Planning With Diffusion for Flexible Behavior Synthesis

Matthew Bronars and Sreyas Venkataraman

Carnegie Mellon University

Paper Information

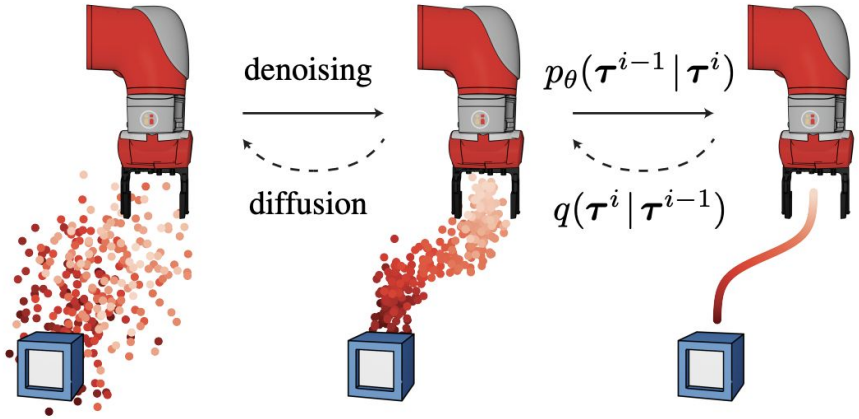
Authors: Michael Janner, Yilun Du, Joshua Tenebaum, Sergey Levine

Publication Venue: ICML 2022

Affiliations: UC Berkely, MIT

Motivation

- Planning in learned world models is easily exploited with standard trajectory optimization
- Can more of the trajectory optimization be folded into the world modeling?
- Desiderata:
  - Data driven
  - Long horizon scalability
  - Task compositionality
  - Temporal compositionality
  - Effective non-greedy planning

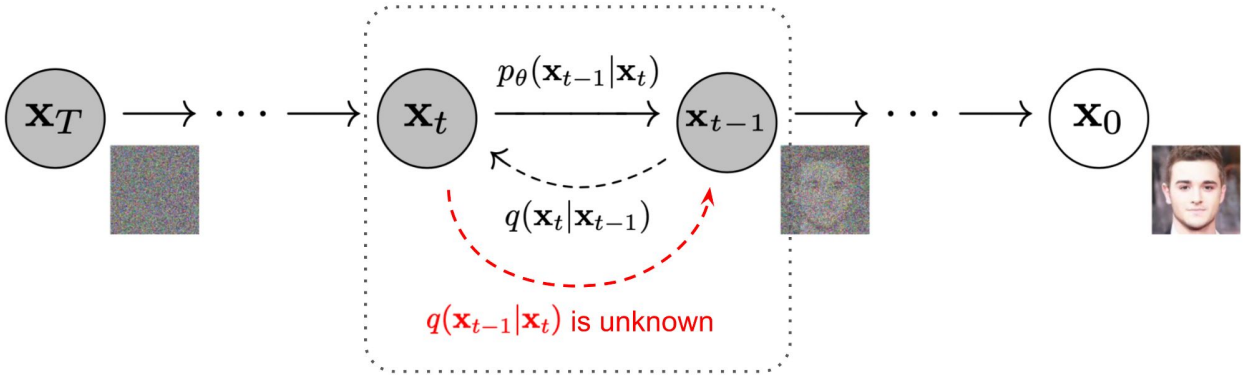


Background

Trajectory Optimization - finding a sequence of actions that maximizes an objective factorized over per-timestep rewards.

$$\mathbf{a}^*_{0:T} = \arg\max_{\mathbf{a}_{0:T}} \mathcal{J}(\mathbf{s}_0, \mathbf{a}_{0:T}) = \arg\max_{\mathbf{a}_{0:T}} \sum_{t=0}^{T} r(\mathbf{s}_t, \mathbf{a}_t)$$

Diffusion Probabilistic Models - pose the data-generation process as an iterative denoising procedure



