which meets the requirements of real-time inference. Since the code for HDMI is not open-source, we make every effort to reproduce HDMI with the original settings to test its runtime cost and thus mark its results with underlines.

Environment	Metric	Diffuser	DD	<u>HDMI</u>	DiffuserLite-D	DiffuserLite-R1	DiffuserLite-R2
MuJoCo	Runtime (s)	0.665	2.142	<u>0.405</u>	0.015	0.013	0.005
	Frequency (Hz)	1.5	0.47	<u>2.5</u>	68.2	79.7	200.7
Kitchen	Runtime (s)	0.790	2.573	<u>0.407</u>	0.017	0.015	0.010
	Frequency (Hz)	1.3	0.4	<u>2.5</u>	58.7	66.0	103.2
Antmaze	Runtime (s)	0.791	2.591	<u>0.410</u>	0.027	0.015	0.010
	Frequency (Hz)	1.3	0.39	<u>2.4</u>	37.3	65.7	101.7
Average Runtime (s)		0.749	2.435	<u>0.407</u>	**0.020**	**0.014**	**0.008**
Average Frequency (Hz)		1.34	0.41	<u>2.46</u>	**51.26**	**69.90**	**122.44**

sampling is sufficient to produce good results). We consider rectified flow an optional backbone for cases that prioritize decision frequency. In appendix B, we offer comprehensive explanations of trajectory generation and training with rectified flow.

5 Experiments

We explored the performance of the DiffuserLite on various tasks on D4RL, Robomimic, and FinRL [12, 34, 41], and aimed to answer the following research questions (RQs): 1) To what extent can DiffuserLite reduce the **runtime cost**? 2) How is the **performance** of DiffuserLite on offline RL tasks? 3) Can DiffuserLite serve as a **flexible plugin** for other diffusion planning algorithms? 4) Can we summarize a list of simple and clear **design choices** for DiffuserLite?

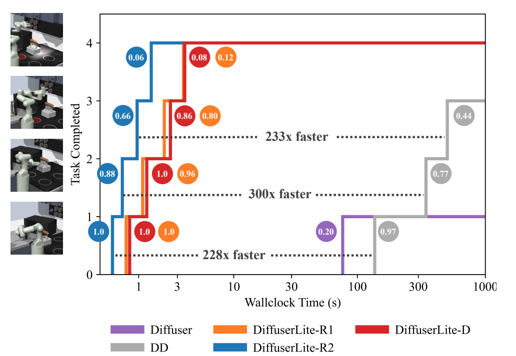

Figure 4: **Runtime and performance comparison in FrankaKitchen.** The y-axis represents the number of completed tasks (maximum of 4), and the x-axis represents the required wall-clock time. Task success rates are presented in colored circles. All results are averaged over 250 rollouts. DiffuserLite demonstrates significant advantages in both wallclock time and success rate.

5.1 Experimental Setup

Benchmarks: We evaluate the algorithm on various offline RL domains, including locomotion in Gym-MuJoCo [4], real-world manipulation in FrankaKitchen [13] and Robomimic [34], long-horizon navigation in Antmaze [12], and real-world stock trading in FinRL [41]. We train all models using publicly available datasets (see appendix A.1 for further details).

Baselines: Our comparisons mainly include the basic imitation learning algorithm BC, existing offline RL methods CQL [25] and IQL [24], as well as two pioneering diffusion planning frameworks, Diffuser [20] and DD [1]. Additionally, we compare with a state-of-the-art algorithm, HDMI [27], which is improved based on DD. (Further details about each baseline and the sources of performance results for each baseline across the experiments are presented in appendix A.2).

Backbones: As mentioned in Section 4, we implement three variations of DiffuserLite, based on different backbones: 1) diffusion model, 2) rectified flow, and 3) rectified flow with an additional *reflow* step. These variations will be used in subsequent experiments and indicated by the suffixes D, R1, and R2, respectively. All backbones utilize three levels, with a total temporal horizon of 129 in MuJoCo and Antmaze, and 49 in Kitchen. For more details about the hyperparameters selection for DiffuserLite, please refer to appendix D.

Computing Power: All runtime results across our experiments are obtained on a server equipped with an Intel(R) Xeon(R) Gold 6326 CPU @ 2.90GHz and an NVIDIA GeForce RTX3090.

Table 2: **D4RL Performance.** Results for DiffuserLite correspond to the mean and standard error over 5 random seeds. We detail the sources for the performance of prior methods in appendix A.2. Following Diffuser [20], we emphasize in bold scores within 5 percent of the maximum per task ($\geq 0.95 \cdot \max$).

Dataset	Environment	BC	CQL	IQL	Diffuser	DD	HDMI	DiffuserLite-D	DiffuserLite-R1	DiffuserLite-R2
Medium-Expert	HalfCheetah	55.2	**91.6**	86.7	79.8	**90.6**	**92.1**	88.5 ± 0.4	$\mathbf{90.8 \pm 0.9}$	84.0 ± 2.9
	Hopper	52.5	105.4	91.5	107.2	**111.8**	**113.5**	$\mathbf{111.6 \pm 0.2}$	$\mathbf{110.3 \pm 0.3}$	$\mathbf{110.1 \pm 0.5}$
	Walker2d	**107.5**	**108.8**	**109.6**	**108.4**	**108.8**	**107.9**	$\mathbf{107.1 \pm 0.6}$	$\mathbf{106.4 \pm 0.3}$	$\mathbf{106.1 \pm 0.7}$
Medium	HalfCheetah	42.6	44.0	**47.4**	44.2	**49.1**	**48.0**	$\mathbf{48.9 \pm 1.1}$	$\mathbf{48.6 \pm 0.7}$	45.3 ± 0.5
	Hopper	52.9	58.5	66.3	58.5	79.3	76.4	$\mathbf{100.9 \pm 1.1}$	$\mathbf{99.5 \pm 0.7}$	$\mathbf{96.8 \pm 0.3}$
	Walker2d	75.3	72.5	78.3	79.7	82.5	79.9	$\mathbf{88.8 \pm 0.6}$	$\mathbf{85.1 \pm 0.5}$	83.7 ± 1.0
Medium-Replay	HalfCheetah	36.6	**45.5**	**44.2**	42.2	39.3	**44.9**	41.6 ± 0.4	42.9 ± 0.4	39.6 ± 0.4
	Hopper	18.1	95.0	94.7	**96.8**	**100.0**	**99.6**	$\mathbf{96.6 \pm 0.3}$	$\mathbf{97.8 \pm 1.3}$	93.2 ± 0.7
	Walker2d	26.0	77.2	73.9	61.2	75.0	80.7	$\mathbf{90.2 \pm 0.5}$	84.6 ± 1.7	78.2 ± 1.7
Average		51.9	77.6	77	75.3	**81.8**	**82.6**	**86.0**	**85.1**	**81.9**
Mixed	Kitchen	51.5	52.4	51.0	50.0	65	69.2	$\mathbf{73.6 \pm 0.7}$	$\mathbf{71.9 \pm 1.4}$	64.8 ± 1.8
Partial	Kitchen	38.0	50.1	46.3	56.2	57	–	$\mathbf{74.4 \pm 0.6}$	69.9 ± 0.7	$\mathbf{71.4 \pm 1.2}$
Average		44.8	51.3	48.7	53.1	61.0	–	**74.0**	**70.9**	68.1
Play	Antmaze-Medium	0.0	65.8	65.8	0.0	0.0	–	78.0 ± 2.2	$\mathbf{88.0 \pm 2.2}$	$\mathbf{88.8 \pm 3.2}$
	Antmaze-Large	0.0	20.8	42.0	0.0	0.0	–	$\mathbf{72.0 \pm 6.2}$	$\mathbf{72.4 \pm 2.3}$	$\mathbf{69.4 \pm 6.5}$
Diverse	Antmaze-Medium	0.8	67.3	73.8	0.0	0.0	–	$\mathbf{92.4 \pm 3.2}$	$\mathbf{89.2 \pm 2.0}$	$\mathbf{87.6 \pm 2.0}$
	Antmaze-Large	0.0	20.5	30.3	0.0	0.0	–	68.0 ± 2.8	$\mathbf{80.4 \pm 5.1}$	75.2 ± 3.5
Average		0.2	43.6	53.0	0.0	0.0	–	77.6	**82.5**	**80.3**
Runtime per action (second)		–	–	–	0.749	2.435	–	**0.020**	**0.014**	**0.008**

