IS CONDITIONAL GENERATIVE MODELING ALL YOU NEED FOR DECISION MAKING?

a.k.a DECISION DIFFUSER

01 논문 요약

Abstract

- **Conditional Generative Modeling**을 **순차적 의사 결정 문제**에 적용해보았다.
- Generative Modeling이 표준 벤치마크에서 기존 오프라인 RL 접근 방식을 능가할 수 있는 Policy도 만들 수 있었다.
- Policy를 Conditional Diffusion model로 모델링함으로써 **Dynamic Programming**을 사용하지 않을 수 있었고 DP의 Side effect도 없앨 수 있었다. (contribution)
- Constraints와 Skill이라는 두 가지 다른 조건 변수를 추가하여 Conditional Diffusion model로서 정책 모델링의 이점을 추가로 보여준다. (Contribution 2)

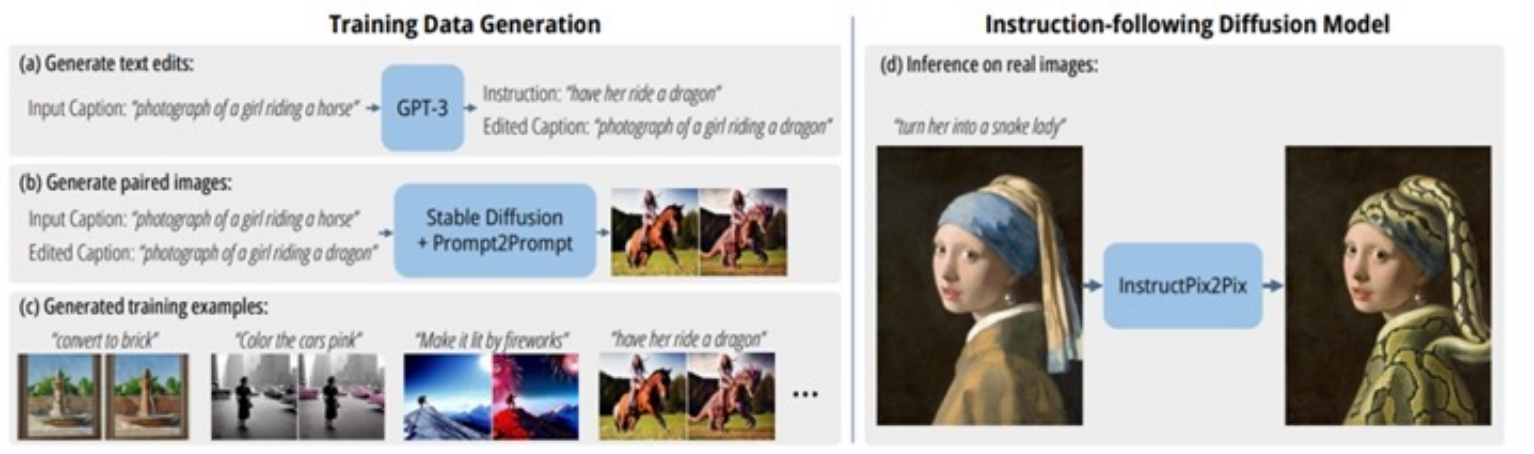
서론

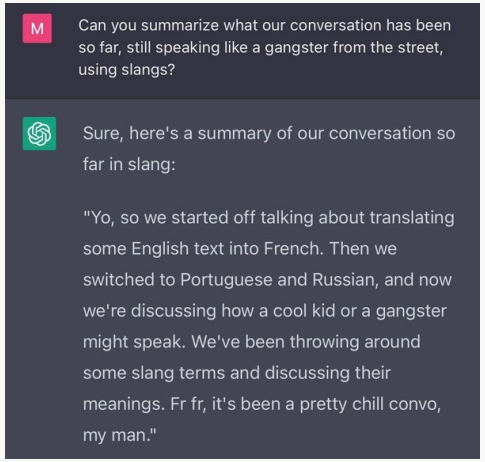
ALL YOU NEED IS DIFFUSER IN REINFORCEMENT LEARNNING?

01 Decision Diffuser - 서론

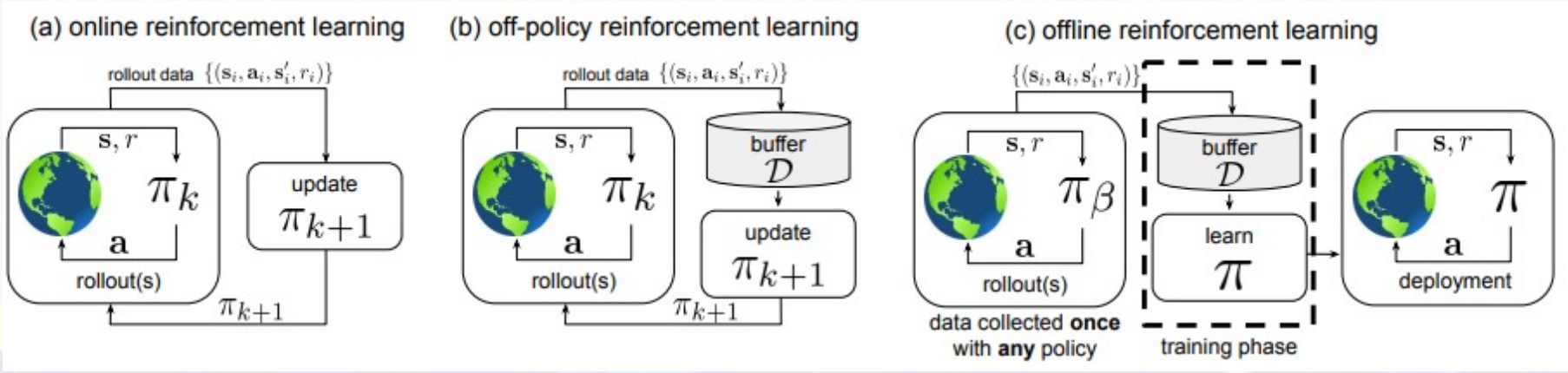
서론

Over the last few years, there have been incredible advances in the field of **conditional generative modeling**





there exists a wide body of work that has looked into recovering high-performing policies from data logged by already operational systems (**offline RL**)