Carnegie Mellon University

Related Work - Guided Diffusion

- Requires differentiable cost/reward function
- Taking gradient step on noise while denoising maximizes reward

Algorithm 2 Classifier guided DDIM sampling, given a diffusion model $\epsilon_\theta(x_t)$

Input: class label y, gradient scale s
$x_T \leftarrow$ sample from $\mathcal{N}(0, \mathbf{I})$
for all t from T to 1 **do**
  $\hat{\epsilon} \leftarrow \epsilon_\theta(x_t) - \sqrt{1-\bar{\alpha}_t}\,\nabla_{x_t} \log p_\phi(y|x_t)$
  $x_{t-1} \leftarrow \sqrt{\bar{\alpha}_{t-1}}\left(\frac{x_t - \sqrt{1-\bar{\alpha}_t}\hat{\epsilon}}{\sqrt{\bar{\alpha}_t}}\right) + \sqrt{1-\bar{\alpha}_{t-1}}\hat{\epsilon}$
end for
return x_0

$$x_{0|t} = \frac{1}{\sqrt{\bar{\alpha}_t}}(x_t - \sqrt{1-\bar{\alpha}_t}\epsilon_\theta(x_t, t))$$

No Guidance

Guidance

Carnegie Mellon University

Related Work - Planning in World Models

- Model Based Reinforcment Learning
  - Still asymptotically dominated by model-free methods
  - Borrow heavily from model-free methods

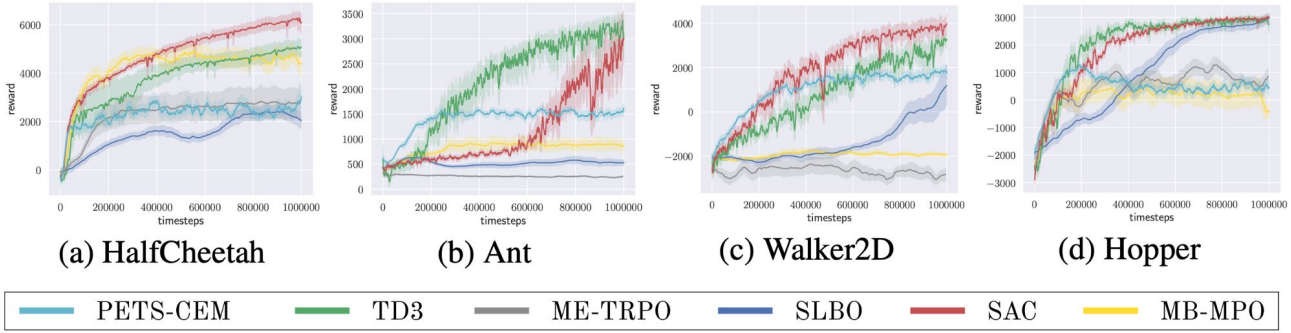


Figure 2: Performance curve for each algorithm trained for 1 million time-steps.

- World Model + Traditional Planning
  - Traj optimization based methods can generate adversarial examples
  - Random shooting - does not have same exploitation, but may be too simple

Carnegie Mellon University

Method - Overview

Key Idea:

- Instead of having one model for world modeling and another model for acting, model both at same time
- Sampling = Planning
- Model whole trajectory at same time
- Single diffusion model can be used for multiple tasks in same environment

$$\tilde{p}_\theta(\boldsymbol{\tau}) \propto p_\theta(\boldsymbol{\tau}) h(\boldsymbol{\tau}).$$