Table 3: **Robomimic Performance.**

Dataset	BC	CQL	BCQ	IRIS	DiffuserLite-D
Lift-PH	**100.0**	92.7	**100.0**	**100.0**	**100.0**
Can-PH	95.3	38.0	88.7	**100.0**	**100.0**
Square-PH	78.7	5.3	50.0	78.7	**81.8**
Average	91.3	45.3	79.6	92.9	**93.9**

Table 4: **FinRL Performance.**

Dataset	BC	CQL	MB-PPO	DD	HDMI	DiffuserLite-D
FinRL-H-999	270	444	787	782	**801**	796
FinRL-M-999	504	621	698	712	754	**762**
Average	387	533	743	747	778	**779**

5.2 Runtime Cost (RQ1)

The primary objective of DiffuserLite is to increase the decision-making frequency. Therefore, we first test the wall-clock runtime cost (time consumption for one action inference) of DiffuserLite under three different backbones, compared to Diffuser, DD, and HDMI, to determine the extent of the advantage gained. We present the test results in table 1 [3], which shows that the runtime cost of DiffuserLite with D, R1, and R2 backbones is only 1.23%, 0.89%, and 0.51% of the average runtime cost of Diffuser and DD, respectively. **The remarkable improvement in decision-making frequency does not harm its performance.** As shown in fig. 4, compared to the average success rates of Diffuser (purple line) and DD (grey line) on four FrankaKitchen sub-tasks, DiffuserLite improves them by $[41.5\%, 56.2\%, 55.3\%, 8.7\%]$, respectively, while being 200-300 times faster. These improvements are attributed to ignoring redundant information in PRP, which reduces the complexity of the distribution that the backbone generative model needs to fit, allowing us to employ a light neural network backbone and use fewer sampling steps to conduct *perfect-enough* planning. Its success in FrankaKitchen, a realistic robot manipulation scenario, also reflects its potential application in real-world settings.

5.3 Performance (RQ2)

DiffuserLite is then evaluated on various popular domains in D4RL, Robomimic, and FinRL, to test how well it can maintain the performance when significantly increasing the decision-making frequency. All results are presented in table 2, table 3, and table 4, and the detailed descriptions of the sources of all baseline results are listed in appendix A.2. Results in *D4RL* table show significant performance improvements across all benchmarks with high decision-making frequency. This advantage is particularly pronounced in FrankaKitchen and Antmaze environments, indicating that the structure of DiffuserLite enables more accurate and efficient planning in long-horizon tasks, thus yielding greater benefits. In the MuJoCo environments, more notable advantages are shown on sub-optimal datasets, i.e., "medium" and "medium-replay" datasets. This sub-optimal advantage can be attributed to the PRP planning structure, which does not require one-shot generation of a consistent long trajectory, but explicitly demands stitching. This allows for better utilization of high-quality segments in low-quality datasets, leading to improved performance. Results in *Robomimic and*

[3] Since the code for HDMI is not open-source, we make every effort to reproduce HDMI with the original settings to test its runtime cost and thus underline its results in table 1.

 https://doi.org/10.52202/079017-3895

Table 5: **Integrate with DiffuserLite as a plugin.** We refer to AlignDiff with DiffuserLite plugin as AlignDiff-Lite. A larger value of *MAE Area* indicates a stronger alignment capability. AlignDiff-Lite greatly increases decision-making frequency, while only experiencing a small performance drop.

Metric	**GC**	**AlignDiff**	**AlignDiff-Lite**
MAE Area	0.319 ± 0.005	0.621 ± 0.023	0.601 ± 0.018 (3.2% ↓)
Frequency (Hz)	–	6.9	45.5 (560% ↑)

Table 6: **Performance of DiffuserLite with various PRP design choices.** The left part shows a comparison with 2/3/4 planning levels and the right part shows a comparison with 4 temporal horizon designs. Results correspond to the mean and standard error over 5 random seeds, the highest scores are emphasized in bold, and the default design choices used across other experiments are underlined.

	Temporal horizon of each level						
Planning horizon=129	[9,17]	[5,5,9]	[5,3,5,5]	[3,5,17]	[5,5,9]	[9,5,5]	[17,5,3]
HalfCheetah-me	75.6 ± 8.3	<u>$\mathbf{88.5 \pm 0.4}$</u>	88.3 ± 0.5	85.6 ± 0.6	<u>88.5 ± 0.4</u>	88.6 ± 0.7	$\mathbf{89.0 \pm 1.7}$
Antmaze-ld	0.0 ± 0.0	<u>68.0 ± 2.8</u>	$\mathbf{69.3 \pm 3.4}$	34.7 ± 4.1	<u>$\mathbf{68.0 \pm 2.8}$</u>	67.3 ± 3.4	34.0 ± 4.3
Planning horizon=49	[7,9]	[4,5,5]	[3,4,3,5]	[3,3,13]	[4,5,5]	[5,5,4]	[13,3,3]
Kitchen-p	72.8 ± 0.5	<u>$\mathbf{74.4 \pm 0.6}$</u>	72.3 ± 1.9	66.7 ± 1.7	<u>$\mathbf{74.4 \pm 0.6}$</u>	74.2 ± 0.6	31.7 ± 2.7

FinRL table are obtained by models trained on real-world datasets, and demonstrate that DiffuserLite continues to exhibit its superiority in these real-world tasks, achieving performance comparable to SOTA algorithms. This illustrates the potential application of DiffuserLite in real-world scenarios.

5.4 Flexible Plugin (RQ3)

To test the capability of DiffuserLite as a flexible plugin to support other diffusion planning algorithms, Aligndiff [8] is selected as a non-reward-maximizing algorithm backbone, and integrated with DiffuserLite plugin, referred to as AlignDiff-Lite. AlignDiff aims to customize the agent's behavior to align with human preferences and introduces an MAE area metric to measure this alignment capability (refer to appendix A.3 for more details), where a larger value indicates a stronger alignment capability. The comparison of AlignDiff and AlignDiff-Lite is presented in table 5, showing that AlignDiff-Lite achieves a 560% improvement in decision-making frequency compared to AlignDiff, while only experiencing a small performance drop of 3.2%. This result demonstrates the potential of DiffuserLite serving as a plugin to accelerate diffusion planning across various domains.

5.5 Ablations (RQ4)

How to choose the number of planning levels L and the temporal horizons H_l? We compared the performance of DiffuserLite with 2/3/4 planning levels and with four different temporal horizon designs, and reported the results in the left and right part of table 6, respectively. The list in the first row represents the temporal horizon for each level, with larger values on the left indicating more planning at a coarser granularity, while larger values on the right indicate more planning at a finer granularity. For planning level design, results show a performance drop with 2 planning levels, particularly in Antmaze, and a consistently excellent performance with 3 or 4 planning levels. This suggests that for longer-horizon tasks, it is advisable to design more planning levels. For temporal horizon design, results show a performance drop when the temporal horizon of one level becomes excessively long. This suggests the temporal horizon of each level is supposed to be similar and stay close. We summarized and presented a design choice list in appendix C.

