Resisting Stochastic Risks in Diffusion Planners with the Trajectory Aggregation Tree

Lang Feng [1] **Pengjie Gu** [2] **Bo An** [2 3] **Gang Pan** [1 4]

Abstract

Diffusion planners have shown promise in handling long-horizon and sparse-reward tasks due to the non-autoregressive plan generation. However, their inherent stochastic risk of generating infeasible trajectories presents significant challenges to their reliability and stability. We introduce a novel approach, the Trajectory Aggregation Tree (TAT), to address this issue in diffusion planners. Compared to prior methods that rely solely on raw trajectory predictions, TAT aggregates information from both historical and current trajectories, forming a dynamic tree-like structure. Each trajectory is conceptualized as a branch and individual states as nodes. As the structure evolves with the integration of new trajectories, unreliable states are marginalized, and the most impactful nodes are prioritized for decision-making. TAT can be deployed without modifying the original training and sampling pipelines of diffusion planners, making it a training-free, ready-to-deploy solution. We provide both theoretical analysis and empirical evidence to support TAT's effectiveness. Our results highlight its remarkable ability to resist the risk from unreliable trajectories, guarantee the performance boosting of diffusion planners in 100% of tasks, and exhibit an appreciable tolerance margin for sample quality, thereby enabling planning with a more than $3\times$ acceleration.

1. Introduction

Planning is a crucial aspect of decision-making and has led to many successful outcomes, like simulated robot control (Tassa et al., 2012; Varlamov & Aladin, 2024) and board games (Silver et al., 2016; 2017). Conventionally, planning relies on the complete model of the environment that provides the necessary information about action effects and outcome probabilities. In instances where such environment's dynamics are not available, model learning and function approximation techniques are usually employed (Deisenroth & Rasmussen, 2011; Levine & Abbeel, 2014; Hafner et al., 2019; Schrittwieser et al., 2020), like model-based reinforcement learning (Sutton & Barto, 2018).

Recent advances have focused on leveraging diffusion models (Ho et al., 2020), known for their ability to generate high-quality samples (Luo & Hu, 2021; Li et al., 2022; Nichol et al., 2022), to create diffusion-based planners (Janner et al., 2022; Ajay et al., 2023; Liang et al., 2023). These planners approach sequential decision optimization as a data-driven trajectory optimization problem and simultaneously predict all timesteps of a trajectory. This innovative paradigm achieves non-autoregressive plan generation and aligns model learning with planning, thereby avoiding the compounding rollout errors (Asadi et al., 2018) and adversarial plans (Talvitie, 2014; Ke et al., 2018). Impressively, diffusion planners have demonstrated remarkable abilities to handle challenges like long-horizon, sparse-reward, as well as offline control tasks (Fu et al., 2020).

Despite these advantages, diffusion planners are unable to guarantee the generation of reliable and feasible plans due to the inherent *stochasticity* of diffusion models. Unlike deterministic models that yield the same output for a given input, the outputs of diffusion models are probabilistic. This characteristic means that while diffusion models may often produce high-quality samples, there is always a risk of generating unreliable samples, which are referred to as "*artifacts*" in vision task (Bau et al., 2019; Shen et al., 2020).

Current diffusion planners directly base their decisions on trajectories produced by diffusion models (Janner et al., 2022; Ajay et al., 2023; Liang et al., 2023). However, these planners often presuppose the high quality of the generated trajectories, neglecting the stochastic risks inherent in the process. This neglect can result in decision-making reliant on unreliable trajectories, leading to potentially ineffective or harmful actions for the intended task. Consequently, the stability and reliability of these diffusion planners are com-

[1]Zhejiang University, China [2]Nanyang Technological University, Singapore [3]Skywork AI, Singapore [4]State Key Laboratory of Brain-Machine Intelligence, China. Correspondence to: Gang Pan <gpan@zju.edu.cn>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

promised, particularly in safety-critical applications (Lee et al., 2023). Moreover, to enhance the generation of effective trajectories, these planners necessitate extra denoising steps. Such steps, however, lead to considerable delays in the decision-making process and introduce a critical limitation for scenarios that demand real-time responses, such as autonomous vehicle navigation (Koh et al., 2020).

In this study, we explore a novel approach inspired by the "wisdom of the crowd" concept (Sagi & Rokach, 2018), which posits that the collective opinion of a group often exceeds the accuracy of an individual's judgment. Our proposal, the Trajectory Aggregation Tree (TAT), aims to enhance decision-making in diffusion planners by resisting stochastic risks. Contrasting with previous methods that rely solely on raw trajectory predictions, TAT integrates and analyzes collective data from both historical and current trajectories. TAT functions as a dynamic tree-like structure. Each trajectory is conceptualized as a branch, with individual states forming the nodes. This structure evolves as new trajectories are integrated, akin to a tree growing branches. The impact of each trajectory on the TAT's overall structure is variable, with more trajectories exerting a more substantial influence on specific nodes. This mechanism prioritizes vital data while filtering out less reliable inputs. Ultimately, TAT makes its decisions based on the most impactful nodes, thereby enhancing the robustness and reliability.

Our proposed TAT has several appealing advantages: **(1)** Our study provides a theoretical justification to prove TAT's effectiveness in mitigating the impact of artifacts via trajectory aggregation. In our analysis, TAT-reinforced diffusion planners always outperform the original ones. **(2)** TAT exhibits an appreciable tolerance margin for artifact generation. This feature allows diffusion planners to sacrifice the generation quality, such as reducing denoising steps, to speed up planning without a noticeable performance decline. **(3)** TAT functions as a training-free and ready-to-deploy solution for existing diffusion planners; there is no need for fine-tuning and re-training of the original model. **(4)** In our empirical evaluations, we develop TAT on various diffusion planners, subjecting them to a diverse set of decision-making tasks. Our results demonstrate its impressive ability to filter out stochastic artifacts, consistently improve the performance of diffusion planners in all tasks, and enable planning speeds more than threefold faster.

2. Related Work

Diffusion probabilistic models (Sohl-Dickstein et al., 2015; Ho et al., 2020) have proven their highly-expressive generative capabilities in various domains, like computer vision (Lugmayr et al., 2022; Brempong et al., 2022; Fang et al., 2024), natural language processing (Austin et al., 2021; Li et al., 2022), and temporal data modeling (Tashiro et al., 2021; Kong et al., 2021; Wang et al., 2024). In recent developments, diffusion models have found applications in reinforcement learning (RL), with a particular focus on offline settings. These applications involves the use of diffusion models as planners (Janner et al., 2022; Ajay et al., 2023), policies (Wang et al., 2023; Chen et al., 2023a), and for data augmentation (Chen et al., 2023b).

Regarding planners, one of the representative works is the Diffuser (Janner et al., 2022), which plans by iteratively denoising trajectories using a diffusion model. It combines trajectory optimization with modeling, making sampling from the model and planning with it nearly identical. The Diffuser has demonstrated the long-horizon planning and test-time flexibility in offline control tasks. Building upon this framework, recent works have proposed several improvements. For example, Decision Diffuser (Ajay et al., 2023) incorporates classifier-free guidance, therefore avoiding the need for a separate reward function. AdaptDiffuser (Liang et al., 2023) uses an evolutionary planning method to self-evolve the diffusion model and generate rich synthetic expert data for goal-conditioned tasks. Lee et al. (2023) introduced the restoration gap to enhance the quality of trajectories, which requires training an additional gap predictor and using its gradient to refine the sampling of the diffusion model. Additionally, diffusion planners have been applied to multi-task RL (He et al., 2023; Ni et al., 2023), hierarchical RL (Li et al., 2023; Chen et al., 2024), and multi-agent RL (Zhu et al., 2023). Despite these advances, the probabilistic denoising process of diffusion models leads to the stochastic risk of yielding impractical and unreliable plans. This unpredictability introduces considerable challenges to ensuring the stability and reliability of these planners.

3. Background

Problem Setting. Consider a discounted Markov decision process (MDP), defined by the tuple $(\mathcal{S}, \mathcal{A}, P, r, \gamma)$. Here, $\mathcal{S}$ and $\mathcal{A}$ are the state space and action space respectively. The transition probability function $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ determines the likelihood of moving from one state to another given a specific action. $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function. $\gamma \in [0, 1)$ is the discount factor. Within each given state $\mathbf{s}_t \in \mathcal{S}$, the agent acts with action $\mathbf{a}_t \in \mathcal{A}$. This action leads to a transition to a new state: $\mathbf{s}_{t+1} = P(\mathbf{s}_t, \mathbf{a}_t)$, and the agent receives a reward $r(\mathbf{s}_t, \mathbf{a}_t)$.

In the sequence modeling, the goal of trajectory optimization is to identify an optimal sequence of actions $\mathbf{a}^*_{0:T}$ that maximizes the expected return $\mathcal{J}(\boldsymbol{\tau})$ with respect to trajectory $\boldsymbol{\tau} = (\mathbf{s}_0, \mathbf{a}_0, \mathbf{s}_1, \mathbf{a}_1, \ldots, \mathbf{s}_T, \mathbf{a}_T)$:

$$\mathbf{a}^*_{0:T} = \arg\max_{\mathbf{a}_{0:T}} \mathcal{J}(\boldsymbol{\tau}) = \arg\max_{\mathbf{a}_{0:T}} \sum_{t=0}^{T} \gamma^t r(\mathbf{s}_t, \mathbf{a}_t), \quad (1)$$

where t denotes the planning step and T denotes the plan-

ning horizon.

Diffusion Model. Synthesizing high-quality plans and behaviors entails learning a model distribution $p_\theta(\boldsymbol{\tau})$ that effectively approximates the ground-truth distribution of the trajectories $q(\boldsymbol{\tau})$. To this end, diffusion probabilistic models (Ho et al., 2020) typically factorize the data distribution $p_\theta(\boldsymbol{\tau})$ as a Markov chain of Gaussian transitions:

$$p_\theta(\boldsymbol{\tau}_0) = \int p_\theta(\boldsymbol{\tau}_K) \prod_k^K p_\theta(\boldsymbol{\tau}_{k-1}|\boldsymbol{\tau}_k) d\boldsymbol{\tau}_{1:K}, \quad (2)$$

where $\boldsymbol{\tau}_0$ denotes the original (noiseless) data, $\boldsymbol{\tau}_1, \ldots, \boldsymbol{\tau}_K$ denote the latent variables with the same dimensionality of $\boldsymbol{\tau}_0$, $p_\theta(\boldsymbol{\tau}_K) = \mathcal{N}(\boldsymbol{\tau}_K; \mathbf{0}, \mathbf{I})$ is the Gaussian prior, and $p_\theta(\boldsymbol{\tau}_{k-1}|\boldsymbol{\tau}_k)$ is the trainable reverse denoising process defined by

$$p_\theta(\boldsymbol{\tau}_{k-1}|\boldsymbol{\tau}_k) = \mathcal{N}(\boldsymbol{\tau}_{k-1}; \mu_\theta(\boldsymbol{\tau}_k, k), \Sigma_\theta(\boldsymbol{\tau}_k, k)). \quad (3)$$

Diffusion model predefines a forward noising process $q(\boldsymbol{\tau}_k|\boldsymbol{\tau}_{k-1}) = \mathcal{N}(\boldsymbol{\tau}_k; \sqrt{1-\beta_k}\boldsymbol{\tau}_{k-1}, \beta_k \mathbf{I})$, which progressively adding Gaussian noise to the data according to the variance schedule β_k. The trajectory generation involves an iterative denoising procedure $p_\theta(\boldsymbol{\tau}_{k-1}|\boldsymbol{\tau}_k)$, starting from a Gaussian noise $\boldsymbol{\tau}_K$.

However, a tractable variational lower-bound exists on $\mathbb{E}_{q(\boldsymbol{\tau}_0)}[\log p_\theta(\boldsymbol{\tau}_0)]$. A better scheme arises from fixing $\Sigma_\theta(\boldsymbol{\tau}_k, k)$ and optimizing a surrogate loss (Ho et al., 2020):

$$\mathcal{L}^{\text{simple}}(\theta) = \mathbb{E}_{k\sim\mathcal{U}, \boldsymbol{\tau}_0\sim q, \epsilon\sim\mathcal{N}(\mathbf{0},\mathbf{I})} \left[||\epsilon - \epsilon_\theta(\boldsymbol{\tau}_k, k)||^2 \right], \quad (4)$$

where $\mathcal{U}$ is a uniform distribution between 1 and K. The nosie-predictor model $\epsilon_\theta(\boldsymbol{\tau}_k, k)$ aims to estimates the added noise, from which $\mu_\theta(\boldsymbol{\tau}_k, k)$ can be readily derived.