- Guided sampling allows for flexible planning

Algorithm 1 Guided Diffusion Planning

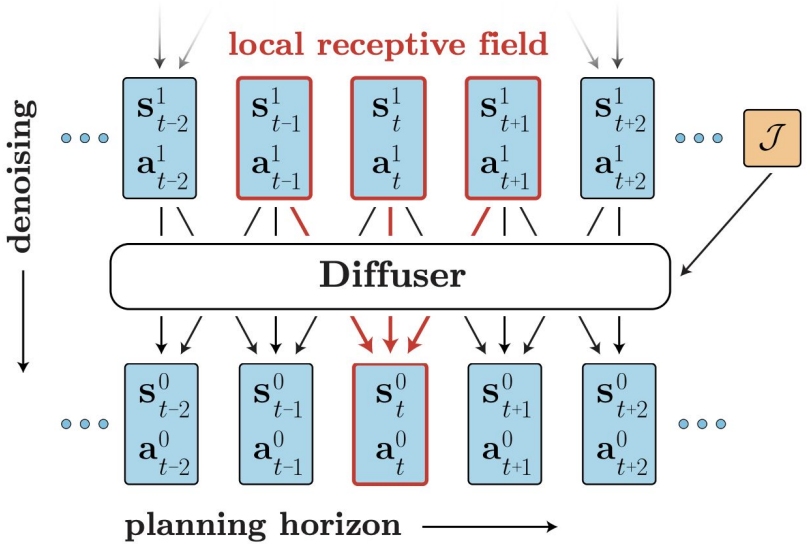
Require Diffuser μ_θ, guide $\mathcal{J}$, scale α, covariances Σ^i
while not done **do**
  Observe state $\mathbf{s}$; initialize plan $\boldsymbol{\tau}^N \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$
  for $i = N, \ldots, 1$ **do**
    // parameters of reverse transition
    $\mu \leftarrow \mu_\theta(\boldsymbol{\tau}^i)$
    // guide using gradients of return
    $\boldsymbol{\tau}^{i-1} \sim \mathcal{N}(\mu + \alpha \Sigma \nabla \mathcal{J}(\mu), \Sigma^i)$
    // constrain first state of plan
    $\boldsymbol{\tau}^{i-1}_{\boldsymbol{s}_0} \leftarrow \boldsymbol{s}$
  Execute first action of plan $\boldsymbol{\tau}^0_{\mathbf{a}_0}$

Carnegie Mellon University

Method - Design Decisions

Key Features:

- Jointly denoise states and actions
- No causal masking - predict all timesteps of the plan concurrently
- Convolutional -
  - Denoising steps are temporally local
  - No fixed planning horizon
- Plan and rollout in receding horizon fashion



Guided Diffusion Planning

Practical Pipeline:

1. Train diffusion model on all available trajectory data (states + actions)
2. Train reward model J to predict cumulative rewards of noisy samples
3. Guide reverse process using reward gradients

At test time:

- Execute first action of sampled plan, replan (receding horizon)

$$p_\theta(\boldsymbol{\tau}^{i-1} \mid \boldsymbol{\tau}^i, \mathcal{O}_{1:T}) \approx \mathcal{N}(\boldsymbol{\tau}^{i-1}; \mu + \Sigma g, \Sigma)$$

$$\begin{aligned} g &= \nabla_{\boldsymbol{\tau}} \log p(\mathcal{O}_{1:T} \mid \boldsymbol{\tau})|_{\boldsymbol{\tau}=\mu} \\ &= \sum_{t=0}^{T} \nabla_{\mathbf{s}_t, \mathbf{a}_t} r(\mathbf{s}_t, \mathbf{a}_t)|_{(\mathbf{s}_t, \mathbf{a}_t)=\mu_t} = \nabla \mathcal{J}(\mu). \end{aligned}$$

Carnegie Mellon University

RL as Inpainting

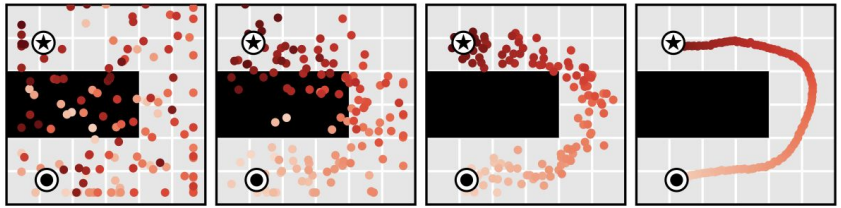
Core Idea:

- Goal-conditioned RL = constraint satisfaction, not reward max
- Trajectory as 2D array: constrained states/actions = “observed pixels”
- Diffusion model fills in unobserved cells consistently

Implementation:

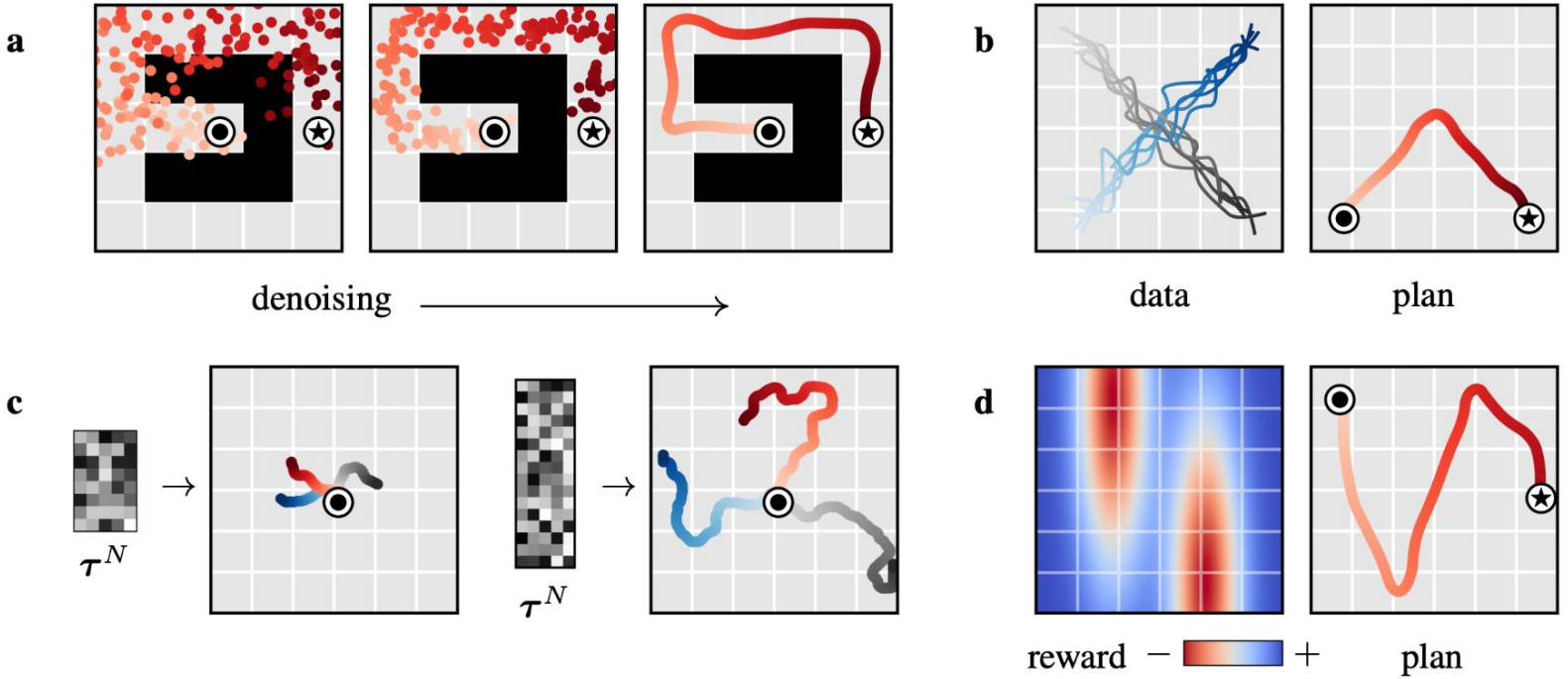
- Sample from unperturbed reverse process, then replace constrained values at every denoising step
- Always inpaint current state s at t=0 (receding horizon)

$$h(\boldsymbol{\tau}) = \delta_{\mathbf{c}_t}(\mathbf{s}_0, \mathbf{a}_0, \ldots, \mathbf{s}_T, \mathbf{a}_T) = \begin{cases} +\infty & \text{if } \mathbf{c}_t = \mathbf{s}_t \\ 0 & \text{otherwise} \end{cases}$$