Has the last level (short horizon) of DiffuserLite already performed well in decision making? This is equivalent to the direct use of the shorter planning horizon. If it is true, key points generated by former levels may not have an impact, making PRP meaningless. To address this question, we conduct tests using a one-level DiffuserLite with the same temporal horizon as the last level of the default model, referred to as **Lite** *w/ only last level*. The results are presented in table 7, column 2. The notable performance drop demonstrates the importance of a long-enough planning horizon.

Can decision-making be effectively accomplished without progressive refinement planning (PRP)? To address this question, we conduct tests using a one-level DiffuserLite with the same temporal horizon as the default model, referred to as **Lite** *w/o PRP*. This model only supports one-shot generation at the inference phase. We also test a "smaller" DD with the same network parameters and sampling steps as DiffuserLite, to verify whether one can speed up DD by simply

Table 7: **Ablation tests conducted to examine the effectiveness of PRP. Lite** *w/ only last level* and **Lite** *w/o PRP* are two ablated versions of DiffuserLite, while **DD**-*small* is a version of DD that uses the same network parameters and sampling steps as DiffuserLite. All results are obtained over 5 seeds. The varying degrees of performance drop observed in each experiment highlight the importance of PRP.

Environment	**Oracle**	**Lite** *w/ only last level*	**Lite** *w/o PRP*	**DD** *small*
Hopper-me	111.6 ± 0.2	27.8 ± 10.2	96.6 ± 1.0	67.9 ± 24.7
Hopper-m	100.9 ± 1.1	19.4 ± 2.0	66.4 ± 20.2	16.7 ± 8.0
Hopper-mr	96.6 ± 0.3	2.0 ± 0.2	62.5 ± 31.3	1.0 ± 0.4
Average	103.1	16.4 (84.1% ↓)	75.2 (27.1% ↓)	28.5 (72.3% ↓)
Kitchen-m	73.6 ± 0.7	48.2 ± 1.4	26.9 ± 1.0	0.0 ± 0.0
Kitchen-p	74.4 ± 0.6	38.6 ± 3.1	23.9 ± 0.5	1.8 ± 0.8
Average	74.0	43.4 (41.3% ↓)	25.4 (65.7% ↓)	0.8 (98.8% ↓)

reducing the parameters, referred to as **DD**-*small*. The results are presented in Table 7, column 3-4. The large standard deviation and the significant performance drop provide strong evidence for the limitations of one-shot generation planning, having difficulties in modeling the distribution of detailed long-horizon trajectories. However, DiffuserLite can maintain high performance with fast decision-making frequency due to its lite architecture and simplified fitted distribution.

More ablations. We conducted further ablation studies on model size, sampling steps, planning horizon, with or without value condition, and visual comparison to better elucidate DiffuserLite. Please refer to appendix E for these sections.

6 Related Works

Diffusion models are a type of score-based generative model [46]. Two pioneering frameworks, Diffuser [20] and DD [1], were the first to attempt using diffusion models for trajectory generation and planning in decision-making tasks. Based on these two frameworks, diffusion planning has been continuously improved and applied to various decision-making domains [36, 10, 54, 8, 17, 26, 21]. However, long-horizon estimation and prediction often suffer from potential exponentially increasing variance concerning the temporal horizon, called the "curse of horizon" [42]. To address this, HDMI [27] is the first algorithm that proposed a hierarchical decision framework to generate sub-goals at the upper level and reach goals at the lower level, achieving improvements in long-horizon tasks. However, HDMI is limited by cluster-based dataset pre-processing to obtain high-quality sub-goal data for upper training. Another concurrent work, HD-DA [5] introduces a similar hierarchical structure and allows the high-level diffusion model to automatically discover sub-goals from the dataset, achieving better results. However, the motivation behind DiffuserLite is completely different, which aims to increase the decision-making frequency of diffusion planning. Also, DiffuserLite allows for more hierarchy levels and refines only the first interval of the previous layer using PRP. Since HD-DA does not ignore redundant information, it fails to achieve a notable frequency increase. Compared to related works, DiffuserLite has more clean plugin design and undoubtedly contributes to increasing decision-making frequency and performance. We believe it can serve as a reference for the design of future diffusion planning frameworks.

7 Conclusion

In this paper, we introduce DiffuserLite, a super fast and lightweight diffusion planning framework that significantly increases decision-making frequency by employing the plan refinement process (PRP). PRP enables coarse-to-fine-grained trajectory generation, reducing the modeling of redundant information. Experimental results on various D4RL benchmarks demonstrate that DiffuserLite achieves a super-high decision-making frequency of 122.2Hz (112.7x faster than previous mainstream frameworks) while maintaining SOTA performance. DiffuserLite provides three generative model backbones to adapt to different requirements and can be flexibly integrated into other diffusion planning algorithms as a plugin. However, DiffuserLite currently has limitations mainly caused by the classifier-free guidance (CFG). CFG sometimes requires adjusting the target condition, which becomes more cumbersome on the multi-level structure of DiffuserLite. In future works, designing better guidance mechanisms, devising an optimal temporal jump adjustment, or integrating all levels in one diffusion model to simplify the framework is worth considering. DiffuserLite may also have some societal impacts, such as expediting the deployment of robotic products that could be utilized for military purposes.

https://doi.org/10.52202/079017-3895

8 Acknowledgements

This work is supported by the National Natural Science Foundation of China (Grant Nos. 62422605, 92370132, 62106172), the National Key R&D Program of China (Grant No. 2022ZD0116402) and the Xiaomi Young Talents Program of Xiaomi Foundation.

References

[1] 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, ICLR*, 2023.

[2] Kendall E. Atkinson. *An Introduction to Numerical Analysis*. John Wiley & Sons, 1989. URL http://www.worldcat.org/isbn/0471500232.

[3] Lei Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *arXiv preprint 1607.06450, ArXiv*, 2016.

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

[5] Chang Chen, Fei Deng, Kenji Kawaguchi, Caglar Gulcehre, and Sungjin Ahn. Simple hierarchical planning with diffusion. In *arXiv preprint 2401.02644, ArXiv*, 2024.

[6] Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Proceedings of Robotics: Science and Systems, RSS*, 2023.

[7] Prafulla Dhariwal and Alexander Quinn Nichol. Diffusion models beat GANs on image synthesis. In *Advances in Neural Information Processing Systems, NIPS*, 2021.

[8] Zibin Dong, Yifu Yuan, Jianye Hao, Fei Ni, Yao Mu, Yan Zheng, Yujing Hu, Tangjie Lv, Changjie Fan, and Zhipeng Hu. Aligndiff: Aligning diverse human preferences via behavior-customisable diffusion model. In *The Twelfth International Conference on Learning Representations, ICLR*, 2024.

[9] J. R. Dormand and P.J. Prince. A family of embedded runge-kutta formulae. *Journal of Computational and Applied Mathematics*, 1980.

[10] Yilun Du, Sherry Yang, Bo Dai, Hanjun Dai, Ofir Nachum, Joshua B. Tenenbaum, Dale Schuurmans, and Pieter Abbeel. Learning universal policies via text-guided video generation. In *Thirty-seventh Conference on Neural Information Processing Systems, NIPS*, 2023.