To sample trajectories meeting certain constraints, it is necessary to formulate the problem as a conditional distribution $p_\theta(\boldsymbol{\tau}|y(\boldsymbol{\tau}))$, where $y(\boldsymbol{\tau})$ is some specific condition on trajectory sample $\boldsymbol{\tau}$. There are two common choices to do so: classifier guidance (Dhariwal & Nichol, 2021) and classifier-free guidance (Ho & Salimans, 2021). See Appendix A.

Planning with Diffusion. Given a well-trained diffusion model, planning is based on synthesizing a trajectory $\boldsymbol{\tau}_0$ through an iterative denoising process that progresses from $\boldsymbol{\tau}_K$ to $\boldsymbol{\tau}_0$. The trajectory is mathematically represented as:

$$\boldsymbol{\tau}_k = \{\boldsymbol{x}_0, \boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_T\}_k, \quad (5)$$

where k indicates the denoising sample step. The trajectory element, $\boldsymbol{x}_t$, manifests in two distinct forms, indicative of differing decision-making strategies: (1) The first is state-centric form, with $\boldsymbol{x}_t$ being solely the state, denoted as $\boldsymbol{x}_t = \mathbf{s}_t$ (Ajay et al., 2023). In this format, action generation is computed using an inverse dynamics model (Agrawal et al., 2016; Pathak et al., 2018), based on consecutive states: $\mathbf{a}_t = f_\phi(\mathbf{s}_t, \mathbf{s}_{t+1})$. (2) The second is state-action form, integrating state and preceding action: $\boldsymbol{x}_t = (\mathbf{s}_t, \mathbf{a}_{t-1})$ (Janner et al., 2022; Liang et al., 2023), where $\mathbf{a}_{-1}$ in $\boldsymbol{x}_0$ indicates an absence of action. This form enables direct extraction of actions for decision-making. More details of diffusion planners are provided in Appendix B.

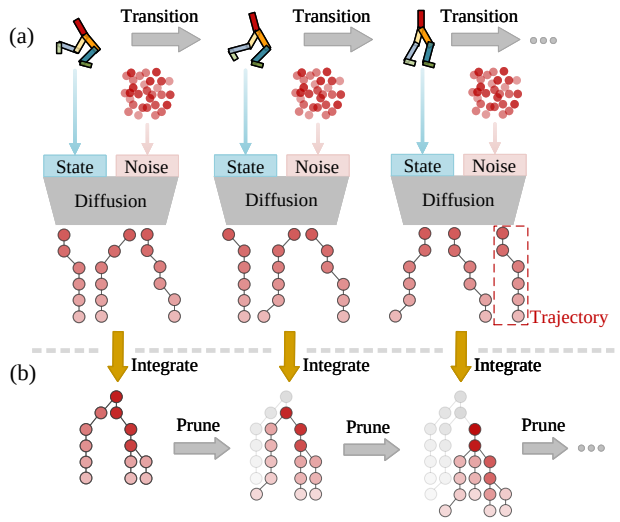

Figure 1. Overview of TAT. (a) Diffusion pipeline. Given the environment state, TAT utilizes the original diffusion planner to sample trajectories. (b) TAT pipeline. The tree incorporates past and current trajectories to construct a comprehensive experience of future states. The darker the red color, the higher the weight the node has. The tree grows dynamically and keeps in sync with the environment by pruning branches (the light transparent gray part).

In these planning strategies, the quality of trajectory $\boldsymbol{\tau}$ (short for $\boldsymbol{\tau}_0$) is essential: trajectories of low quality can greatly disrupt action generation and decision-making. Unfortunately, the stochasticity of diffusions often leads to the generation of unreliable trajectories, exacerbating the severity of this issue within prior planning strategies. Additionally, to adapt to changing environments, the planner strategically creates a new trajectory for planning at each step, leaving the old one behind, even though it may contain valuable information for future decisions.

4. Trajectory Aggregation Tree

In response to the stochastic risks in diffusion planners, we introduce the Trajectory Aggregation Tree (TAT) method. TAT innovatively utilizes data from both current and previous steps, akin to ensemble learning techniques that harness collective wisdom to mitigate individual uncertainties. Figure 1 provides an overview of TAT and illustrates its key aspects. It conceptualizes each trajectory $\boldsymbol{\tau} = \{\boldsymbol{x}_0, \ldots, \boldsymbol{x}_T\}$

as a branch and introduces a tree structure to **merge** trajectories with similar states, while **branching out** for divergent trajectories. This procedure is **executed efficiently**, as TAT operates without necessitating new models, extra trainable parameters, or additional generation steps. In Figure 1, we utilize the original pipeline of diffusion planners to generate trajectories, treating TAT as a downstream process. Moreover, TAT grows **dynamically** as the diffusion cycle repeats and keeps up-to-date with the environment, facilitating real-time planning. With each timestep, it seamlessly incorporates new trajectories from the diffusion pipeline, enriching its structure with additional data. TAT evaluates the **cumulative effect** of all trajectories on each node by assigning weights. This strategy emphasizes fundamental nodes assigned with the highest cumulative weights while pruning substandard individual nodes, effectively mitigating the impact of stochastic artifacts.

In the following sections, we will delve into the node structure of TAT, outline the process of constructing TAT and its application in planning, and provide a theoretical analysis of TAT's effectiveness in mitigating risks associated with unreliable data generation.

4.1. Node Structure

We next establish the node structure of the TAT. Our objective is to enable the aggregation of analogous information across multiple trajectories while weighting decision-related data. To this end, we define each node within the tree by the expression $e_{t,i}$. Each node is represented by the tuple:

$$e_{t,i} = \big\{X(e_{t,i}), V(e_{t,i}), \boldsymbol{x}(e_{t,i})\big\}, \tag{6}$$

where t indicates the node's depth, corresponding to the environmental step, and $i = 0, 1, \dots$ indexes the nodes under the same parent. The edges between nodes symbolize temporal relationships. Node $e_{t,i}$ comprises a set of statistics:

- $X(e_{t,i})$ contains a set of **similar trajectory elements** $\boldsymbol{x}_t$ from multiple trajectories. $\boldsymbol{x}_t$ is representative of either state $\mathbf{s}_t$ or state-action pairs $(\mathbf{s}_t, \mathbf{a}_{t-1})$, based on the planning strategies outlined in Section 3.
- $V(e_{t,i})$ contains the **weights** assigned to node $e_{t,i}$ from these trajectory elements, reflecting its visitation frequency and significance.
- $\boldsymbol{x}(e_{t,i})$, the **node state**, distills the joint feature from the elements within $X(e_{t,i})$, serving as a canonical representation of the node's aggregated data.

Further elaboration on these statistics will be provided in the subsequent sections, detailing their operational roles.

4.2. Planning with TAT

This section describes how to construct the TAT and plan with it, incorporating details from Figure 2.

Initialization. Initially, we create the foundation for our TAT, symbolized as Υ. This step involves establishing an empty root node $e_{0,0}$, where $X(e_{0,0})$ starts empty, indicating no data is yet collected. $V(e_{0,0})$ is also empty showing no weights assigned. $\boldsymbol{x}(e_{0,0})$ is set to $\mathbf{0}$, representing the initial node state. Next, TAT evolves synchronized with environmental steps. If new trajectories emerge, they are integrated into the existing Υ. Integration of a trajectory $\boldsymbol{\tau} = \{\boldsymbol{x}_0, \dots, \boldsymbol{x}_T\}$ involves two key steps: (1) **Merging** similar branch to consolidate common patterns and (2) **Expanding** different branch to accommodate new information.

Merging. This process aims to merge the trajectory elements $\boldsymbol{x}_t$ into the nodes $e_{i,t}$ with similar state characteristics, following a path beginning at the root node $e_{0,0}$. The initial state $\boldsymbol{x}_0$ is merged directly into $e_{0,0}$ since it has no impact on decision-making. At each subsequent time step t, we evaluate the child nodes $e_{t,i}$ of the last merged node at depth $t-1$, seeking a node whose state closely aligns with $\boldsymbol{x}_t$. For discrete states, this involves checking for exact matches, while for continuous states, cosine similarity, $\mathfrak{S}^{\cos}(\boldsymbol{x}_t, e_{t,i}) = \frac{\boldsymbol{x}_t \cdot \boldsymbol{x}(e_{t,i})}{\|\boldsymbol{x}_t\| \cdot \|\boldsymbol{x}(e_{t,i})\|}$, is calculated to identify the most suitable node. A threshold α, marginally less than one[1], is employed to guarantee adequate similarity. The node with the highest similarity while satisfying $\mathfrak{S}^{\cos} > \alpha$ is selected for merging. Then, the visited node receives an additional weight and updates its statistics:

$$X(e_{t,i}).\texttt{insert}(\boldsymbol{x}_t), V(e_{t,i}).\texttt{insert}(\lambda^t), \tag{7}$$

$$\boldsymbol{x}(e_{t,i}) = \frac{\sum_{\{\boldsymbol{x}_i, \lambda^i\} \in \{X(e_{t,i}), V(e_{t,i})\}} \boldsymbol{x}_i \cdot \lambda^i}{\sum_{\lambda^i \in V(e_{t,i})} \lambda^i}, \tag{8}$$

where Equation (7) represents the inclusion of the new state $\boldsymbol{x}_t$ and its weight λ^t, thus enhancing the node's significance, with λ^t denoting the weight assigned to each node by a decay factor λ (where $\lambda \leq 1$), prioritizing recent states for predictive accuracy, as shown in Figure 2 (a). Equation (8) updates the node state $\boldsymbol{x}(e_{t,i})$ to the weighted average of its states, ensuring dynamic adaptation to new data.

Expanding. The merging procedure is terminated (presumably at step L) when there are no suitable nodes for transition, e.g., when $\mathfrak{S}^{\cos}(\boldsymbol{x}_t, e_{t,i}) \leq \alpha, \forall i \in \{0, 1, \dots\}$. At this point, the latter segment of the trajectory, previously unexplored by the tree, remains unmerged. As such, the entire trajectory $\boldsymbol{\tau}$ is naturally divided into two parts: the former $\boldsymbol{\tau}^{\text{former}} = \{\boldsymbol{x}_0, \dots, \boldsymbol{x}_{L-1}\}$ and the latter $\boldsymbol{\tau}^{\text{latter}} = \{\boldsymbol{x}_L, \dots, \boldsymbol{x}_T\}$. The former sub-trajectory $\boldsymbol{\tau}^{\text{former}}$

[1]The value of cosine similarity is [-1,1], and the more similar the two states are, the closer the cosine similarity is to 1.

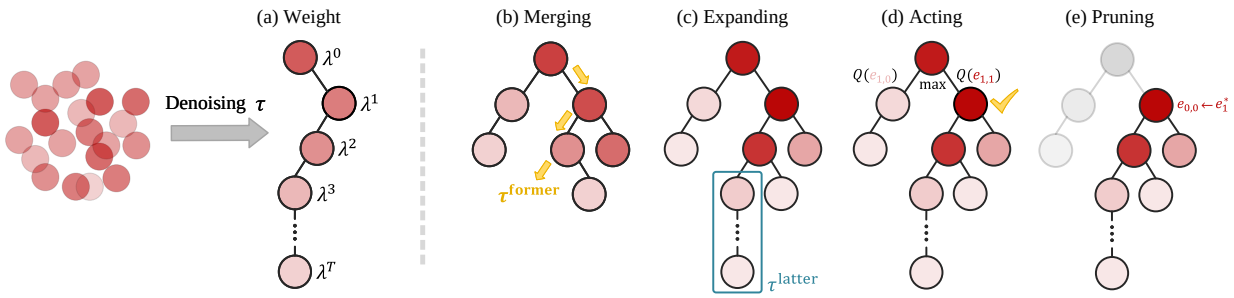

Figure 2. Planning with TAT. The darker the red color, the higher the weight the node has. (a) Weight allocation to each state in the sampled trajectory $\boldsymbol{\tau}$. (b) Merging procedure starting from the root node and sequentially traversing the states in $\boldsymbol{\tau}$ to assign weights to the tree nodes. (c) Expanding the tree with the sub-trajectory $\boldsymbol{\tau}^{\text{latter}}$ that the tree has not yet visited. (d) Acting by selecting the node with the highest weight among the child nodes for decision-making. (e) Pruning of the tree to synchronize with the environment.

is the part that can be traced within the tree. Conversely, the latter sub-trajectory $\boldsymbol{\tau}^{\text{latter}}$ represents the part that the tree has not yet visited before. Consequently, we undertake an expansion for the tree with the latter sub-trajectory $\boldsymbol{\tau}^{\text{latter}}$, as demonstrated in Figure 2 (c). We initialize a node $\{X(e_{t,i}) = \{\boldsymbol{x}_t\}, V(e_{t,i}) = \{\lambda^t\}, \boldsymbol{x}(e_{t,i}) = \boldsymbol{x}_t\}$ for each state $\boldsymbol{x}_t \in \boldsymbol{\tau}^{\text{latter}}$, where $t = L, \dots, T$. Then, we attach the latter state's node as a child node of the previous state's node, sequentially expanding the tree.