Carnegie Mellon University

Some interesting properties



Experiments - Overview

Settings they evaluate:

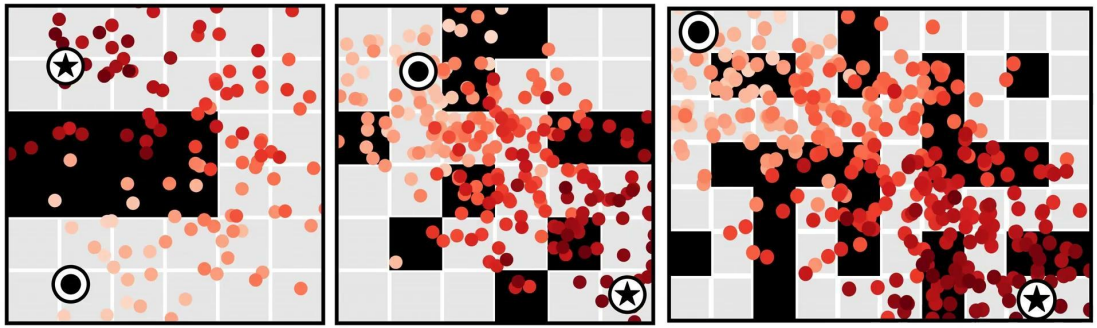
- (1) Long-horizon planning — sparse rewards, no reward shaping
- (2) Test-time flexibility — generalize to novel goals/tasks unseen during training
- (3) Offline RL — recover effective control from heterogeneous data
- (4) Warmstarting to improve inference speed

Long-Horizon Planning

Environment: Maze2D

- Sparse reward — reward of 1 only at goal location
- Single-task: fixed goal | Multi-task: goal resampled each episode
- Uses inpainting to condition on start and goal state
- Diffuser does NOT retrain for multi-task — just changes inpainting goal

Environment		MPPI	CQL	IQL	Diffuser
Maze2D	U-Maze	33.2	5.7	47.4	**113.9** ±3.1
Maze2D	Medium	10.2	5.0	34.9	**121.5** ±2.7
Maze2D	Large	5.1	12.5	58.6	**123.0** ±6.4
Single-task Average		16.2	7.7	47.0	**119.5**
Multi2D	U-Maze	41.2	-	24.8	**128.9** ±1.8
Multi2D	Medium	15.4	-	12.1	**127.2** ±3.4
Multi2D	Large	8.0	-	13.9	**132.1** ±5.8
Multi-task Average		21.5	-	16.9	**129.4**



Carnegie Mellon University

Task Compositionality

Block stacking: 3 tasks, 1 trained model

- Unconditional Stacking — maximize stack height
- Conditional Stacking — specified block order
- Rearrangement — match reference configuration

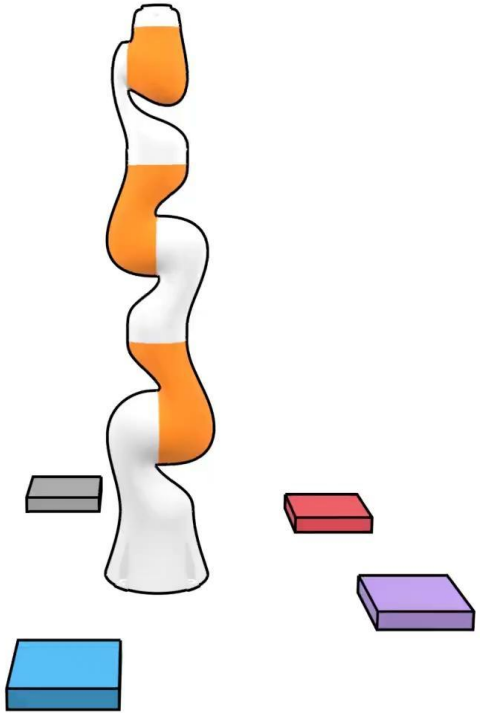
Method: compose h(tau) perturbation functions