[11] Scott Emmons, Benjamin Eysenbach, Ilya Kostrikov, and Sergey Levine. Rvs: What is essential for offline RL via supervised learning? In *International Conference on Learning Representations, ICLR*, 2022.

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

[13] Abhishek Gupta, Vikash Kumar, Corey Lynch, Sergey Levine, and Karol Hausman. Relay policy learning: Solving long-horizon tasks via imitation and reinforcement learning. In *Proceedings of the Conference on Robot Learning, PMLR*, 2020.

[14] Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *International Conference on Machine Learning, ICML*, 2019.

[15] Nicklas Hansen, Xiaolong Wang, and Hao Su. Temporal difference learning for model predictive control. In *International Conference on Machine Learning, ICML*, 2022.

[16] Haoran He, Chenjia Bai, Kang Xu, Zhuoran Yang, Weinan Zhang, Dong Wang, Bin Zhao, and Xuelong Li. Diffusion model is an effective planner and data synthesizer for multi-task reinforcement learning. In *Thirty-seventh Conference on Neural Information Processing Systems, NIPS*, 2023.

[17] Haoran He, Chenjia Bai, Kang Xu, Zhuoran Yang, Weinan Zhang, Dong Wang, Bin Zhao, and Xuelong Li. Diffusion model is an effective planner and data synthesizer for multi-task reinforcement learning. *arXiv preprint arXiv:2305.18459*, 2023.

[18] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. In *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications, NIPS*, 2021.

[19] Mineui Hong, Minjae Kang, and Songhwai Oh. Diffused task-agnostic milestone planner. *arXiv preprint 2312.03395, ArXiv*, 2023.

[20] Michael Janner, Yilun Du, Joshua B. Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. In *International Conference on Machine Learning, ICML*, 2022.

[21] Chiyu Jiang, Andre Cornman, Cheolho Park, Benjamin Sapp, Yin Zhou, Dragomir Anguelov, et al. Motiondiffuser: Controllable multi-agent motion prediction using diffusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9644–9653, 2023.

[22] Bahjat Kawar, Shiran Zada, Oran Lang, Omer Tov, Huiwen Chang, Tali Dekel, Inbar Mosseri, and Michal Irani. Imagic: Text-based real image editing with diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR*, 2023.

[23] Diederik P Kingma, Tim Salimans, Ben Poole, and Jonathan Ho. Variational diffusion models. In *Advances in Neural Information Processing Systems, NIPS*, 2021.

[24] Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. In *The Tenth International Conference on Learning Representations, ICLR*, 2022.

[25] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. In *Advances in Neural Information Processing Systems, NIPS*, 2020.

[26] Kyowoon Lee, Seongun Kim, and Jaesik Choi. Refining diffusion planner for reliable behavior synthesis by automatic detection of infeasible plans. In *Advances in Neural Information Processing Systems, NIPS*, 2023.

[27] Wenhao Li, Xiangfeng Wang, Bo Jin, and Hongyuan Zha. Hierarchical diffusion for offline decision making. In *Proceedings of the 40th International Conference on Machine Learning, ICML*, 2023.

[28] Zhixuan Liang, Yao Mu, Mingyu Ding, Fei Ni, Masayoshi Tomizuka, and Ping Luo. Adaptdiffuser: Diffusion models as adaptive self-evolving planners. In *International Conference on Machine Learning, ICML*, 2023.

[29] 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. In *Proceedings of the 40th International Conference on Machine Learning, ICML*, 2023.

[30] 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.

[31] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR*, 2019.

[32] 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. In *Advances in Neural Information Processing Systems, NIPS*, 2022.

 https://doi.org/10.52202/079017-3895

[33] Cong Lu, Philip J. Ball, and Jack Parker-Holder. Synthetic experience replay. In *Workshop on Reincarnating Reinforcement Learning at ICLR*, 2023.

[34] Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. In *Conference on Robot Learning, CoRL*, 2021.

[35] Diganta Misra. Mish: A self regularized non-monotonic activation function. In *31st British Machine Vision Conference, BMVC*, 2020.

[36] Fei Ni, Jianye Hao, Yao Mu, Yifu Yuan, Yan Zheng, Bin Wang, and Zhixuan Liang. Metadiffuser: Diffusion model as conditional planner for offline meta-rl. *International Conference on Machine Learning, ICML*, 2023.

[37] Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models, 2021.

[38] Tim Pearce, Tabish Rashid, Anssi Kanervisto, Dave Bignell, Mingfei Sun, Raluca Georgescu, Sergio Valcarcel Macua, Shan Zheng Tan, Ida Momennejad, Katja Hofmann, and Sam Devlin. Imitating human behaviour with diffusion models. In *The Eleventh International Conference on Learning Representations, ICLR*, 2023.

[39] Tim Pearce, Tabish Rashid, Anssi Kanervisto, Dave Bignell, Mingfei Sun, Raluca Georgescu, Sergio Valcarcel Macua, Shan Zheng Tan, Ida Momennejad, Katja Hofmann, and Sam Devlin. Imitating human behaviour with diffusion models. In *The Eleventh International Conference on Learning Representations, ICLR*, 2023.

[40] William Peebles and Saining Xie. Scalable diffusion models with transformers. *arXiv preprint 2212.09748, ArXiv*, 2022.

[41] Rong-Jun Qin, Xingyuan Zhang, Songyi Gao, Xiong-Hui Chen, Zewen Li, Weinan Zhang, and Yang Yu. NeoRL: A near real-world benchmark for offline reinforcement learning. In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track, NIPS*, 2022.

[42] Tongzheng Ren, Jialian Li, Bo Dai, Simon S Du, and Sujay Sanghavi. Nearly horizon-free offline reinforcement learning. *Advances in neural information processing systems*, 34:15621–15634, 2021.

[43] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. *arXiv preprint 1505.04597, ArXiv*, 2015.

[44] Nataniel Ruiz, Yuanzhen Li, Varun Jampani, Yael Pritch, Michael Rubinstein, and Kfir Aberman. Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR*, 2023.

[45] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *9th International Conference on Learning Representations, ICLR*, 2021.

[46] 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, ICLR*, 2021.

[47] Richard S. Sutton and Andrew G. Barto. Reinforcement learning: An introduction. *IEEE Trans. Neural Networks*, 1998.

[48] Lei Tai, Jingwei Zhang, Ming Liu, Joschka Boedecker, and Wolfram Burgard. A survey of deep network solutions for learning control in robotics: From reinforcement to imitation. In *arXiv preprint 1612.07139, ArXiv*, 2018.

[49] Thomas George Thuruthel, Egidio Falotico, Federico Renda, and Cecilia Laschi. Model-based reinforcement learning for closed-loop dynamic control of soft robotic manipulators. *IEEE Transactions on Robotics*, 2019.

[50] 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, ICLR*, 2023.

[51] Tianhe Yu, Ted Xiao, Austin Stone, Jonathan Tompson, Anthony Brohan, Su Wang, Jaspiar Singh, Clayton Tan, Dee M, Jodilyn Peralta, Brian Ichter, Karol Hausman, and Fei Xia. Scaling robot learning with semantically imagined experience. In *Proceedings of Robotics: Science and Systems, RSS*, 2023.

[52] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision, ICCV*, 2023.

[53] Siyuan Zhou, Yilun Du, Shun Zhang, Mengdi Xu, Yikang Shen, Wei Xiao, Dit-Yan Yeung, and Chuang Gan. Adaptive online replanning with diffusion models. In *arXiv preprint 2310.09629, ArXiv*, 2023.