Acting. In the current state (at the root node $e_{0,0}$), the next state needs to be chosen for decision-making. TAT makes its decisions based on the most impactful node, which has the highest weight among the child nodes $\{e_{1,i}\}_{i=0,1,\dots}$:

$$e_1^* = \underset{e \in \{e_{1,i}\}_{i=0,1,\dots}}{\arg\max} \quad Q(e), \tag{9}$$

where $Q(e) = \sum_{\lambda^i \in V(e)} \lambda^i$ is the accumulated weight of node e. We then take its represented state $\boldsymbol{x}_1^* = \boldsymbol{x}(e_1^*)$ as our target next state. To decide the action $\mathbf{a}_0^*$ for the current environment state $\mathbf{s}_0$, we adopt the original strategies from prior methods. If it is the state-action pair, like Diffuser (Janner et al., 2022), the action is directly derived from $\boldsymbol{x}_1^* = (\mathbf{s}_1^*, \mathbf{a}_0^*)$; If it is the state-centric form ($\boldsymbol{x}_1^* = \mathbf{s}_1^*$), like Decision Diffuser (Ajay et al., 2023), the action is derived via an inverse dynamics model $\mathbf{a}_0^* = f_\phi(\mathbf{s}_0, \mathbf{s}_1^*)$.

Pruning. After making the decision, the environment transitions to the next state, and the tree is pruned to reflect this transition. The subtree rooted at the selected node e_1^* becomes the new tree: $e_{0,0} \leftarrow e_1^*$. As such, the tree is dynamically updated over time with the progression of the decision-making process, ensuring that the model's predictions are in sync with the environment state transitions.

Pseudocode of closed-loop planning with TAT in a single episode is given in Algorithm 1, where lines 11-18 correspond to the merging, lines 20-23 correspond to the expanding, line 25 corresponds to the acting, and line 27 corresponds to the pruning. Open-loop planning can be easily implemented by sampling trajectories (lines 5-9) just once within the initial while loop.

Algorithm 1 Planning with TAT

1: **Require:** Pretrained diffusion model g_θ
2: Initialize an empty tree $\Upsilon \leftarrow e_{0,0}$
3: **for** each step of the environment **do**
4: // Sampling plans via diffusion
5: Observe current state $\mathbf{s}$; Initialize plan $\boldsymbol{\tau}_K \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
6: **for** $k = K, \dots, 1$ **do**
7: Set $\mathbf{s}$ as the first state of $\boldsymbol{\tau}_k$
8: Perform one-step denoising: $\boldsymbol{\tau}_{k-1} \leftarrow g_\theta(\boldsymbol{\tau}_k, k)$
9: **end for**
10: // Merging
11: **for** $t = 1, \dots$ **do**
12: Compute cosine similarities $\{\mathfrak{S}^{\cos}(\boldsymbol{x}_t, e_{t,i})\}_{i=1=0,1,\dots}$
13: **if** $\forall i \in \{0, 1, \dots\}, \mathfrak{S}^{\cos}(\boldsymbol{x}_t, e_{t,i}) \leq \alpha$ **then**
14: Break the loop (presumably at $t = L$)
15: **end if**
16: Select the node with the highest $\mathfrak{S}^{\cos}$
17: Merge $\boldsymbol{x}_t$ into the selected node by Equation (7), (8)
18: **end for**
19: // Expanding
20: **for** $t = L, \dots, T$ **do**
21: Initialize a node with state $\boldsymbol{x}_t$
22: Expand Υ by attaching this node as a new leaf node
23: **end for**
24: // Acting
25: Execute action $\mathbf{a}_0$ drawn from e_1^* in Equation (9)
26: // Pruning
27: Transition to the next state and prune the tree $e_{0,0} \leftarrow e_1^*$
28: **end for**

4.3. Theoretical Analysis

In this section, we provide a theoretical analysis of TAT's effectiveness in the presence of artifacts. All the proofs pertinent to this section are included in Appendix C. We first give some key definitions.

Definition 4.1 (Artifact Probability)**.** We let ε denote the

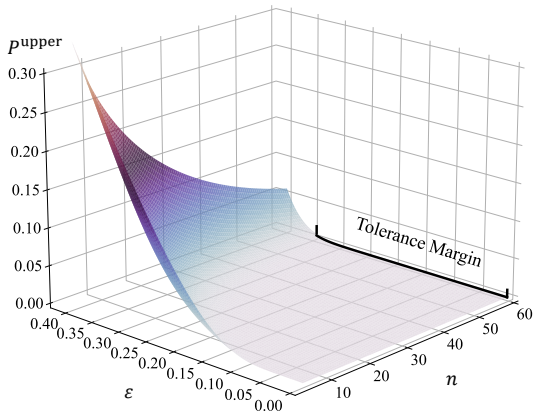

Figure 3. The dynamic trend of the upper bound P^{upper} of $P^{\text{artifact}}(n;\Upsilon)$ with respect to ε and n. As n increases, it monotonically decreases and becomes less sensitive to the variation of ε.

probability of each x_t being an artifact.

Definition 4.2 (Number of Trajectories)**.** We let n denote the total number of trajectories involved across all candidate child nodes $\{e_{1,i}\}_{i=0,1,\dots}$ in Equation (9), i.e., $n = \sum_{i=0,1,\dots} |X(e_{1,i})|$. Hence, n is a positive integer greater than 1, i.e., $n \in \mathbb{Z}^+_{>1}$.

We assume independence in the generation of artifacts across different trajectories, which generally holds as trajectories are derived from i.i.d. Gaussian noise (Sohl-Dickstein et al., 2015). We also assume that the diffusion planner has practical significance, leaning towards producing reliable states rather than unreliable ones, i.e., $\varepsilon < 0.5$. We proceed to introduce a proposition that involves the bounded probability of TAT choosing an artifact for decision-making.

Proposition 4.3. *In TAT's planning, each action is a product of aggregated information from n trajectories. Consider the case of $\lambda = 1$, the probability of TAT choosing the artifact can be bounded by*

$$P^{\text{artifact}}(n;\Upsilon) < \frac{1}{2}\left[1 - \text{erf}\left(\frac{n/2 - n\varepsilon}{\sqrt{2n\varepsilon(1-\varepsilon)}}\right)\right],$$

where $\text{erf}(\cdot)$ *denotes the error function.*

Proposition 4.3 implies that as the number of trajectories n increases, the term inside the error function grows, which in turn leads to a monotonically decreasing upper bound of $P^{\text{artifact}}(n;\Upsilon)$. Hence, involving more trajectories in the system becomes a feasible strategy to enhance the performance. Besides, we can further derive the following corollary.

Corollary 4.4. *For $\forall n \in \mathbb{Z}^+_{>1}$, the probability of TAT selecting an artifact satisfies $P^{\text{artifact}}(n;\Upsilon) < \varepsilon$ and $\lim_{n\to\infty} P^{\text{artifact}}(n;\Upsilon) = 0$.*

Corollary 4.4 highlights two important properties of $P^{\text{artifact}}(n;\Upsilon)$. Firstly, TAT's probability of selecting an artifact is always lower than that of an individual trajectory, ensuring performance improvement. Secondly, with a sufficiently large number of trajectories, the likelihood approaches zero. In Figure 3, we provide a visual illustration of the dynamic trend of the upper bound to support our theoretical claims. Moreover, we observe that when n is large (e.g., $n = 60$), the upper bound is not sensitive to variations in ε within a wide interval (e.g., $\varepsilon \in [0.00, 0.35]$). This observation indicates that TAT manifests a tolerance margin for the generation of artifacts, which allows diffusion planners to achieve faster yet reliable planning by reducing sampling steps.

5. Experiments

In this section, we present the empirical evaluations of the proposed TAT across a range of decision-making tasks in offline control settings. As our TAT serves as plug-and-play and training-free enhancement, we use the open-source pre-trained models of existing diffusion planners, deploying TAT directly. In cases where pre-trained models are not publicly available, we retrain them using the original hyperparameters. For more detailed information, please refer to Appendix D. Our experiments are designed to demonstrate **(1)** the ability of TAT to filter out potential artifacts and improve the reliability of the planner; **(2)** its ability to boost the performance of diffusion planners as a plug-and-play enhancement; **(3)** its ability to achieve faster yet reliable planning benefiting from its tolerance for artifact generation. Source code is available at `https://github.com/langfengQ/tree-diffusion-planner`.

5.1. Risk Resistance

In this part, we deploy our TAT (Υ) on the pre-trained Diffuser (Janner et al., 2022) and denote it as Diffuser$^{\Upsilon}$. We evaluate TAT on the Maze2D environments (Fu et al., 2020) to show its effectiveness in minimizing the artifacts' risks in the original Diffuser. Maze2D presents a navigation task, challenging the agent to plan a path to reach a designated goal location, where only a reward of 1 is granted. The sampling procedure of Diffuser in Maze2D tasks is conditioned on a start and goal location without return guidance.

As shown in Figure 4, Diffuser inevitably produces some infeasible transitions in both open-loop and closed-loop planning, such as instances of wall-crossing, with 5 out of 64 trajectories in Figure 4 (a). Therefore, it is risky and unreliable to plan with an individual trajectory in prior methods. In contrast, Diffuser$^{\Upsilon}$ aggregates multiple trajectories, dynamically expands and prunes the tree, and makes decisions based on the most impactful nodes. In Figure 4, Diffuser$^{\Upsilon}$ filters out all impractical transitions and exhibits a strong

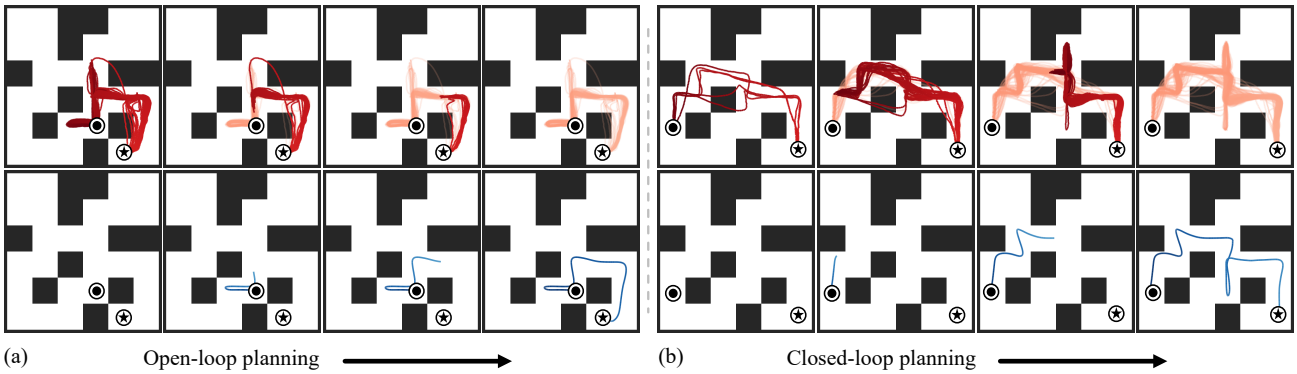

Figure 4. The planning process of Diffuser$^{\Upsilon}$ in open-loop (a) and closed-loop (b) manners. ⦿ denotes the starting position and ⍟ denotes the goal position. The red lines on the top row are the plans of the baseline Diffuser. It is evident that Diffuser generates artifacts at the beginning (a) or introduces new artifacts in subsequent steps (b). Diffuser$^{\Upsilon}$ can dynamically filter out and prune these inefficient branches indicated by the light transparent segments. The blue lines on the below row are the results of Diffuser$^{\Upsilon}$.