- Perturbation 1: maximize likelihood of target final state
- Perturbation 2: enforce contact constraint (end-effector to cube)
- No retraining needed — new tasks at test time via h(tau)

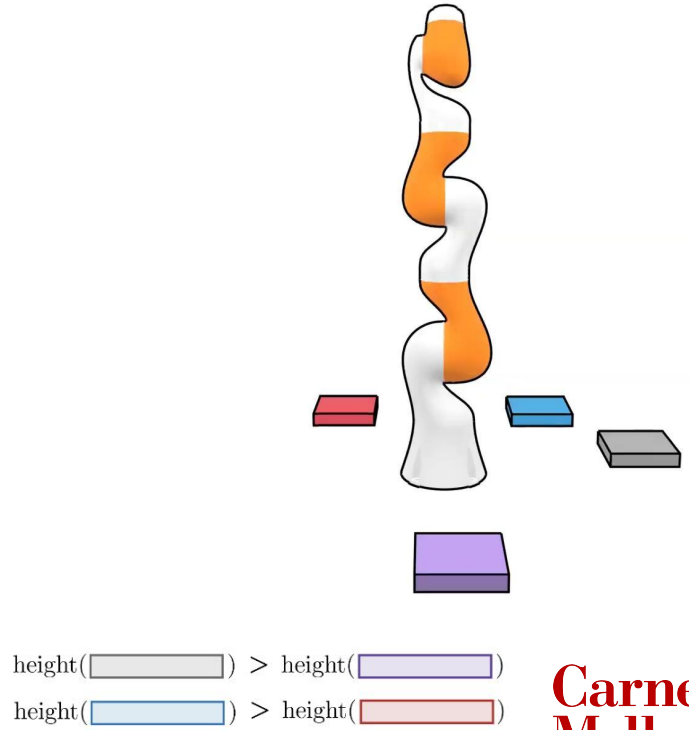
Environment	BCQ	CQL	Diffuser
Unconditional Stacking	0.0	24.4	**58.7** ±2.5
Conditional Stacking	0.0	0.0	**45.6** ±3.1
Rearrangement	0.0	0.0	**58.9** ±3.4
Average	0.0	8.1	**54.4**