[54] Zhengbang Zhu, Minghuan Liu, Liyuan Mao, Bingyi Kang, Minkai Xu, Yong Yu, Stefano Ermon, and Weinan Zhang. Madiff: Offline multi-agent learning with diffusion models. In *arXiv preprint 2305.17330, ArXiv*, 2023.

[55] Zhengbang Zhu, Hanye Zhao, Haoran He, Yichao Zhong, Shenyu Zhang, Yong Yu, and Weinan Zhang. Diffusion models for reinforcement learning: A survey. In *arXiv preprint 2311.01223, ArXiv*, 2023.

https://doi.org/10.52202/079017-3895

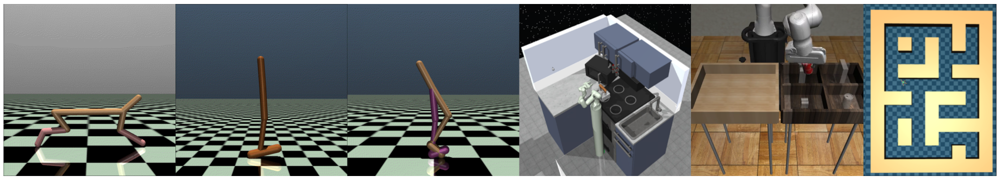

Figure 5: Part of selected benchmarks. From left to right, they are HalfCheetah, Hopper, Walker2d, FrankaKitchen, Robomimic, and Antmaze.

A Details of Experimental Setup

A.1 Test Domains

Gym-MuJoCo [4] on D4RL consists of three popular offline RL locomotion tasks (Hopper, HalfCheetah, Walker2d). These tasks require controlling three Mujoco robots to achieve maximum movement speed while minimizing energy consumption under stable conditions. D4RL provides three different quality levels of offline datasets: "medium" containing demonstrations of "medium" level performance, "medium-replay" containing all recordings in the replay buffer observed during training until the policy reaches "medium" performance, and "medium-expert" which combines "medium" and "expert" level performance equally.

FrankaKitchen [13] requires controlling a realistic 9-DoF Franka robot in a kitchen environment to complete several common household tasks. In offline RL testing, algorithms are often evaluated on "partial" and "mixed" datasets. The former contains demonstrations that partially solve all tasks and some that do not, while the latter contains no trajectories that completely solve the tasks. Therefore, these datasets place higher demands on the policy's "stitching" ability. During testing, the robot's task pool includes four sub-tasks, and the evaluation score is based on the percentage of tasks completed.

Antmaze [12] requires controlling the 8-DoF "Ant" quadruped robot in MuJoCo to complete maze navigation tasks. In the offline dataset, the robot only receives a reward upon reaching the endpoint, and the dataset contains many trajectory segments that do not lead to the endpoint, making it a difficult decision task with sparse rewards and a long horizon. The success rate of reaching the endpoint is used as the evaluation score, and common model-free offline RL algorithms often struggle to achieve good performance.

Robomimic [34] requires learning policies to complete complex manipulation tasks from a small number of human demonstrations. Due to the non-Markovian nature of human demonstrations and the demonstration quality variance, learning from human datasets is significantly more challenging than learning from machine-generated datasets. In our experiments, we used a dataset of PH (Proficient-Human) type, which consists of 200 demonstrations collected through teleoperation by one single experienced teleoperator.

FinRL [41] provides a way to build a trading simulator that replicates the real stock market and supports backtesting with important market frictions such as transaction costs, market liquidity, and investor risk aversion, among other factors. In the FinRL environment, one trade can be made per trading day for the stocks in the pool (30 stocks). The reward function is the difference in the total asset value between the end of the day and the day before. The environment may evolve itself as time elapses. In our experiments, the dataset has two suffixes, where "H" and "M" respectively indicate the quality of the dataset ("High" and "Medium"), and 999 indicates that the dataset includes 999 rollouts.

A.2 Baselines

A.2.1 Runtime Testing

- We run Diffuser [4] using the official repository from the original paper with default hyperparameters.

[4] https://github.com/jannerm/diffuser

- We run DD [5] using the official repository from the original paper with default hyperparameters.
- Since the code for HDMI has not been released, we made every effort to reproduce HDMI based on the DiffuserLite codebase, strictly adhering to every detail mentioned in the paper.

All runtime results are obtained on a server equipped with an Intel(R) Xeon(R) Gold 6326 CPU @ 2.90GHz and an NVIDIA GeForce RTX3090.

A.2.2 Reward-maximizing

- The performance of BC, CQL [25] and IQL [24] in table 2 is reported in the D4RL [12], Table 2;
- The performance of Diffuser [20] in table 2, MuJoCo, is reported in [20], Table 1; The performance in Kitchen is reported in [5], Table 2; And the performance in Antmaze is obtained by the official repository from the original paper with default hyperparameters.
- The performance of DD [1] in table 2, MuJoCo and Kitchen, is reported in [1], Table 1; And the performance in Antmaze is reported in [19], Table 1;
- The performance of HDMI [27] in table 2 is reported in [27], Table 3.

A.2.3 Behavior-customizing

- The performance of GC (goal conditioned BC) [11] and AlignDiff [8] in table 5 is reported in [8], Table 2.

A.3 Details of AlignDiff

Why AlignDiff? Since DiffuserLite can be seen as integrating PRP into DD, the main experiments in the paper can be considered as results obtained when DiffuserLite is used as a reward-maximizing algorithm plugin. Therefore, to evaluate the more general plugin capability of DiffuserLite, a non-reward-maximizing algorithm backbone must be selected. AlignDiff, unlike the common setting in Offline RL, uses diffusion planning for behavior-customizing, which aligns with our requirements.

AlignDiff [8] is a diffusion planning algorithm used for behavior-customizing. In our experiments, we integrate DiffuserLite as a plugin into AlignDiff, achieving a significant increase in decision-making frequency of approximately 560% with minimal performance loss (around 3.2%). This demonstrates the flexibility and effectiveness of DiffuserLite serving as a plugin. In this section, we provide a detailed description of the problem setting for AlignDiff and the evaluation metric used to assess the algorithm's performance, aiming to help the understanding of the experimental content.

Problem setting: AlignDiff considers a reward-free Markov Decision Process (MDP) denoted as $\mathcal{M} = \langle S, A, P, \boldsymbol{\alpha} \rangle$. Here, S represents the set of states, A represents the set of actions, $P : S \times A \times S \to [0, 1]$ is the transition function, and $\boldsymbol{\alpha} = \{\alpha_1, \cdots, \alpha_k\}$ represents a set of k predefined attributes used to characterize the agent's behaviors. Given a state-only trajectory $\tau^l = \{s_0, \cdots, s_{l-1}\}$, it assumes the existence of an attribute strength function that maps the trajectory to a relative strength vector $\zeta^{\boldsymbol{\alpha}}(\tau^l) = \boldsymbol{v}^{\boldsymbol{\alpha}} = [v^{\alpha_1}, \cdots, v^{\alpha_k}] \in [0, 1]^k$. Each element of the vector indicates the relative strength of the corresponding attribute. A value of 0 for v^{α_i} implies the weakest manifestation of attribute α_i, while a value of 1 represents the strongest manifestation. Human preferences are formulated as a pair of vectors $(\boldsymbol{v}^{\boldsymbol{\alpha}}_{\text{targ}}, \boldsymbol{m}^{\boldsymbol{\alpha}})$, where $\boldsymbol{v}^{\boldsymbol{\alpha}}_{\text{targ}}$ represents the target relative strengths, and $\boldsymbol{m}^{\boldsymbol{\alpha}} \in \{0, 1\}^k$ is a binary mask indicating which attributes are of interest. The objective is to find a policy $a = \pi(s|\boldsymbol{v}^{\boldsymbol{\alpha}}_{\text{targ}}, \boldsymbol{m}^{\boldsymbol{\alpha}})$ that minimizes the L1 norm $||(\boldsymbol{v}^{\boldsymbol{\alpha}}_{\text{targ}} - \zeta^{\boldsymbol{\alpha}}(\mathbb{E}_\pi[\tau^l])) \circ \boldsymbol{m}^{\boldsymbol{\alpha}}||_1$, where $\circ$ denotes the Hadamard product.