Table 1. The performance of Diffuser$^{\Upsilon}$, Diffuser, and varieties of prior approaches in the Maze2D environment. We report the mean and the standard error of Diffuser$^{\Upsilon}$ over 1000 planning seeds.

Environment	**MPPI**	**CQL**	**IQL**	**Diffuser**	**Diffuser$^{\Upsilon}$**
Maze2d U-Maze	33.2	5.7	47.4	**113.9**	**114.5**$\pm$1.2
Maze2d Medium	10.2	5.0	34.9	121.5	**130.7**$\pm$0.6
Maze2d Large	5.1	12.5	58.6	123.0	**133.4**$\pm$1.7
Single-task Average	16.2	7.7	47.0	119.5	**126.2**
Multi2d U-Maze	41.2	-	24.8	**128.9**	**129.4**$\pm$0.6
Multi2d Medium	15.4	-	12.1	127.2	**135.4**$\pm$0.9
Multi2d Large	8.0	-	13.9	132.1	**143.8**$\pm$1.5
Multi-task Average	21.5	-	16.9	129.4	**136.2**

ability to resist the stochastic risk from artifacts. As indicated by the blue paths, Diffuser$^{\Upsilon}$ offers a more reliable and artifact-free path towards the goal location.

5.2. Performance Boosting

We evaluate TAT's ability to boost the performance of existing diffusion planners in three different offline RL tasks.

Maze2D. We first evaluate our method on sparse-reward, long-horizon tasks in Maze2D (Fu et al., 2020). As we describe above, it is a particularly demanding navigation task as the agent often needs to take hundreds of steps without receiving any rewards until reaching the final goal. Successfully navigating the maze requires the agent to adeptly handle sparse rewards and exhibit proficiency in long-horizon planning. We also assess multi-task flexibility on Multi2D, a variant of Maze2D that introduces goal randomization for each episode (Janner et al., 2022). We compare our TAT-reinforced Diffuser$^{\Upsilon}$ with original Diffuser (Janner et al., 2022) and other baselines in Table 1.

Table 2. The performance of Diffuser$^{\Upsilon}$, Diffuser and various prior approaches on block stacking tasks. We report the mean and the standard error of Diffuser$^{\Upsilon}$ and DD$^{\Upsilon}$ over 100 planning seeds.

Environment	**BCQ**	**CQL**	**Diffuser**	**Diffuser$^{\Upsilon}$**	**DD**	**DD$^{\Upsilon}$**
Unconditional	0.0	24.4	54.3	**60.7**$\pm$2.3	56.0	**63.7**$\pm$2.6
Conditional	0.0	0.0	48.7	56.7$\pm$3.5	54.0	**62.3**$\pm$3.5
Rearrangement	0.0	0.0	51.3	63.3$\pm$3.0	59.8	**69.2**$\pm$3.1
Average	0.0	8.1	51.4	60.2	56.6	**65.1**

Notably, model-based trajectory optimizer MPPI (Williams et al., 2015) and model-free offline algorithms CQL (Kumar et al., 2020) and IQL (Kostrikov et al., 2022) all struggle to reach the goal. The state-of-the-art model-free IQL exhibits a significant performance drop in multi-task navigation due to the great difficulties in performing credit assignments. In contrast, Diffuser performs much better in both single-task and multi-task scenarios, but its performance is affected by artifacts shown in Figure 4. Our TAT effectively addresses this risk and boosts the performance of the Diffuser, particularly in larger and more complex mazes, where the likelihood of generating artifacts increases. We can observe an improvement of around 10 scores in such cases.

Kuka block stacking. The Kuka block stacking suite (Janner et al., 2022) is designed for evaluating algorithms' test-time flexibility. It involves the manipulation of a Kuka robotic arm (Schreiber et al., 2010) to accomplish three tasks: *unconditional stacking* to build the tallest possible block tower, *conditional stacking* to build a block tower in a specified order, and *rearrangement* to match the locations of reference blocks in a new arrangement. We deploy TAT on Diffuser (Janner et al., 2022) and Decision Diffuser

Table 3. The performance of various diffusion planners with/without TAT, and varieties of prior approaches on MuJoCo locomotion tasks. We report the normalized average returns and the standard errors of Diffuser$^{\Upsilon}$ and RGG$^{\Upsilon}$ over 50 planning seeds. We highlight scores that fall within 5 percent of the maximum per task ($\geq 0.95 \cdot \max$).

Dataset	Environment	BC	CQL	IQL	DT	TT	MOPO	MOReL	MBOP	Diffuser	Diffuser$^{\Upsilon}$	RGG	RGG$^{\Upsilon}$
Med-Expert	HalfCheetah	55.2	91.6	86.7	86.8	95.0	63.3	53.3	**105.9**	88.9	92.5$\pm$0.8	90.8	95.6$\pm$1.1
Med-Expert	Hopper	52.5	105.4	91.5	**107.6**	**110.0**	23.7	**108.7**	55.1	103.3	**109.4**$\pm$2.1	**109.6**	**111.2**$\pm$1.8
Med-Expert	Walker2d	**107.5**	**108.8**	**109.6**	**108.1**	101.9	44.6	95.6	70.2	**106.9**	**108.8**$\pm$0.3	**107.8**	**109.3**$\pm$0.2
Medium	HalfCheetah	42.6	44.0	**47.4**	42.6	**46.9**	42.3	42.1	44.6	42.8	44.3$\pm$0.3	44.0	**45.7**$\pm$0.4
Medium	Hopper	52.9	58.5	66.3	67.6	61.1	28.0	**95.4**	48.8	74.3	82.6$\pm$1.9	82.5	86.0$\pm$2.3
Medium	Walker2d	75.3	72.5	78.3	74.0	**79.0**	17.8	77.8	41.0	**79.6**	**81.0**$\pm$0.4	**81.7**	**82.7**$\pm$0.8
Med-Replay	HalfCheetah	36.6	45.5	44.2	36.6	41.9	**53.1**	40.2	42.3	37.7	39.2$\pm$1.5	41.0	43.6$\pm$1.6
Med-Replay	Hopper	18.1	**95.0**	**94.7**	82.7	91.5	67.5	93.6	12.4	93.6	**95.3**$\pm$0.3	**95.2**	**99.5**$\pm$0.5
Med-Replay	Walker2d	26.0	77.2	73.9	66.6	**82.6**	39.0	49.8	9.7	70.6	78.2$\pm$2.2	78.3	**82.3**$\pm$1.9
Average		51.9	77.6	77.0	74.7	78.9	42.1	72.9	47.8	77.5	**81.3**	**81.2**	**84.0**

(DD) (Ajay et al., 2023) and the sampling is guided by specific task conditions. The results are given in Table 2.

Compared to model-free offline RL algorithms BCQ (Fujimoto et al., 2019) and CQL (Kumar et al., 2020), diffusion planners exhibit an ability to adapt to different block properties and stacking criteria at test time. Notably, Diffuser$^{\Upsilon}$ and DD$^{\Upsilon}$ demonstrates a clear improvement margin ($> 15\%$ overall) over Diffuser and DD, with more significant enhancements in tasks that involve stacking constraints.

MuJoCo locomotion. At last, we evaluate our method on MuJoCo tasks using D4RL offline locomotion suite (Fu et al., 2020), which involves policy mixtures to assess algorithms' capability on heterogeneous data with varying quality. We choose pre-trained Diffuser (Janner et al., 2022) and Restoration Gap Guidance (RGG) (Lee et al., 2023) as baseline diffusion planners, and equip them with TAT to form Diffuser$^{\Upsilon}$ and RGG$^{\Upsilon}$, respectively. In these tasks, diffusion planners typically adopt closed-loop planning and are guided by high-return conditions. In addition, we also compared our method against various other baselines known for their strong performance in each domain of tasks, including imitation-based method behavior cloning (BC); model-free offline RL algorithms CQL (Kumar et al., 2020) and IQL (Kostrikov et al., 2022); sequence modeling method Decision Transformer (DT) (Chen et al., 2021); model-based RL methods Trajectory Transformer (TT) (Janner et al., 2021), MOPO (Yu et al., 2020), MOReL (Kidambi et al., 2020), MBOP (Argenson & Dulac-Arnold, 2021). The results are presented in Table 3.

It can be seen that TAT-reinforced planners demonstrate better performance compared to model-based approaches like MOReL and MBOP, as well as the sequence modeling approach DT. In addition, while the original Diffuser falls short in performance when compared to offline RL algorithms like CQL and TT, our TAT enables it to surpass these strong algorithms. Notably, TAT-reinforced planners (Diffuser$^{\Upsilon}$ and RGG$^{\Upsilon}$) consistently outperform their respective baseline planners in all tasks, without instances of performance drop. In some tasks, the performance gains can exceed 10%.

Table 4. The planning time and performance of Diffuser$^{\Upsilon}$ on Hopper Medium task with fewer sampling steps. The planning time of each action is averaged over 200 actions' generation and the score is averaged over 50 planning seeds. Originally set at 20 steps, the default sampling for Diffuser yields baseline results of 1.68s (time) and 74.3 (score). The symbols ✓ and ✗ are used to indicate whether Diffuser$^{\Upsilon}$ performs better or worse than corresponding baseline results, respectively.

Step	Warm-Start Sampling		Warm-Start$^{\Upsilon}$ Sampling	
	Time	Score	Time	Score
16	1.43s ✓	79.6$\pm$2.4 ✓	1.44s ✓	81.0$\pm$2.4 ✓
12	1.08s ✓	79.9$\pm$2.2 ✓	1.09s ✓	82.7$\pm$1.9 ✓
10	0.91s ✓	79.4$\pm$2.2 ✓	0.92s ✓	82.6$\pm$1.9 ✓
8	0.75s ✓	77.2$\pm$2.3 ✓	0.76s ✓	82.4$\pm$1.7 ✓
6	0.53s ✓	71.2$\pm$2.2 ✗	0.55s ✓	79.6$\pm$2.0 ✓
4	0.41s ✓	58.9$\pm$1.2 ✗	0.43s ✓	64.8$\pm$1.4 ✗

5.3. Faster Planning

In Proposition 4.3 and Figure 3, we demonstrated that TAT exhibits a tolerance margin for the generation of artifacts. Moreover, Figure 4 also illustrates TAT's ability to manage a growing proportion of infeasible trajectories, effectively filtering them out. This indicates that we can strategically sacrifice the sample quality by reducing denoising steps for faster planning. Such an aspect holds particular significance for diffusion planners, which are commonly subject to extensive time requirements from iterative denoising procedures.

To validate this, we trade sample quality for reduced denoising steps by warm-start planning (Janner et al., 2022). This strategy leverages previously generated plans to create partially noised trajectories, which are then used as starting

points for subsequent planning iterations to reduce denoising steps. We present the results of Diffuser$^{\Upsilon}$ in Table 4. We observed that Diffuser$^{\Upsilon}$ maintains consistent performance for sampling steps exceeding 8. Within this range, we can accelerate the planning of Diffuser$^{\Upsilon}$ while ensuring it outperforms the original Diffuser. Moreover, TAT can optimize this process by rollouting branches with the highest weight for warm-start planning, denoted as warm-start$^{\Upsilon}$ sampling. We find that this technique not only enhances the robustness of Diffuser$^{\Upsilon}$ to variations in sampling steps but also marginally improves performance. We present the results in Table 4. Remarkably, Diffuser$^{\Upsilon}$ achieves better results than vanilla warm-start planning. It consistently outperforms Diffuser when the sampling step is reduced to just 6, significantly decreasing planning latency to 0.55s (more than a threefold increase in speed). In Appendix F, we also present the time and memory budget of TAT, which we find to be quite modest.

6. Conclusion

In this work, we addressed the challenge of stochastic risk from infeasible trajectories in diffusion planners. We proposed a flexible and easy-to-deploy solution, TAT, that aggregates the collective wisdom of past and immediate trajectories. TAT presents a robust and practical structure to mitigate the impact of potential artifacts, and effectively exclude them from decision-making. Theoretical analysis and empirical results both provide support for the effectiveness of TAT. Remarkably, TAT consistently outperforms its baseline counterparts in all tasks and enables a remarkable increase in planning speed. Although our work focuses on the uncertainty risk of diffusion in decision-making domains, the concept of TAT could readily be extended to other areas, where similar stochasticity of generative models poses a challenge. Additionally, more sophisticated weight allocation and tree structures, such as incorporating spatial consideration, could be explored for further improvements. We leave it for future work.