Carnegie Mellon University

Unconditional



Conditional



Carnegie Mellon University

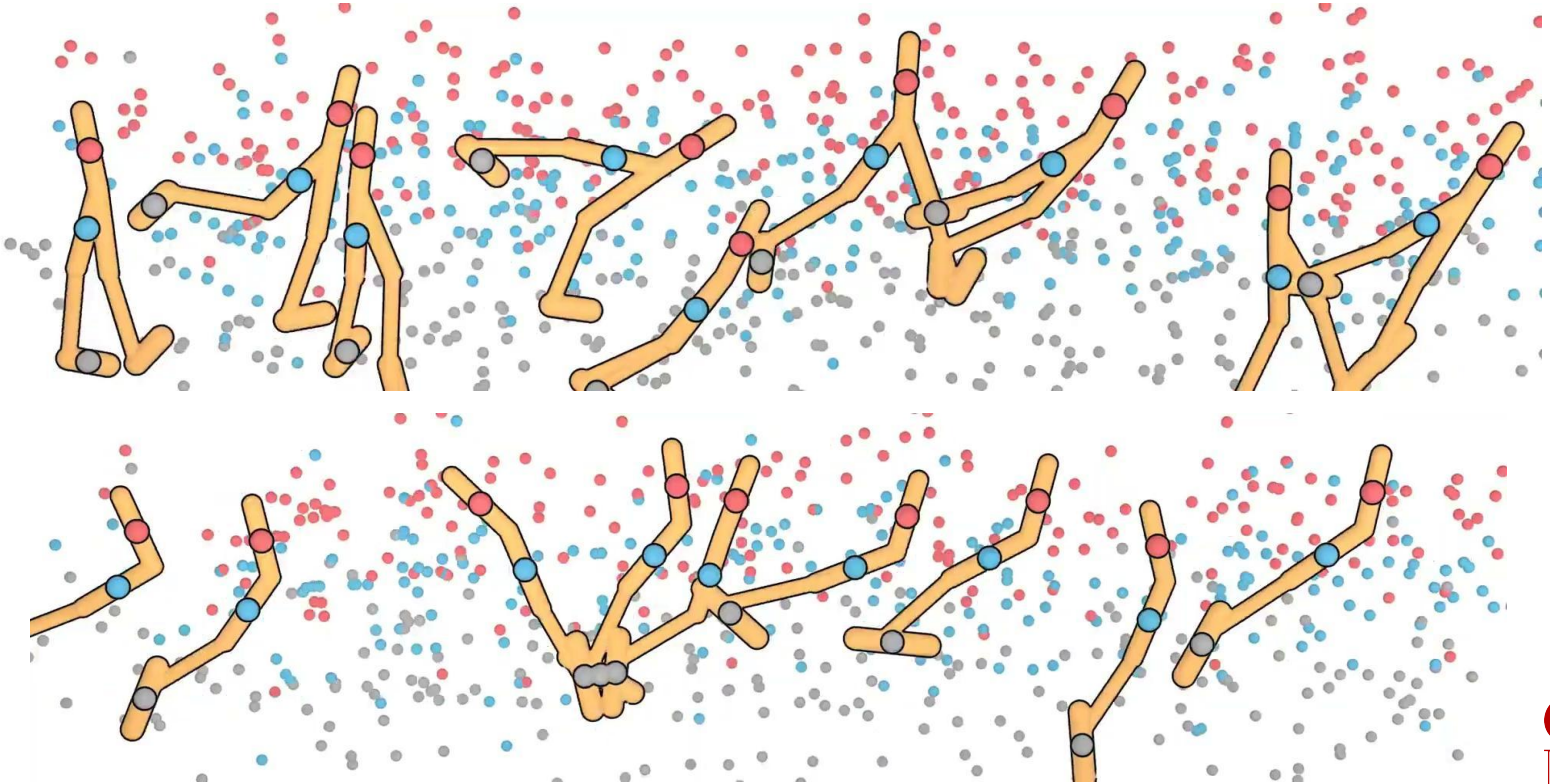
Offline RL (D4RL Locomotion)

Setup:

- Combines guided sampling + inpainting
- Reward model J trained on same offline trajectories
- Inpaint current state s_t at each timestep (receding horizon)

Dataset	Environment	BC	CQL	IQL	DT	TT	MOPO	MOReL	MBOP	Diffuser
Medium-Expert	HalfCheetah	55.2	91.6	86.7	86.8	95.0	63.3	53.3	**105.9**	88.9 ±0.3
Medium-Expert	Hopper	52.5	**105.4**	91.5	**107.6**	**110.0**	23.7	**108.7**	55.1	103.3 ±1.3
Medium-Expert	Walker2d	**107.5**	**108.8**	**109.6**	**108.1**	101.9	44.6	95.6	70.2	**106.9** ±0.2
Medium	HalfCheetah	42.6	44.0	**47.4**	42.6	**46.9**	42.3	42.1	44.6	42.8 ±0.3
Medium	Hopper	52.9	58.5	66.3	67.6	61.1	28.0	**95.4**	48.8	74.3 ±1.4
Medium	Walker2d	75.3	72.5	**78.3**	74.0	**79.0**	17.8	**77.8**	41.0	**79.6** ±0.55
Medium-Replay	HalfCheetah	36.6	45.5	44.2	36.6	41.9	**53.1**	40.2	42.3	37.7 ±0.5
Medium-Replay	Hopper	18.1	**95.0**	**94.7**	82.7	**91.5**	67.5	**93.6**	12.4	**93.6** ±0.4
Medium-Replay	Walker2d	26.0	77.2	73.9	66.6	**82.6**	39.0	49.8	9.7	70.6 ±1.6
Average		51.9	**77.6**	**77.0**	74.7	**78.9**	42.1	72.9	47.8	**77.5**