Area metric: To evaluate the algorithm's performance, the authors suggest that one can conduct multiple trials to collect the mean absolute error (MAE) between the evaluated and target relative strengths. For each trial, we need to sample an initial state s_0, a target strengths $\boldsymbol{v}^{\boldsymbol{\alpha}}_{\text{targ}}$, and a mask $\boldsymbol{m}^{\boldsymbol{\alpha}}$, as conditions for the execution of each algorithm. Subsequently, the algorithm runs for T steps, resulting in the exhibited relative strengths $\boldsymbol{v}^{\boldsymbol{\alpha}}$ evaluated by $\hat{\zeta}_\theta$. Then we can calculate the percentage of samples that fell below pre-designed thresholds to create an MAE curve. The area enclosed by

[5] https://github.com/anuragajay/decision-diffuser/tree/main/code

https://doi.org/10.52202/079017-3895

the curve and the axes can be used to define an area metric. A larger metric value indicates better performance in matching. By integrating DiffuserLite, AlignDiff can maintain almost the same level of performance with only a 3.2% decrease. The overall performance is nearly twice as good as the goal-conditioned behavior clone (GC) while achieving a significant increase in decision frequency of 560%.

B Details of Rectified Flow

Similar to DMs, rectified flow [30] is also a probability flow-based generative model that learns a transfer from q_0 to q_1 through an ODE. In trajectory generation, we can define q_1 as the distribution of trajectories and q_0 as the standard Gaussian. The learned ODE can be represented as:

$$\mathrm{d}\boldsymbol{x}_s = v(\boldsymbol{x}_s, s)ds, \text{ initialized from } \boldsymbol{x}_0 \sim q_0, \text{ such that } \boldsymbol{x}_1 \sim q_1 \tag{11}$$

where $v : \mathbb{R}^d \times [0, 1] \to \mathbb{R}^d$ is a velocity field, learned by minimizing a simple mean square objective:

$$\min_v \mathbb{E}_{(\boldsymbol{x}_0,\boldsymbol{x}_1)\sim\gamma}[\int_0^1 ||\frac{\mathrm{d}}{\mathrm{d}s}\boldsymbol{x}_s - v(\boldsymbol{x}_s, s)||^2 \mathrm{d}s], \text{ with } \boldsymbol{x}_s = (1 - s)\boldsymbol{x}_0 + s\boldsymbol{x}_1 \tag{12}$$

where γ is any coupling of (q_0, q_1), and v is parameterized as a deep neural network and eq. (12) is solved approximately with stochastic gradient methods. A key property of rectified flow is its ability to learn a straight flow, which means:

$$\textit{Straight flow: } \boldsymbol{x}_s = s\boldsymbol{x}_s + (1 - s)\boldsymbol{x}_0 = \boldsymbol{x}_0 + sv(\boldsymbol{x}_0, 0), \forall s \in [0, 1] \tag{13}$$

A straight flow can achieve *perfect* results with fewer sample steps (even a single step).

Reflow is an iterative procedure to straighten the learned flow without modifying the marginal distributions, hence allowing faster sampling at inference time. Assume we have an ODE model $\mathrm{d}\boldsymbol{x}_s = v_k(\boldsymbol{x}_s, s)\mathrm{d}s$ with velocity field v_k at the k-th iteration of the reflow procedure; denote by $\boldsymbol{x}_1 = \mathtt{ODE}[v_k](\boldsymbol{x}_0)$ the $\boldsymbol{x}_s$ we obtained at $t = 1$ when following the v_k-ODE starting from $\mathbf{0}$. A reflow step turns v_k into a new vector field v_{k+1} that yields straighter ODEs while $\boldsymbol{x}_1^{\text{new}} = \mathtt{ODE}[v_{k+1}](\boldsymbol{x}_0)$ has the same distribution as $\boldsymbol{x}_1 = \mathtt{ODE}[v_k](\boldsymbol{x}_0)$,

$$\begin{aligned} v_{k+1} = \arg\min_v \mathbb{E}_{\boldsymbol{x}_0\sim q_0}[\int_0^1 ||(\boldsymbol{x}_1 - \boldsymbol{x}_0) - v(\boldsymbol{x}_s, s)||^2 \mathrm{d}s], \\ \text{with } \boldsymbol{x}_1 = \mathtt{ODE}[v_k](\boldsymbol{x}_0) \text{ and } \boldsymbol{x}_s = s \cdot \boldsymbol{x}_1 + (1 - s) \cdot \boldsymbol{x}_0, \end{aligned} \tag{14}$$

where v_{k+1} is learned using the same rectified flow objective eq. (12), but with the linear interpolation of $(\boldsymbol{x}_0, \boldsymbol{x}_1)$ pairs constructed from the previous $\mathtt{ODE}[v_k]$. For conditional sampling, rectified flow also supports classifier-free guidance, in which we can train a conditional velocity field $v(\boldsymbol{x}_s, s|\boldsymbol{c})$ and apply CFG by:

$$v^w(\boldsymbol{x}_s, s|\boldsymbol{c}) = wv(\boldsymbol{x}_s, s|\boldsymbol{c}) + (1 - w)v(\boldsymbol{x}_s, s), \tag{15}$$

where w is the guidance strength. In this way, we can directly replace diffusion models with rectified flow as the backbone of DiffuserLite. In our experiments, we find that rectified flow achieves similar performance to the diffusion backbone when using the same number of sampling steps and neural network size. By further conducting a *reflow* procedure, the planning of the model becomes more stable under the same number of sampling steps (as evidenced by a decrease in the variance of experimental results). It is even possible to reduce the number of sampling steps to just one, resulting in only a small performance drop.

C PRP Design Insights

As the core planning framework of DiffuserLite, PRP design choices are crucial. Through experimental results and discussions in section 5.5, some simple, stable, and effective PRP design insights can be summarized:

- The planning horizon T is supposed to be determined by the nature of the task. The sparser the rewards and the longer the episodes, the longer the planning horizon should be used.

https://doi.org/10.52202/079017-3895

- The number of planning levels L is supposed to be designed based on the planning horizon, with longer horizon tasks requiring more planning levels. One can start from 3 or 4 planning levels.
- The temporal horizon of each level H_l is supposed to be similar and limited to within 10 to increase decision-making frequency.
- Once the number of planning levels and temporal horizon are determined, the temporal jump is also determined.

Throughout our experiments, we consistently applied this set of design choices, achieving consistently excellent performance without the need for carefully adjusting parameters.