Acknowledgements

This research is supported by Natural Science Foundation of China (No. 61925603, U1909202), STI 2030 Major Projects (2021ZD0200400), and the Key Research and Development Program of Zhejiang Province in China (2020C03004). This research is also supported by the National Research Foundation, Singapore under its Industry Alignment Fund-Prepositioning (IAF-PP) Funding Initiative. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of National Research Foundation, Singapore.

Impact Statement

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

References

Agrawal, P., Nair, A. V., Abbeel, P., Malik, J., and Levine, S. Learning to poke by poking: Experiential learning of intuitive physics. In *Advances in neural information processing systems*, volume 29, 2016.

Ajay, A., Du, Y., Gupta, A., Tenenbaum, J. B., Jaakkola, T. S., and Agrawal, P. Is conditional generative modeling all you need for decision making? In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=sP1fo2K9DFG.

Argenson, A. and Dulac-Arnold, G. Model-based offline planning. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=OMNB1G5xzd4.

Asadi, K., Misra, D., and Littman, M. Lipschitz continuity in model-based reinforcement learning. In *International Conference on Machine Learning*, pp. 264–273. PMLR, 2018.

Austin, J., Johnson, D. D., Ho, J., Tarlow, D., and Van Den Berg, R. Structured denoising diffusion models in discrete state-spaces. In *Advances in Neural Information Processing Systems*, volume 34, pp. 17981–17993, 2021.

Bau, D., Zhu, J.-Y., Strobelt, H., Zhou, B., Tenenbaum, J. B., Freeman, W. T., and Torralba, A. Gan dissection: Visualizing and understanding generative adversarial networks. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Hyg_X2C5FX.

Brempong, E. A., Kornblith, S., Chen, T., Parmar, N., Minderer, M., and Norouzi, M. Denoising pretraining for semantic segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 4175–4186, 2022.

Chen, C., Deng, F., Kawaguchi, K., Gulcehre, C., and Ahn, S. Simple hierarchical planning with diffusion. *arXiv preprint arXiv:2401.02644*, 2024.

Chen, H., Lu, C., Ying, C., Su, H., and Zhu, J. Offline reinforcement learning via high-fidelity generative behavior modeling. In *The Eleventh International Conference on Learning Representations*, 2023a. URL https://openreview.net/forum?id=42zs3qa2kpy.

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

Chen, Z., Kiami, S., Gupta, A., and Kumar, V. Genaug: Retargeting behaviors to unseen situations via generative augmentation. *arXiv preprint arXiv:2302.06671*, 2023b.

Choi, H., Chang, W., and Choi, J. Can we find neurons that cause unrealistic images in deep generative networks? In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence*, pp. 2888–2894, 2022.

Deisenroth, M. and Rasmussen, C. E. Pilco: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on machine learning (ICML-11)*, pp. 465–472, 2011.

Dhariwal, P. and Nichol, A. Diffusion models beat gans on image synthesis. In *Advances in neural information processing systems*, volume 34, pp. 8780–8794, 2021.

Fang, T., Zheng, Q., and Pan, G. Alleviating the semantic gap for generalized fmri-to-image reconstruction. *Advances in Neural Information Processing Systems*, 36, 2024.

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

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

Hafner, D., Lillicrap, T., Fischer, I., Villegas, R., Ha, D., Lee, H., and Davidson, J. Learning latent dynamics for planning from pixels. In *International conference on machine learning*, pp. 2555–2565. PMLR, 2019.

He, H., Bai, C., Xu, K., Yang, Z., Zhang, W., Wang, D., Zhao, B., and Li, X. Diffusion model is an effective planner and data synthesizer for multi-task reinforcement learning. *arXiv preprint arXiv:2305.18459*, 2023.

Ho, J. and Salimans, T. Classifier-free diffusion guidance. In *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*, 2021. URL https://openreview.net/forum?id=qw8AKxfYbI.

Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models. In *Advances in neural information processing systems*, volume 33, pp. 6840–6851, 2020.

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

Janner, M., Du, Y., Tenenbaum, J., and Levine, S. Planning with diffusion for flexible behavior synthesis. In *International Conference on Machine Learning*, pp. 9902–9915. PMLR, 2022.

Jeong, H., Han, J., and Choi, J. An unsupervised way to understand artifact generating internal units in generative neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 1052–1059, 2022.

Ke, N. R., Singh, A., Touati, A., Goyal, A., Bengio, Y., Parikh, D., and Batra, D. Modeling the long term future in model-based reinforcement learning. In *International Conference on Learning Representations*, 2018.

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

Koh, S., Zhou, B., Fang, H., Yang, P., Yang, Z., Yang, Q., Guan, L., and Ji, Z. Real-time deep reinforcement learning based vehicle navigation. *Applied Soft Computing*, 96:106694, 2020.

Kong, Z., Ping, W., Huang, J., Zhao, K., and Catanzaro, B. Diffwave: A versatile diffusion model for audio synthesis. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=a-xFK8Ymz5J.

Kostrikov, I., Nair, A., and Levine, S. Offline reinforcement learning with implicit q-learning. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=68n2s9ZJWF8.

Kumar, A., Zhou, A., Tucker, G., and Levine, S. Conservative q-learning for offline reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 33, pp. 1179–1191, 2020.

Lee, K., Kim, S., and Choi, J. Refining diffusion planner for reliable behavior synthesis by automatic detection of infeasible plans. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=Q5tuGgqJwt.

Levine, S. and Abbeel, P. Learning neural network policies with guided policy search under unknown dynamics. In *Advances in neural information processing systems*, volume 27, 2014.

Li, W., Wang, X., Jin, B., and Zha, H. Hierarchical diffusion for offline decision making. In *International Conference on Machine Learning*, pp. 20035–20064. PMLR, 2023.

Li, X., Thickstun, J., Gulrajani, I., Liang, P. S., and Hashimoto, T. B. Diffusion-lm improves controllable text generation. In *Advances in Neural Information Processing Systems*, volume 35, pp. 4328–4343, 2022.

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

Lugmayr, A., Danelljan, M., Romero, A., Yu, F., Timofte, R., and Van Gool, L. Repaint: Inpainting using denoising diffusion probabilistic models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11461–11471, 2022.

Luo, S. and Hu, W. Diffusion probabilistic models for 3d point cloud generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2837–2845, 2021.

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

Ni, F., Hao, J., Mu, Y., Yuan, Y., Zheng, Y., Wang, B., and Liang, Z. Metadiffuser: Diffusion model as conditional planner for offline meta-rl. In *International Conference on Machine Learning*, pp. 26087–26105. PMLR, 2023.

Nichol, A. Q., Dhariwal, P., Ramesh, A., Shyam, P., Mishkin, P., Mcgrew, B., Sutskever, I., and Chen, M. Glide: Towards photorealistic image generation and editing with text-guided diffusion models. In *International Conference on Machine Learning*, pp. 16784–16804. PMLR, 2022.

Pathak, D., Mahmoudieh, P., Luo, G., Agrawal, P., Chen, D., Shentu, Y., Shelhamer, E., Malik, J., Efros, A. A., and Darrell, T. Zero-shot visual imitation. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pp. 2050–2053, 2018.

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

Sagi, O. and Rokach, L. Ensemble learning: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 8(4):e1249, 2018.

Schreiber, G., Stemmer, A., and Bischoff, R. The fast research interface for the kuka lightweight robot. In *IEEE workshop on innovative robot control architectures for demanding (Research) applications how to modify and enhance commercial controllers (ICRA 2010)*, pp. 15–21. Citeseer, 2010.

Schrittwieser, J., Antonoglou, I., Hubert, T., Simonyan, K., Sifre, L., Schmitt, S., Guez, A., Lockhart, E., Hassabis, D., Graepel, T., et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839): 604–609, 2020.

Shen, Y., Gu, J., Tang, X., and Zhou, B. Interpreting the latent space of gans for semantic face editing. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 9243–9252, 2020.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354–359, 2017.

Sohl-Dickstein, J., Weiss, E., Maheswaranathan, N., and Ganguli, S. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pp. 2256–2265. PMLR, 2015.

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

Talvitie, E. Model regularization for stable sample rollouts. In *UAI*, pp. 780–789, 2014.

Tashiro, Y., Song, J., Song, Y., and Ermon, S. Csdi: Conditional score-based diffusion models for probabilistic time series imputation. In *Advances in Neural Information Processing Systems*, volume 34, pp. 24804–24816, 2021.

Tassa, Y., Erez, T., and Todorov, E. Synthesis and stabilization of complex behaviors through online trajectory optimization. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4906–4913. IEEE, 2012.

Tousi, A., Jeong, H., Han, J., Choi, H., and Choi, J. Automatic correction of internal units in generative neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7932–7940, 2021.

Varlamov, O. and Aladin, D. A new generation of rules-based approach: Mivar-based intelligent planning of robot actions (mipra) and brains for autonomous robots. *Machine Intelligence Research*, pp. 1–22, 2024.

Wang, Y., Zhao, S., Jiang, H., Li, S., Luo, B., Li, T., and Pan, G. Diffmdd: A diffusion-based deep learning framework for mdd diagnosis using eeg. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 2024.

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

Williams, G., Aldrich, A., and Theodorou, E. Model predictive path integral control using covariance variable importance sampling. *arXiv preprint arXiv:1509.01149*, 2015.

Wu, Y. and He, K. Group normalization. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 3–19, 2018.

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

Zhang, X., Karaman, S., and Chang, S.-F. Detecting and simulating artifacts in gan fake images. In *2019 IEEE International Workshop on Information Forensics and Security (WIFS)*, pp. 1–6, 2019.

Zhu, Z., Liu, M., Mao, L., Kang, B., Xu, M., Yu, Y., Ermon, S., and Zhang, W. Madiff: Offline multi-agent learning with diffusion models. *arXiv preprint arXiv:2305.17330*, 2023.

A. Guided Diffusion

To sample trajectories that satisfy certain constraints, it is necessary to formulate the problem as a conditional distribution $p_\theta(\boldsymbol{\tau}|y(\boldsymbol{\tau}))$ using guided diffusion, where $y(\boldsymbol{\tau})$ is some specific condition on trajectory sample $\boldsymbol{\tau}$, e.g., the high-return $\mathcal{J}(\boldsymbol{\tau})$. There are two common choices to do so: classifier guidance (Dhariwal & Nichol, 2021) and classifier-free guidance (Ho & Salimans, 2021). The former combines the score estimate of a diffusion model with the gradient of $y(\boldsymbol{\tau})$ and thereby requires training a separate model y_ψ to predict the scores of trajectory samples. The latter replaces the dedicated separate model with a diffusion model trained by randomly dropping the condition during training. The details of classifier guidance and classifier-free guidance for diffusion planners are as follows:

A.1. Classifier Guidance

Classifier guidance (Dhariwal & Nichol, 2021) modifies the denoising process by combining the score estimate of a diffusion model with the gradients of guidance functions. Therefore, it requires training auxiliary guidance functions y_ψ, which are trained separately. For instance, considering cumulative rewards $y_\psi = \mathcal{J}_\psi$, classifier guidance modifies the denoising process as follows:

$$p_\theta(\boldsymbol{\tau}_{k-1}|\boldsymbol{\tau}_k, y(\boldsymbol{\tau})) = \mathcal{N}(\boldsymbol{\tau}_{k-1}; \mu_\theta(\boldsymbol{\tau}_k, k) + \alpha\Sigma g, \Sigma), \tag{A.1}$$

where $g = \nabla\mathcal{J}(\boldsymbol{\tau})|_{\boldsymbol{\tau}=\mu_\theta}$ and α is a guidance coefficient that controls the strength of the guidance. In this case, an additional model $\mathcal{J}_\psi$ is needed to be trained to predict the cumulative rewards of trajectory samples.