Offline RL (D4RL Locomotion)



Carnegie Mellon University

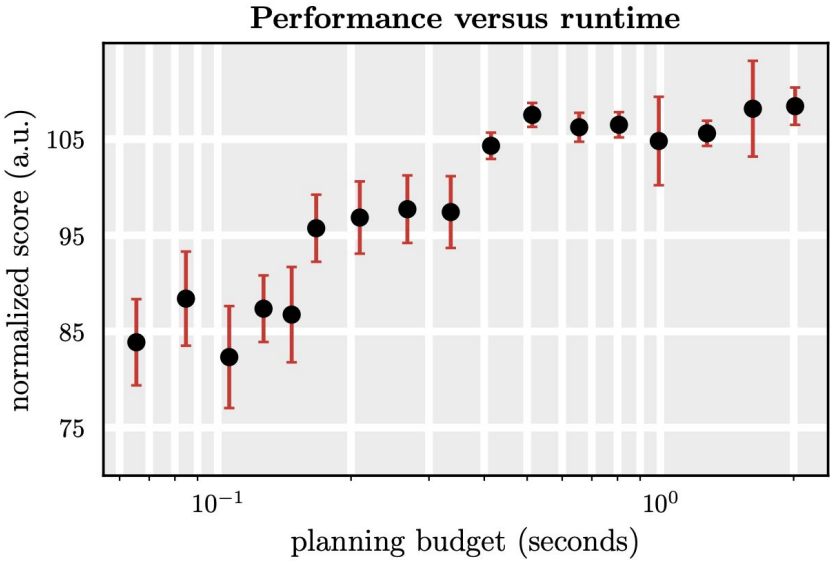
Warmstarting for Faster Planning

Problem:

- Iterative generation is slow — full replan at every execution step

Solution: warm-start from previous plan

- Step 1: apply k forward diffusion steps to previous plan (add noise)
- Step 2: run k denoising steps to regenerate updated plan
- Only k << N steps needed instead of full N-step diffusion



Carnegie Mellon University

Conclusion and Future Work

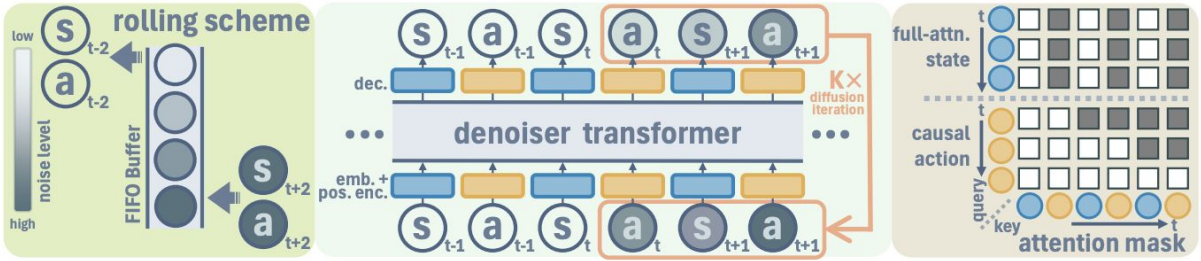


Fig. 3. **Framework of Diffuse-CLoC**. The rolling scheme (Left) is implemented as a FIFO buffer, where at every timestep t, a state and action pair of pure noise is pushed into the denoiser, while the earliest clean pair is popped out and used to step the simulator. The denoiser architecture (Middle) denoises the buffer of states and actions with an increasing noise level along the trajectory. The observation O_t is directly inpainted into the sequence, and only noisy future predictions are denoised. Notice that noisy states and actions have different shades, meaning that their noise levels $\boldsymbol{k_s}$, $\boldsymbol{k_a}$ can be different. The attention mask (Right) shows that state attends to all past and future states but masks out all actions, while action is causal attending to previous states and actions.