D Implementation Details

We introduce the implementation details of DiffuserLite in the section:

- We utilize DiT [40] as the neural network backbone for all diffusion models and rectified flows, with an embedding dimension of 256, 8 attention heads, and 2 DiT blocks. We progressively reduce the model size from DiT-S [40] until the current size setting, and there is still no significant performance drop. This suggests that future works could even explore further reduction of network parameters to achieve faster decision speeds.
- Across all the experiments, we employ DiffuserLite with 3 levels. In Kitchen, we utilize a planning horizon of 49 with temporal jumps for each level set to 16, 4, and 1, respectively. In MuJoCo and Antmaze, we use a planning horizon of 129 with temporal jumps of 32, 8, and 1 for each level, respectively.
- For diffusion models, we use cosine noise schedule [37] for α_s and σ_s with diffusion steps $T = 1000$. We employ DDIM [45] to sample trajectories. In MuJoCo and Kitchen, we use 3 sampling steps, while in Antmaze, we use 5 sampling steps.
- For rectified flows, we use the Euler solver with 3 steps for all benchmarks. After one reflow procedure, we can further reduce it to 1 step for MuJoCo and 2 steps for Kitchen and Antmaze.
- All models utilize the AdamW optimizer [31] with a learning rate of $2e-4$ and weight decay of $1e-5$. We perform 500K gradient updates with a batch size of 256. We do not employ the exponential moving average (EMA) model, as used in [20, 1]. We found that using the EMA model did not yield significant gains in our experiments. For *reflow* training, we first use the trained rectified flow to generate a 2M dataset with 20 sampling steps. Then we use the same optimizer, but with a learning rate of $2e-5$, to train the model for 200K gradient steps.
- For conditional sampling, we tune the guidance strength w within the range of $[0, 1]$. In general, a higher guidance strength leads to better performance but may result in unrealistic plans and instability. On the other hand, a lower guidance strength provides more stability but may lead to a decrease in performance. **In our implementation of DiffuserLite, we observe that the earlier levels are more closely related to decision-making. In contrast, the later levels only need to ensure reaching the key points provided by the previous levels. Therefore, we only apply conditional sampling to level 0, while the other levels are not guided.**
- In MuJoCo, we only utilize the cumulative rewards as the generation condition. However, in Kitchen and Antmaze, we employ the discounted cumulative return and the value of the last generated state as the generation condition. The values are evaluated using a pre-trained IQL-value function [24]. We have found that this generation condition is beneficial for training models in reward-sparse environments, as it helps to prevent the model from becoming confused by suboptimal or poor trajectories.
- The inverse dynamic model is implemented as a 3-layer MLP. The first two layers consist of a Linear layer followed by a Mish activation [35] and a LayerNorm [3]. And, the final layer is followed by a Tanh activation. This model utilizes the same optimizer as the diffusion models and is trained for 200K gradient steps.

 https://doi.org/10.52202/079017-3895

Table 8: **Performance of DiffuserLite under four different model sizes.** All results are obtained over 5 seeds. The results under default choice are underlined and the highest scores are emphasized in bold.

Environment	Model sizes			
	0.68M	**1.53M**	**2.7M**	**4.22M**
HalfCheetah-me	75.4 ± 5.1	86.9 ± 0.6	$\underline{88.5 \pm 0.4}$	$\mathbf{90.6 \pm 1.1}$
Runtime (s)	$\mathbf{0.0153}$	0.0154	$\underline{0.0155}$	0.0156
Antmaze-ld	0.3 ± 0.5	60.7 ± 2.5	$\underline{68.0 \pm 2.8}$	$\mathbf{73.0 \pm 3.6}$
Runtime (s)	$\mathbf{0.0262}$	0.0268	$\underline{0.0270}$	0.0271
Kitchen-p	64.5 ± 0.8	74.8 ± 0.2	$\underline{74.4 \pm 0.6}$	$\mathbf{75.0 \pm 0.0}$
Runtime (s)	$\mathbf{0.0163}$	0.0169	$\underline{0.0170}$	0.0172

Table 9: **Performance of DiffuserLite with different sampling steps.** All results are obtained over 5 seeds. The results under default choice are underlined and the highest scores are emphasized in bold.

Environment	Sampling steps			
	1	**3**	**5**	**10**
HalfCheetah-me	74.4 ± 15.7	$\underline{88.5 \pm 0.4}$	89.0 ± 0.7	$\mathbf{89.2 \pm 1.3}$
Runtime (s)	$\mathbf{0.0065}$	$\underline{0.0155}$	0.026	0.048
Antmaze-ld	0.7 ± 0.9	16.7 ± 2.5	$\underline{68.0 \pm 2.8}$	$\mathbf{74.3 \pm 5.4}$
Runtime (s)	$\mathbf{0.0068}$	0.0167	$\underline{0.0270}$	0.050
Kitchen-p	5.7 ± 1.2	$\underline{74.4 \pm 0.6}$	73.8 ± 0.8	$\mathbf{74.7 \pm 0.5}$
Runtime (s)	$\mathbf{0.0068}$	$\underline{0.0170}$	0.029	0.053

E Additional Experiment Results

E.1 Performance with Different Model Sizes

We compare the performance of DiffuserLite using four different model sizes. Due to the use of DiT as the backbone, we fix the dimensions of each attention head, and the parameter design of the Transformer is as follows: [`hidden_size`, `n_heads`, `depth`] respectively set to [128, 4, 2] (0.68M), [192, 6, 2] (1.53M), [256, 8, 2] (2.7M), and [320, 10, 2] (4.22M). The results are presented in table 8, which shows that slightly increasing the model size does not significantly decrease the inference speed, but it does improve the performance. This suggests that increasing the model parameter size can further enhance the performance of DiffuserLite.

E.2 Performance with Different Sampling Steps

We compare the performance of DiffuserLite using four different sampling steps. The results are presented in table 9, which shows a natural trade-off: more sampling steps lead to better performance but at the cost of slower decision-making speed. DiffuserLite strikes a balance between performance and speed by offering the choice of 3 or 5 sampling steps. Researchers can flexibly adjust the sampling steps based on the specific requirements of their tasks.

E.3 Performance under Different Planning Horizon Choices

The planning horizon is an essential parameter that influences the performance of planning algorithms. To investigate the performances of DiffuserLite under different temporal horizon choices, we test it using three temporal horizon choices: 49, 129, and 257. The results are presented in table 10. In Hopper environment, a longer planning horizon is required to avoid greedy and rapid jumps that may lead to falls. Consequently, the performance of DiffuserLite is slightly poorer under the 49 temporal horizon compared to the other two choices (129 and 257), where no significant performance differences are observed. For Kitchen environment, the total length of the episode (280 time steps) in the dataset poses a limitation. An excessively long planning horizon can confuse the model, as it may struggle to determine the appropriate actions to take after completing all tasks. As a result, the performance of DiffuserLite is poor under the 257 temporal horizon, while no significant performance differences are observed under the 49 and 129 temporal horizons.

https://doi.org/10.52202/079017-3895

Figure 6: **Visual comparison between one-shot generated plans (upper) and PRP (lower) in Hopper.** The figure showcases 100 diffusion-generated plans starting from the same initial state. The one-shot generated plans directly generate all states from $t = 0$ to $t = 96$, while PRP utilizes three levels with temporal horizons of 97, 33, and 9, and temporal jumps of 32, 8, and 1, respectively. For ease of observation, only Hopper motion every 4 steps is displayed in the figure.

Table 10: **Performance of DiffuserLite under three different temporal horizon choices.** All results are obtained over 5 seeds and the results of default choice are emphasized in bold scores.