A.2. Classifier-Free Guidance

Classifier-free guidance (Ho & Salimans, 2021) eliminates the need for separate guidance functions, with the diffusion model trained by randomly dropping the condition during training. Classifier-free guidance modifies the denoising process as follows:

$$p_\theta(\boldsymbol{\tau}_{k-1}|\boldsymbol{\tau}_k, y(\boldsymbol{\tau})) = \mathcal{N}\left(\boldsymbol{\tau}_{k-1}; \mu_\theta(\boldsymbol{\tau}_k, k) + \omega(\mu_\theta(\boldsymbol{\tau}_k, k, y(\boldsymbol{\tau})) - \mu_\theta(\boldsymbol{\tau}_k, k)), \Sigma\right), \tag{A.2}$$

where ω is referred to as the guidance scale. $\mu_\theta(\boldsymbol{\tau}_k, k, y(\boldsymbol{\tau}))$ is the mean of the conditional diffusion model, and $\mu_\theta(\boldsymbol{\tau}_k, k)$ is the mean of the unconditional diffusion model. This method can be further simplified by applying it at the noise level:

$$\hat{\epsilon}_\theta(\boldsymbol{\tau}_k, k, y(\boldsymbol{\tau})) = \epsilon_\theta(\boldsymbol{\tau}_k, k) + \omega\left(\epsilon_\theta(\boldsymbol{\tau}_k, k, y(\boldsymbol{\tau})) - \epsilon_\theta(\boldsymbol{\tau}_k, k)\right), \tag{A.3}$$

where $\hat{\epsilon}_\theta(\boldsymbol{\tau}_k, k)$ is the perturbed noise, which will be used to later generate samples. $\epsilon_\theta(\boldsymbol{\tau}_k, k)$ and $\epsilon_\theta(\boldsymbol{\tau}_k, k, y(\boldsymbol{\tau}))$ are the unconditional and conditional noise models respectively.

B. Details of Diffusion Planners

In this section, we detail the planning process of diffusion planners. Generally, diffusion planning can be conducted in two ways: closed-loop planning and open-loop planning.

Closed-loop planning. The planner generates a new plan $\boldsymbol{\tau}_0$ with a horizon of T at each step of the environment. The plan is only used for decision-making at the current step. The planner can adjust the generated trajectory at each step based on environmental feedback to better adapt to the dynamic changes of the environment. As an illustrative example, we present the workflow of closed-loop planning implemented in Diffuser (Janner et al., 2022) in Algorithm B.1.

Algorithm B.1 Closed-loop planning with Diffuser

Input: Noise model ϵ_θ, guide $\mathcal{J}$
for each step of the environment **do**
  Observe current state $\mathbf{s}$
  Initialize plan $\boldsymbol{\tau}_K \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
  for $k = K \dots 1$ **do**
    Constrain first state of plan: $\boldsymbol{\tau}_k \leftarrow \mathbf{s}$
    Compute the parameters of reverse transition: $\epsilon \leftarrow \epsilon_\theta(\boldsymbol{\tau}_k, k)$, $\mu_{k-1} \leftarrow \texttt{Denoise}(\boldsymbol{\tau}_k, \epsilon)$
    Guide using gradients of return: $g = \nabla \mathcal{J}(\boldsymbol{\tau})|_{\boldsymbol{\tau}=\mu_{k-1}}$
    Sample: $\boldsymbol{\tau}_{k-1} \sim \mathcal{N}(\mu_{k-1} + \alpha \Sigma g, \Sigma)$
  end for
  Execute the first action of plan $\boldsymbol{\tau}_0$
end for

Open-loop planning. The planner generates a trajectory (plan) $\boldsymbol{\tau}_0$ with a horizon of T only at the beginning of the game. The plan is fixed and used throughout the entire episode, regardless of subsequent changes in the environment. As an illustrative example, we present the workflow of open-loop planning implemented in Diffuser (Janner et al., 2022) in Algorithm B.2.

Algorithm B.2 Open-loop planning with Diffuser

Input: Noise model ϵ_θ, guide $\mathcal{J}$
Observe the initial state $\mathbf{s}_0$
Initialize plan $\boldsymbol{\tau}_K \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
for $k = K \dots 1$ **do**
  Constrain first state of plan: $\boldsymbol{\tau}_k \leftarrow \mathbf{s}_0$
  Compute the parameters of reverse transition: $\epsilon \leftarrow \epsilon_\theta(\boldsymbol{\tau}_k, k)$, $\mu_{k-1} \leftarrow \texttt{Denoise}(\boldsymbol{\tau}_k, \epsilon)$
  Guide using gradients of return: $g = \nabla \mathcal{J}(\boldsymbol{\tau})|_{\boldsymbol{\tau}=\mu_{k-1}}$
  Sample: $\boldsymbol{\tau}_{k-1} \sim \mathcal{N}(\mu_{k-1} + \alpha \Sigma g, \Sigma)$
end for
$t = 0$
for each step of the environment **do**
  Execute t-th action of plan $\boldsymbol{\tau}_0$
  $t = t + 1$
end for

C. Proofs

Proposition 4.3. *In TAT's planning, each action is a product of aggregated information from n trajectories. Consider the case of $\lambda = 1$, the probability of TAT choosing the artifact can be bounded by*

$$P^{\text{artifact}}(n;\Upsilon) < \frac{1}{2}\left[1 - \text{erf}\left(\frac{n/2 - n\varepsilon}{\sqrt{2n\varepsilon(1-\varepsilon)}}\right)\right],$$

where $\text{erf}(\cdot)$ *denotes the error function.*

Proof. Given that the probability of artifact is ε, we apply the Binomial Distribution to denote the probability of having m artifacts within n states:

$$P(m;n,\varepsilon) = \binom{n}{m}\varepsilon^m(1-\varepsilon)^{n-m}, \tag{C.1}$$

To establish an upper bound, we consider the worst-case scenario where all artifacts are the same. This means when m is greater than or equal to $\lceil n/2 \rceil$, the tree will choose the artifact. Thus, we have the probability of choosing the artifact as:

$$P^{\text{artifact}}(n;\Upsilon) \leq \sum_{m=\lceil n/2 \rceil}^{n} \binom{n}{m}\varepsilon^m(1-\varepsilon)^{n-m} \tag{C.2}$$

Next, we can use the central limit theorem to approximate this sum. Considering that n is sufficiently large, the binomial distribution can be approximated by a normal distribution $\mathcal{N}(\mu,\sigma^2)$, where $\mu = n\varepsilon$ and $\sigma^2 = n\varepsilon(1-\varepsilon)$. In this case, we can express the sum as

$$\int_{m=\lceil n/2 \rceil}^{n} \frac{1}{\sqrt{2\pi n\varepsilon(1-\varepsilon)}} \exp\left(-\frac{(m-n\varepsilon)^2}{2n\varepsilon(1-\varepsilon)}\right) \tag{C.3}$$

$$< \int_{m=n/2}^{\infty} \frac{1}{\sqrt{2\pi n\varepsilon(1-\varepsilon)}} \exp\left(-\frac{(m-n\varepsilon)^2}{2n\varepsilon(1-\varepsilon)}\right) \tag{C.4}$$

$$= \frac{1}{2}\left[1 - \text{erf}\left(\frac{n/2 - n\varepsilon}{\sqrt{2n\varepsilon(1-\varepsilon)}}\right)\right] \tag{C.5}$$

$$= P^{\text{upper}}(n;\Upsilon) \tag{C.6}$$

where the error function is a mathematical function defined as $\text{erf}(z) = \frac{2}{\sqrt{\pi}}\int_0^z \exp(-t^2)dt$. □

Corollary 4.4. *For $\forall n \in \mathbb{Z}^+_{>1}$, the probability of TAT selecting an artifact satisfies $P^{\text{artifact}}(n;\Upsilon) < \varepsilon$ and $\lim_{n\to\infty} P^{\text{artifact}}(n;\Upsilon) = 0$.*

Proof. **(1)** We first prove $\lim_{n\to\infty} P^{\text{artifact}}(n;\Upsilon) = 0$ as follows.

By understanding the properties of the error function $\text{erf}(\cdot)$, we analyze its behavior as n approaches infinity. Notably, the error function is monotonically increasing, and $\text{erf}(-\infty) = -1$, $\text{erf}(0) = 0$, and $\text{erf}(\infty) = 1$. Consequently, the upper bound of the probability $P^{\text{upper}}(n;\Upsilon)$ approaches 0, implying that $P^{\text{artifact}}(n;\Upsilon)$ also tends to 0. This is shown mathematically as:

$$n \to \infty \Rightarrow \frac{n/2 - n\varepsilon}{\sqrt{2n\varepsilon(1-\varepsilon)}} \to \infty \Rightarrow \text{erf}\left(\frac{n/2 - n\varepsilon}{\sqrt{2n\varepsilon(1-\varepsilon)}}\right) \to 1 \Rightarrow P^{\text{upper}}(n;\Upsilon) \to 0 \Rightarrow P^{\text{artifact}}(n;\Upsilon) \to 0. \tag{C.7}$$

(2) We then prove $P^{\text{artifact}}(n;\Upsilon) < \varepsilon$ as follows.

As $\text{erf}(\cdot)$ is monotonically increasing, the upper bound probability $P^{\text{upper}}(n;\Upsilon)$ decreases as n increases:

$$n \uparrow \Rightarrow P^{\text{upper}}(n;\Upsilon) \downarrow \tag{C.8}$$

Thus, we have

$$P^{\text{upper}}(n;\Upsilon) \leq P^{\text{upper}}(2;\Upsilon) = \frac{1}{2}\left[1 - \text{erf}\left(\frac{1-2\varepsilon}{\sqrt{4\varepsilon(1-\varepsilon)}}\right)\right]. \tag{C.9}$$

We then aim to prove $P^{\text{upper}}(2;\Upsilon)$ is a monotonically increasing convex function with respect to ε. This involves differentiating the function with respect to ε and examining the signs of the first and second derivatives. Details are as follows.

First, we consider a composite function $F(x)$, which is defined as a composition of two functions $f(x)$ and $g(x)$:

$$F(x) = f(g(x)) = \frac{1}{2}\left[1 - \text{erf}\left(\frac{1-2x}{\sqrt{4x(1-x)}}\right)\right], \text{where } f(x) = \frac{1}{2}\left[1 - \text{erf}(x)\right], g(x) = \frac{1-2x}{\sqrt{4x(1-x)}}. \tag{C.10}$$

The derivatives of $F(x)$ can be obtained by applying the chain rule:

$$F'(x) = f'(g(x)) \cdot g'(x), \tag{C.11}$$

$$F''(x) = f''(g(x)) \cdot g'(x)^2 + g''(x) \cdot f'(g(x)). \tag{C.12}$$

To proceed, we need to determine $f'(x)$, $f''(x)$, $g'(x)$, and $g''(x)$.

The first and second derivatives of $f(x)$ with respect to x are:

$$f'(x) = \frac{\partial \frac{1}{2}\left[1 - \text{erf}(x)\right]}{\partial x} = -\frac{1}{2}\frac{\partial \text{erf}(x)}{\partial x} = -\frac{1}{2}\frac{2}{\sqrt{\pi}}\exp(-x^2) = -\frac{1}{\sqrt{\pi}}\exp(-x^2), \tag{C.13}$$

$$f''(x) = \frac{\partial -\frac{1}{\sqrt{\pi}}\exp(-x^2)}{\partial x} = \frac{2x}{\sqrt{\pi}}\exp(-x^2). \tag{C.14}$$

For the function $g(x)$, we introduce two auxiliary functions $\varphi(x) = \sqrt{\frac{1-x}{x}}$ and $\omega(x) = \sqrt{\frac{x}{1-x}}$ for convenience. Using these, $g(x)$ can be expressed as:

$$g(x) = \frac{1-2x}{\sqrt{4x(1-x)}} = \frac{1}{2}\left(\sqrt{\frac{1-x}{x}} - \sqrt{\frac{x}{1-x}}\right) = \frac{1}{2}\left(\varphi(x) - \omega(x)\right). \tag{C.15}$$

The derivatives of auxiliary functions $\varphi(x)$ and $\omega(x)$ are easily derived:

$$\frac{\partial \varphi(x)}{\partial x} = \frac{\partial \sqrt{\frac{1-x}{x}}}{\partial x} = -\frac{1}{2}\frac{\sqrt{\frac{1-x}{x}} + \sqrt{\frac{x}{1-x}}}{x} = -\frac{1}{2}\frac{\varphi(x) + \omega(x)}{x}, \tag{C.16}$$

$$\frac{\partial \omega(x)}{\partial x} = \frac{\partial \sqrt{\frac{x}{1-x}}}{\partial x} = \frac{1}{2}\frac{\sqrt{\frac{1-x}{x}} + \sqrt{\frac{x}{1-x}}}{1-x} = \frac{1}{2}\frac{\varphi(x) + \omega(x)}{1-x}, \tag{C.17}$$

which are then used to find the derivatives of $g(x)$:

$$
\begin{aligned}
g'(x) =& \frac{\partial \frac{1}{2}\left(\varphi(x)-\omega(x)\right)}{\partial x} && \text{(C.18)}\\
=& \frac{1}{2}\left(-\frac{1}{2}\frac{\varphi(x)+\omega(x)}{x}-\frac{1}{2}\frac{\varphi(x)+\omega(x)}{1-x}\right) && \text{(C.19)}\\
=& -\frac{1}{4}\left(\varphi(x)+\omega(x)\right)\left(\frac{1}{x}+\frac{1}{1-x}\right), && \text{(C.20)}\\
g''(x) =& -\frac{1}{4}\frac{\partial\left[\left(\varphi(x)+\omega(x)\right)\left(\frac{1}{x}+\frac{1}{1-x}\right)\right]}{\partial x} && \text{(C.21)}\\
=& -\frac{1}{4}\left[\frac{\partial(\varphi(x)+\omega(x))}{\partial x}\frac{1}{x(1-x)}+\frac{\partial\frac{1}{x(1-x)}}{\partial x}\Big(\varphi(x)+\omega(x)\Big)\right] && \text{(C.22)}\\
=& -\frac{1}{4}\left[\left(-\frac{1}{2}\frac{\varphi(x)+\omega(x)}{x}+\frac{1}{2}\frac{\varphi(x)+\omega(x)}{1-x}\right)\frac{1}{x(1-x)}+\left(\frac{2x-1}{(x(1-x))^2}\right)\Big(\varphi(x)+\omega(x)\Big)\right] && \text{(C.23)}\\
=& -\frac{1}{8}\left[\left(\frac{2x-1}{x(1-x)}\frac{1}{x(1-x)}+\frac{2x-1}{(x(1-x))^2}\right)\Big(\varphi(x)+\omega(x)\Big)\right] && \text{(C.24)}\\
=& -\frac{1}{4}\left[\left(\frac{2x-1}{(x(1-x))^2}\right)\Big(\varphi(x)+\omega(x)\Big)\right]. && \text{(C.25)}\\
& && \text{(C.26)}
\end{aligned}
$$

With these derivatives in hand, we can now evaluate the first and second derivatives of $P^{\text{upper}}(2;\Upsilon)$ with respect to ε as follows:

$$
\begin{aligned}
&\frac{\partial P^{\text{upper}}(2;\Upsilon)}{\partial\varepsilon} = F'(\varepsilon) && \text{(C.27)}\\
&= f'(g(\varepsilon))\cdot g'(\varepsilon) && \text{(C.28)}\\
&= \underbrace{-\frac{1}{\sqrt{\pi}}\exp\Big(-\left(g(\varepsilon)\right)^2\Big)}_{\text{Negative}}\cdot\underbrace{\left(-\frac{1}{4}\left(\varphi(\varepsilon)+\omega(\varepsilon)\right)\left(\frac{1}{\varepsilon}+\frac{1}{1-\varepsilon}\right)\right)}_{\text{Negative}} > 0, && \text{(C.29)}\\
&\frac{\partial^2 P^{\text{upper}}(2;\Upsilon)}{\partial\varepsilon^2} = F''(\varepsilon) && \text{(C.30)}\\
&= f''(g(\varepsilon))\cdot g'(\varepsilon)^2 + g''(\varepsilon)\cdot f'(g(\varepsilon)) && \text{(C.31)}\\
&= \underbrace{\frac{2g(\varepsilon)}{\sqrt{\pi}}\exp\Big(-\left(g(\varepsilon)\right)^2\Big)\cdot g'(\varepsilon)^2}_{\text{Positive}} + \underbrace{\left(-\frac{1}{4}\left(\frac{2\varepsilon-1}{(\varepsilon(1-\varepsilon))^2}\right)\Big(\varphi(\varepsilon)+\omega(\varepsilon)\Big)\right)}_{\text{Negative}}\cdot\underbrace{\left(-\frac{1}{\sqrt{\pi}}\exp\Big(-\left(g(\varepsilon)\right)^2\Big)\right)}_{\text{Negative}} && \text{(C.32)}\\
&> 0. && \text{(C.33)}
\end{aligned}
$$

These positive first and second derivatives indicate that $P^{\text{upper}}(2;\Upsilon)$ is indeed a monotonically increasing convex function. Lastly, we analyze the behavior of $P^{\text{upper}}(2;\Upsilon)$ at the extreme values of ε within the range (0, 0.5):

$$\lim_{\varepsilon\to 0} P^{\text{upper}}(2;\Upsilon) = 0, \quad \lim_{\varepsilon\to 0.5} P^{\text{upper}}(2;\Upsilon) = 0.5. \tag{C.34}$$

Since they are equal at the edges of (0, 0.5) and $P^{\text{upper}}(2;\Upsilon)$ is a monotonically increasing convex function, the following inequality holds:

$$\forall\varepsilon\in(0,0.5), \quad \varepsilon > P^{\text{upper}}(2;\Upsilon) \geq p^{\text{upper}}(n;\Upsilon) > P^{\text{artifact}}(n;\Upsilon) \tag{C.35}$$

□

D. Experimental Details

D.1. Maze2D

- **Baseline in Maze2D.** We take the scores of MPPI, CQL, IQL, and Diffuser from Table 1 in (Janner et al., 2022).
- **Tree planners in Maze2D.** We deploy TAT on Diffuser, using the pre-trained models of Diffuser released from the authors: (*Diffuser Repository*, https://github.com/anuragajay/decision-diffuser). We adopt default hyperparameters from Diffuser's implementation. Regarding the additional hyperparameters of TAT, we set $\lambda = 0.98$ and $1 - \alpha = 0.0005$. The bath size of sampled trajectories is set to 128.

D.2. Block Stacking

- **Baseline in block stacking.** We take the scores of BCQ, and CQL from Table 2 in (Janner et al., 2022). We ran Diffuser using the official implementation and released pre-trained models from *Diffuser Repository*. Given that there are no publicly available pre-trained models of Decision Diffuser (DD), we retrain DD using the official implementation and default hyperparameters from the authors: (*DD Repository*, https://github.com/anuragajay/decision-diffuser).
- **Tree planners in block stacking.** Diffuser$^{\Upsilon}$ uses the pre-trained models of Diffuser released from *Diffuser Repository* and inherits all default hyperparameters and settings. DD$^{\Upsilon}$ is deployed on top of DD's official implementation and inherits all default hyperparameters and settings from DD. Regarding the hyperparameters of TAT, we set $\lambda = 0.98$ and $1 - \alpha = 0.002$. The bath size of trajectories is set to 64.

D.3. Locomotion

- **Baseline in locomotion.** We take the scores of BC, CQL, and IQL from Table 1 in (Kostrikov et al., 2022); DT from Table 2 in (Chen et al., 2021); TT from Table 1 in (Janner et al., 2021); MOPO from Table 1 in (Yu et al., 2020); MOReL from Table 2 in (Kidambi et al., 2020); MBOP from Table 1 in (Argenson & Dulac-Arnold, 2021); Diffuser from Table 2 in (Janner et al., 2022); RGG from Table 1 in (Lee et al., 2023).
- **Tree planners in locomotion.** Both Diffuser$^{\Upsilon}$ and RGG$^{\Upsilon}$ are deployed on the pre-trained models of Diffuser and Restoration Gap Guidance (RGG), available in their respective official repositories *Diffuser Repository* and *RGG Repository* (https://github.com/leekwoon/rgg). Diffuser$^{\Upsilon}$ and RGG$^{\Upsilon}$ keep all default hyperparameters from Diffuser and RGG. For the hyperparameters of TAT, we set $\lambda = 0.98$, $1 - \alpha = 0.005$ for Walker2d and Hopper, and $1 - \alpha = 0.002$ for HalfCheetah. We found that we could reduce the sampling step for many tasks through warm-start$^{\Upsilon}$ planning (e.g., from default 20 to 10 in the Hopper tasks). Since Diffuser and RGG generate default 64 trajectories for each planning step, we use the first 32 (half) for TAT construction.

D.4. Other Details

- Different diffusion planners have slight variations in their network architectures for diffusion models, but overall, they all feature a U-Net architecture (Ronneberger et al., 2015) with repeated convolutional residual blocks. Each residual block consists of two convolutional layers, followed by group norm (Wu & He, 2018) and a Mish activation function (Misra, 2019).
- All experiments are run on the NVIDIA GeForce RTX 3080 GPU core.

E. The Impact of the Aggregated Trajectory Number

Proposition 4.3 previously established that the likelihood of artifact selection decreases as the number n of involved trajectories increases. It suggests that a higher number leads to more stable and trustworthy outcomes. To validate this, we varied the batch size for Diffuser$^{\Upsilon}$ in the Hopper Medium task and measured the average number of trajectories contributing to an action generation. The results are shown in Table E.1. We can see that more trajectories correlate with better performance, which is consistent with the theoretical analysis. However, a higher number also means a greater computational budget. Meanwhile, the performance gain becomes marginal as the number becomes sufficiently large. Thus, there is a trade-off.

Table E.1. The performance of Diffuser$^{\Upsilon}$ under varying numbers of trajectories in the Hopper Medium task.

Batch Size	Average Trajectories per Action	Score
1	$1.0_{\pm 0.0}$	$74.0_{\pm 2.2}$
8	$13.4_{\pm 0.3}$	$77.4_{\pm 2.4}$
16	$24.0_{\pm 0.4}$	$80.7_{\pm 2.3}$
(Default) 32	$49.4_{\pm 0.7}$	$82.6_{\pm 1.9}$
64	$96.1_{\pm 0.9}$	$84.5_{\pm 2.0}$
128	$179.9_{\pm 1.2}$	$85.2_{\pm 2.0}$
256	$358.7_{\pm 3.0}$	$86.2_{\pm 2.1}$
512	$668.1_{\pm 6.5}$	$86.9_{\pm 1.9}$

F. Time and Memory Budget

In this part, we analyze the additional time and memory budget for maintaining TAT.

We present the average planning time of generating an action taken by baseline Diffuser and Diffuser$^{\Upsilon}$ on MuJoCo locomotion (Fu et al., 2020) in Table F.1. Notably, the time consumption of TAT is less than 5% of the total planning time.

We present the memory usage of baseline Diffuser and Diffuser$^{\Upsilon}$ on Hopper Medium task in Table F.2. As shown, the memory usage for the tree is small. TAT typically manages an average of 200-400 nodes, with each node only storing 3 statistics. This volume of data is significantly smaller than that of U-Net's sampling in the vanilla diffusion planner.

Table F.1. The planning time of Diffuser$^{\Upsilon}$ and Diffuser on MuJoCo locomotion tasks. The results are averaged over 200 actions' generation. All results are tested on a single NVIDIA GeForce RTX 3080 GPU core. The unit is second (s).

*-Medium	Method	Sampling Time	Tree Time	Total
HalfCheetah	Diffuser	1.43	-	1.43
HalfCheetah	Diffuser$^{\Upsilon}$	1.43	0.02	1.45
Hopper	Diffuser	1.68	-	1.68
Hopper	Diffuser$^{\Upsilon}$	1.68	0.06	1.74
Walker2d	Diffuser	1.69	-	1.69
Walker2d	Diffuser$^{\Upsilon}$	1.68	0.07	1.75

Table F.2. The memory usage of Diffuser$^{\Upsilon}$ and Diffuse on Hopper Medium task. All results are tested on a single NVIDIA GeForce RTX 3080 GPU core.

Method	U-Net Sampling Memory	Tree Memory	Other Fixed Memory	Total
Diffuser	2501.7 MiB	-	2965.3 MiB	5467.1 MiB
Diffuser$^{\Upsilon}$	2501.9 MiB	16.8 MiB	2964.9 MiB	5483.7 MiB

G. Visual Comparison in Maze2D

We show the stochastic infeasible plans of Diffuser in the Maze2D environments in Figure G.1. Given the same scenarios, Diffuser$^{\Upsilon}$ can effectively filter them out by TAT.