Environment	**Temporal Horizon - 1**		
	48	**128**	**256**
Hopper-me	101.1 ± 0.5	$\mathbf{111.6 \pm 0.2}$	110.0 ± 0.4
Hopper-m	96.6 ± 10.9	$\mathbf{100.9 \pm 1.1}$	98.9 ± 0.4
Hopper-mr	74.5 ± 27.2	$\mathbf{96.6 \pm 0.3}$	98.2 ± 2.1
Average	90.7	**103.1**	102.4
Kitchen-m	$\mathbf{73.6 \pm 0.7}$	72.7 ± 1.0	25.1 ± 0.2
Kitchen-p	$\mathbf{74.4 \pm 0.6}$	75.0 ± 0.0	25.2 ± 0.4
Average	**74.0**	73.9	25.2

E.4 Performance with or without Value Condition

DiffuserLite offers two optional generation conditions, resulting in two different critics, as introduced in section 4. To determine the suitability of these two approaches in different tasks, we conduct ablation experiments and present the results in table 11. We find that in dense reward tasks, such as Hopper and HalfCheetah, the performance of both conditions is nearly identical. However, in sparse reward scenarios, such as Kitchen and Antmaze, the use of values demonstrates a significant advantage. We present a visual comparison in Antmaze in fig. 7, which shows 100 generated plans. With a pure-rewards condition, it is difficult to discern the endpoint location, as the planner often desires to stop at a certain point on the map. However, when using values, the planner indicates a desire to move towards the endpoint. This suggests that sparse reward tasks are prone to confusing the conditional generative model, leading to poor planning. The introduction of value-assisted guidance can address this issue.

E.5 Visual Comparison between One-shot Generated Plans and PRP

We visually compare the one-shot generation and PRP in Hopper, presenting the results in appendix E.5. Regarding efficiency, PRP significantly reduces the length of sequences that need to be generated due to each level refining only the first jumpy interval of the previous level. This results in a noticeable increase in the forward speed of neural networks, especially for transformer-based backbone models like DiT [40]. Visually, we can more clearly perceive the difference in generated sequence lengths, and this advantage will further expand as the total planning horizon increases. In terms of quality, PRP narrows down the search space of planning, leading to better consistency among different plans, while the one-shot generated plans exhibit more significant divergence in the far horizon.

 https://doi.org/10.52202/079017-3895

Algorithm 1 DiffuserLite Training

Input: number of planning levels L, temporal horizon H_l, temporal jump I_l and noise estimators ϵ_{θ_l} for each level $l \in \{0, 1, \cdots, L-1\}$, dataset $\mathcal{D} = \{\boldsymbol{x}\}$, where $\boldsymbol{x}$ is the sequence of state-action pairs, critic $\mathcal{C}$, diffusion steps T, condition mask probability $1-p$;
while not done **do**
  sample a batch of $(\boldsymbol{x}, \mathcal{C}(\boldsymbol{x}))$ from dataset $\mathcal{D}$
  for $l = 0$ **to** $L-1$ **do**
    extract $\boldsymbol{x}_{0,H_l,1}$ and $\boldsymbol{x}_{0,H_l,I_l}$ from $\boldsymbol{x}$
    $\hat{\mathcal{C}}(\boldsymbol{x}_{0,H_l,I_l}) \leftarrow \mathcal{C}(\boldsymbol{x}_{0,H_l,1})$
    sample $s_l \sim$ Uniform(T), $\boldsymbol{\epsilon}_l \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$
    first state of $\boldsymbol{\epsilon}_l \leftarrow$ first state of $\boldsymbol{x}_{0,H_l,I_l}$
    if $l > 0$ **then**
      last state of $\boldsymbol{\epsilon}_l \leftarrow$ last state of $\boldsymbol{x}_{0,H_l,I_l}$
    end if
    $\boldsymbol{x}^s_{0:H_l:I_l} \leftarrow \alpha_s \boldsymbol{x}_{0:H_l:I_l} + \sigma_s \boldsymbol{\epsilon}_l$
    update θ_l by minimizing
    $||\boldsymbol{\epsilon}_{\theta_l}(\boldsymbol{x}^s_{0:H_l:I_l}, s, \mathcal{C}(\boldsymbol{x}_{0:H_l:I_l})) - \boldsymbol{\epsilon}||^2$ with probability p else
    $||\boldsymbol{\epsilon}_{\theta_l}(\boldsymbol{x}^s_{0:H_l:I_l}, s)) - \boldsymbol{\epsilon}||^2$
  end for
end while

Algorithm 2 DiffuserLite Inference

Input: number of planning levels L, temporal horizon H_l, temporal jump I_l and noise estimators ϵ_{θ_l} for each level $l \in \{0, 1, \cdots, L-1\}$, critic $\mathcal{C}$, inverse dynamic h, diffusion steps T, current state o_t;
for $l = 0$ **to** $L-1$ **do**
  sample $\boldsymbol{\epsilon}_l \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$.
  first state of $\boldsymbol{\epsilon}_l \leftarrow o_t$
  if $l > 0$ **then**
    last state of $\boldsymbol{\epsilon}_l \leftarrow$ last state of $\boldsymbol{x}_{0:H_{l-1}:I_{l-1}}$
  end if
  obtain $\boldsymbol{x}_{0:H_l:I_l}$ by solving diffusion ODE with DDIM solver
end for
extract o_t, o_{t+1} from $\boldsymbol{x}_{0:H_{L-1}:1}$
$a_t = h(o_t, o_{t+1})$

Table 11: **Performance of DiffuserLite using conditions with or without values.** Present only average performance on varying-quality datasets, which are obtained over 5 seeds.

Condition	Hopper	HalfCheetah	Kitchen	Antmaze
w/ Value	103.6	60.7	74.0	77.6
w/o Value	103.1	59.7	54.1	19.7

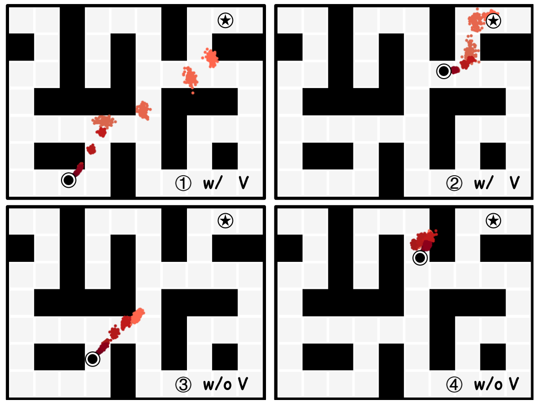

Figure 7: **Visual comparison of DiffuserLite using conditions with (upper) or without (lower) values.** It displays 100 plans generated at the current state, where darker colors indicate closer proximity to the current state, and lighter colors indicate further. With the pure-rewards condition, we can observe that the planned states in lighter colors tend to cluster together at a certain point on the map, indicating the planner tries to stop at that non-endpoint. However, with the introduction of values, the planner can make correct long-term plans that lead to the endpoint.

https://doi.org/10.52202/079017-3895

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: See the abstract's last three sentences and the introduction's last paragraph in Section 1.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: See the conclusion in Section 7.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

Justification: The paper does not include theoretical results.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: We have the code and model checkpoints ready for release. Besides, in Appendix D, we provide sufficient implementation details for researchers to reproduce the results.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

https://doi.org/10.52202/079017-3895

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: The code and model checkpoints have been released.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: We provided full details in Section 5.1, Appendix A.1 and Appendix A.2.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We provided the mean and standard error over several random seeds in the experimental results to demonstrate statistical significance.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We detailed the compute resources used for the experiments in Section 5.1.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research conducted in the paper conforms, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: See the conclusion in Section 7.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.

https://doi.org/10.52202/079017-3895

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: For all the datasets and algorithm baselines used in the paper, we have cited the original papers and provided the license, copyright information, and terms of use in the package in our code repository.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, paperswithcode.com/datasets has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: New assets introduced in the paper are well documented and the documentation is provided alongside the assets.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing or research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing or research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

https://doi.org/10.52202/079017-3895

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.