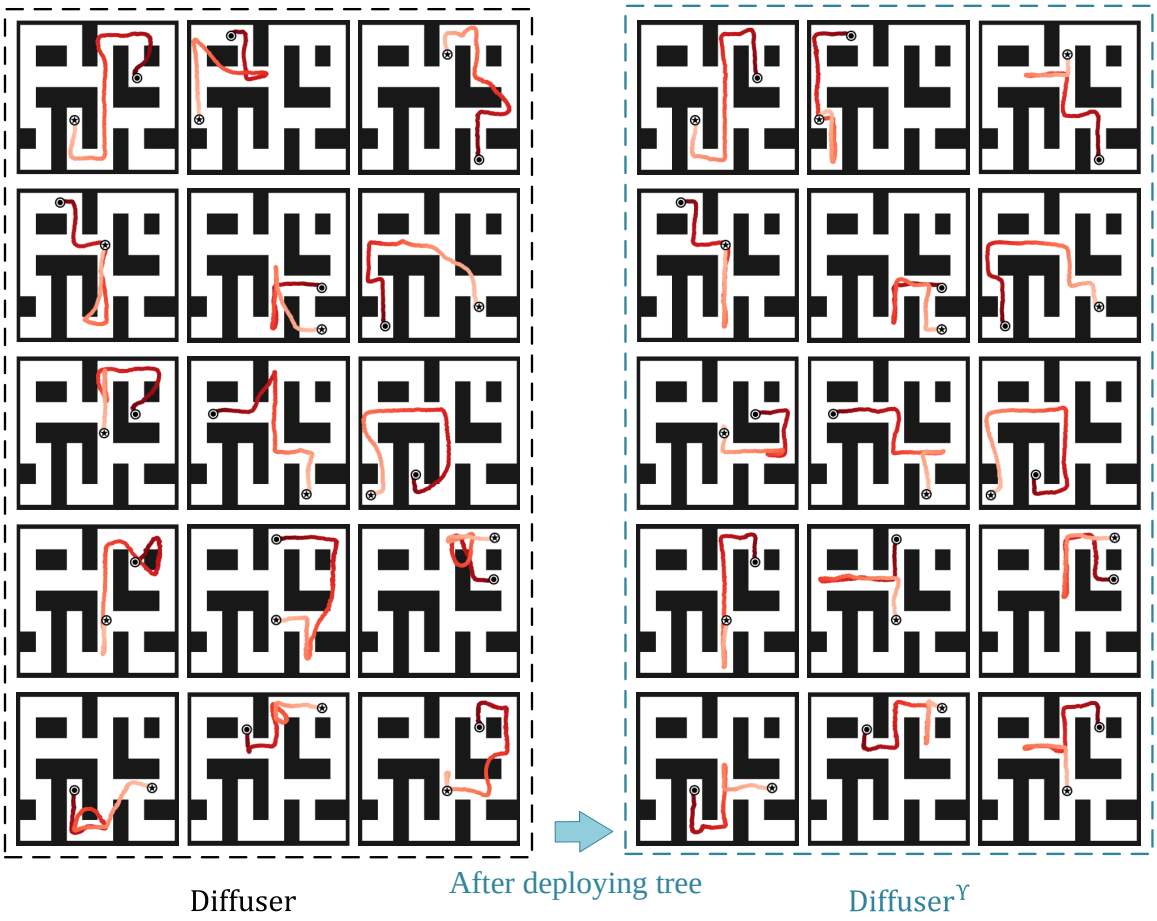

Figure G.1. Visual comparison of Diffuser and Diffuser$^{\Upsilon}$ in the presence of artifacts in the Maze2D-Large environments. ⦿ denotes the starting position and ⊛ denotes the goal position.

H. Visual Comparison in Kuka Block Stacking

We present the stochastic artifacts of Diffuser in the Kuka block stacking environments in Figure H.1. Given the same scenarios, Diffuser$^{\Upsilon}$ can effectively handle these artifacts by TAT.

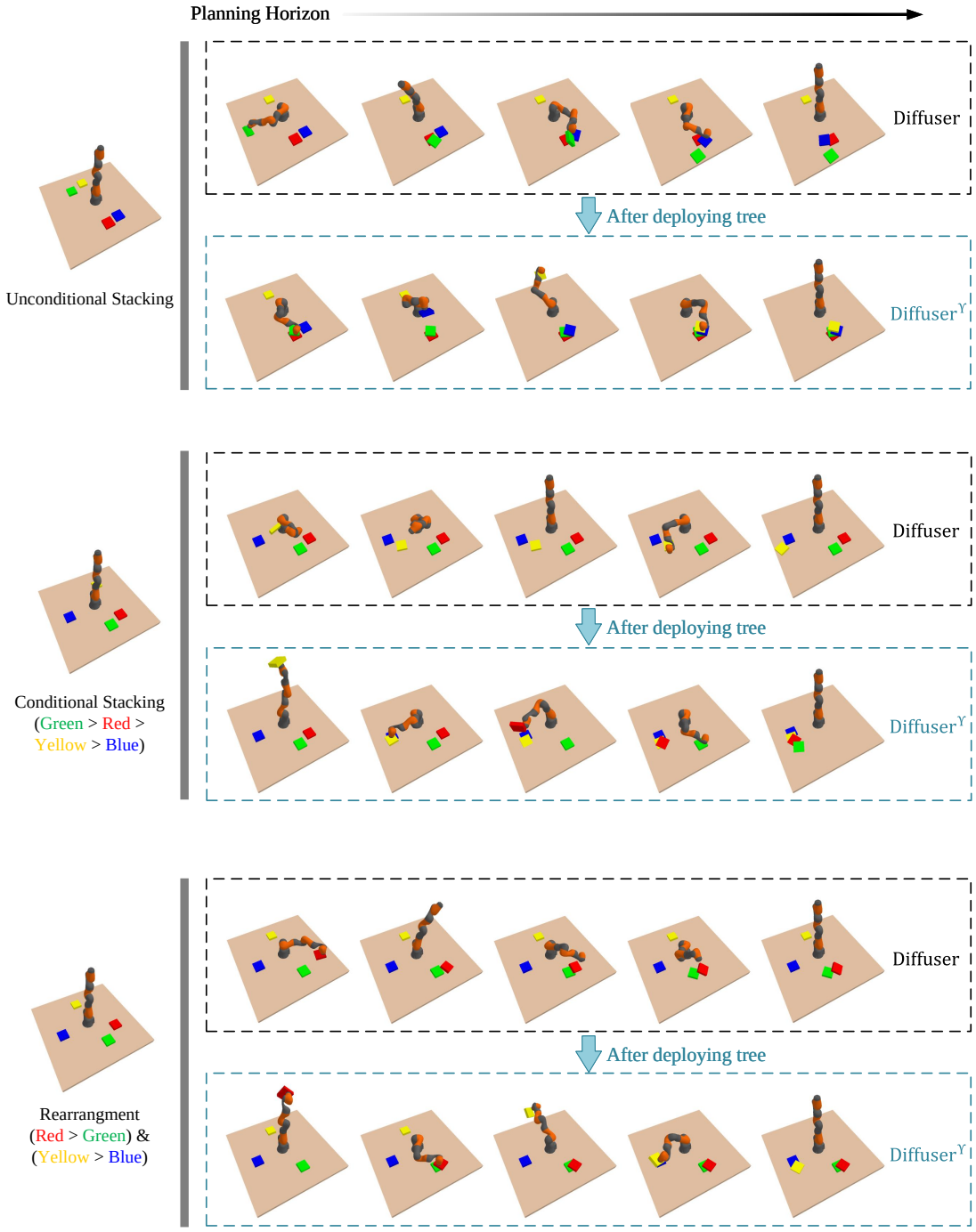

Figure H.1. Visual comparison of Diffuser and Diffuser$^{\Upsilon}$ in the presence of artifacts in the Kuka block stacking environments.

I. Visual Comparison in MuJoCo Locomotion

We show the stochastic suboptimal and unreliable plans of Diffuser in the MuJoCo locomotion tasks in Figures I.1. Given the same scenarios, Diffuser$^{\Upsilon}$ can effectively handle these suboptimal and unreliable plans by TAT.

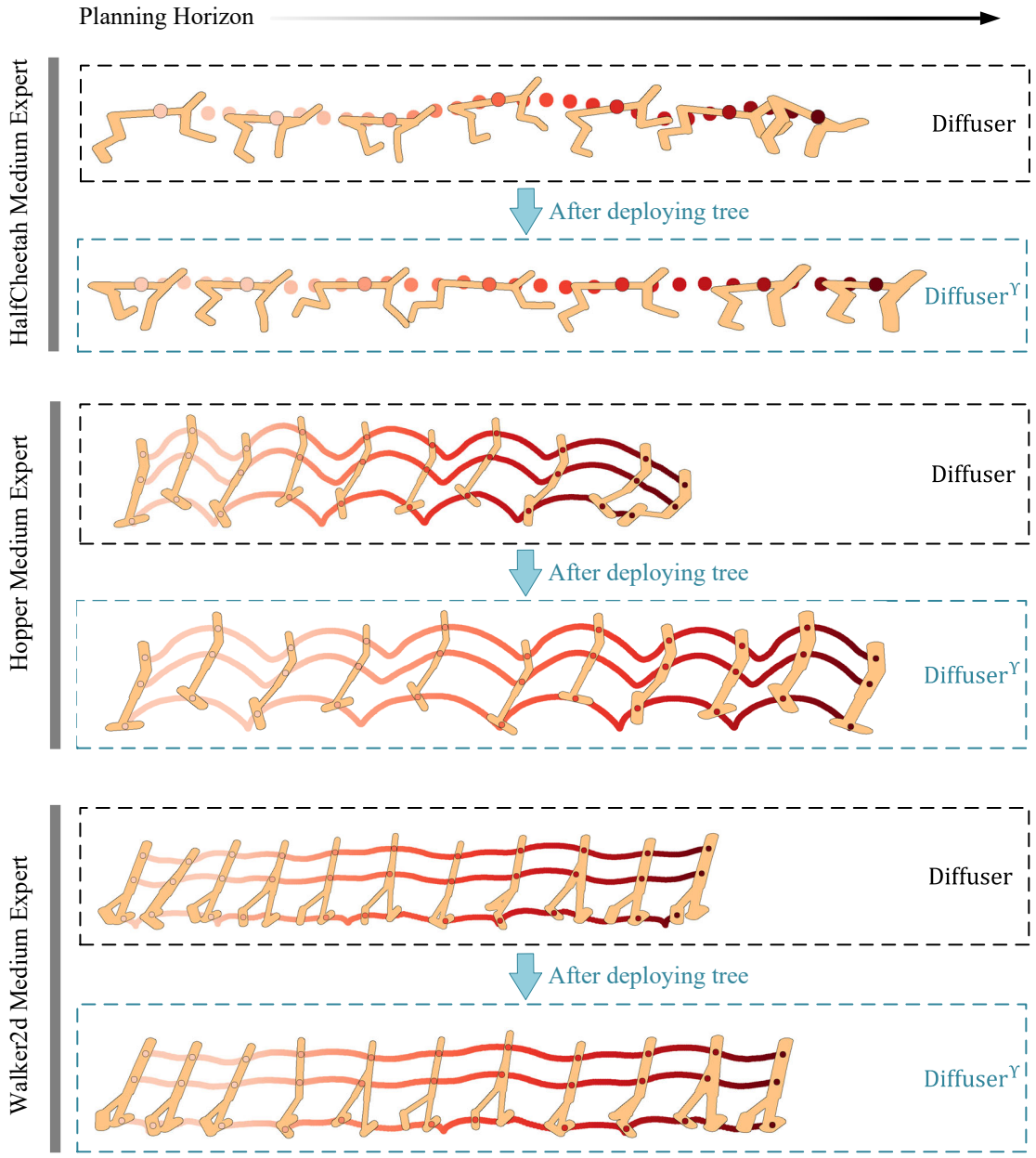

Figure I.1. Visual comparison of Diffuser and Diffuser$^{\Upsilon}$ in the presence of artifacts in the MuJoCo locomotion tasks.

J. Artifacts in Vision Domains

The artifacts, denoting defective or infeasible outcomes, have been a key concern in studies on generative neural networks, particularly in vision domains. Bau et al. (2019); Zhang et al. (2019); Tousi et al. (2021) have made efforts to understand the inner workings of Generative Adversarial Networks (GAN). These studies identified and adjusted the internal units that produced artifacts within GANs. Shen et al. (2020) further elucidated how the latent space coded different semantics and trained a linear classifier based on artifact-labeled data to mitigate artifact generation. Other works (Choi et al., 2022; Jeong et al., 2022) have been centered on unsupervised methods for detecting and rectifying artifacts in generative models.