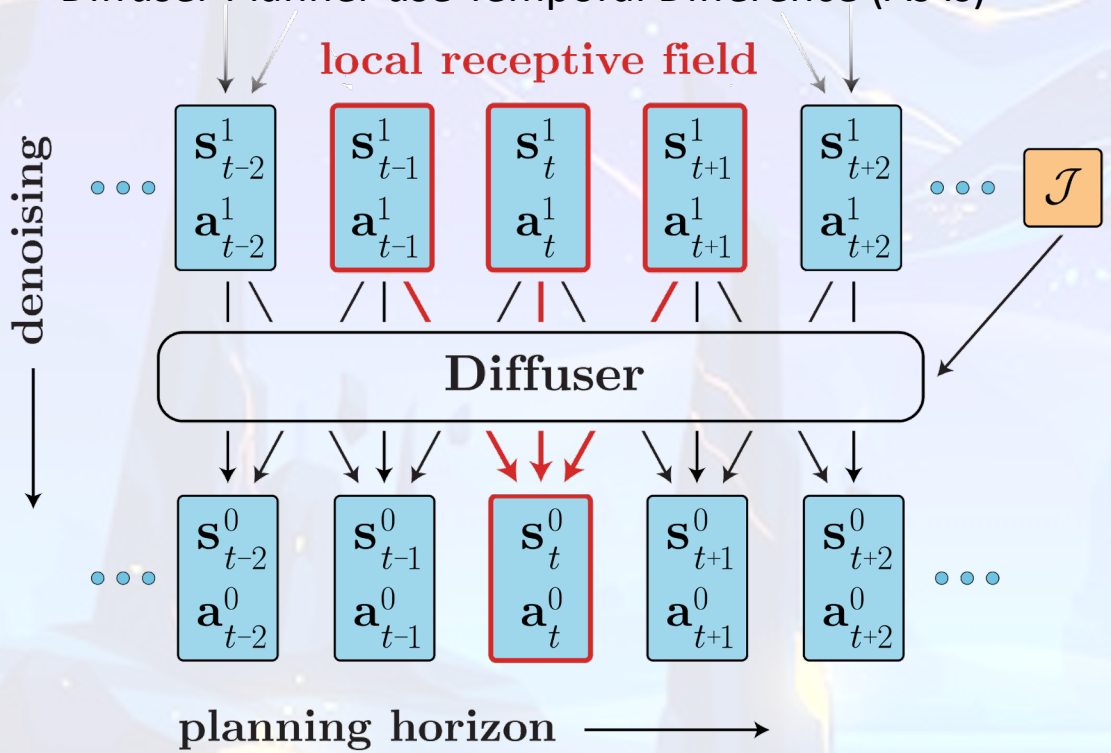
01 Decision Diffuser - 서론

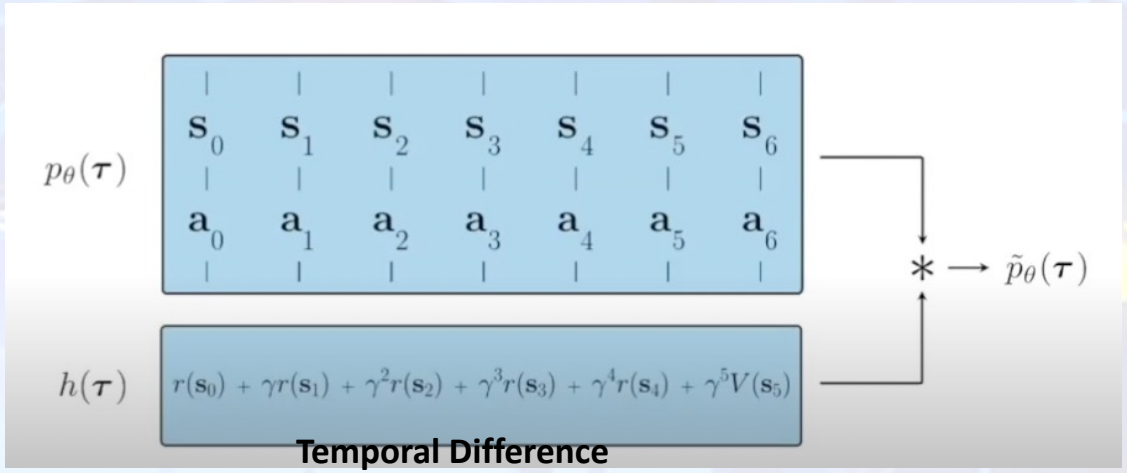
서론

이 논문에서는 기존에 Diffuser Planner(Janner et al.)에서 사용했던 Guidance를 변경했다가 주 포인트입니다.

Classifier Guidance -> Classifier free guidance

Diffuser Planner use Temporal Difference (As Is)





서론

We formulate the problem of **offline decision-making** from the perspective of **conditional generative modeling.**

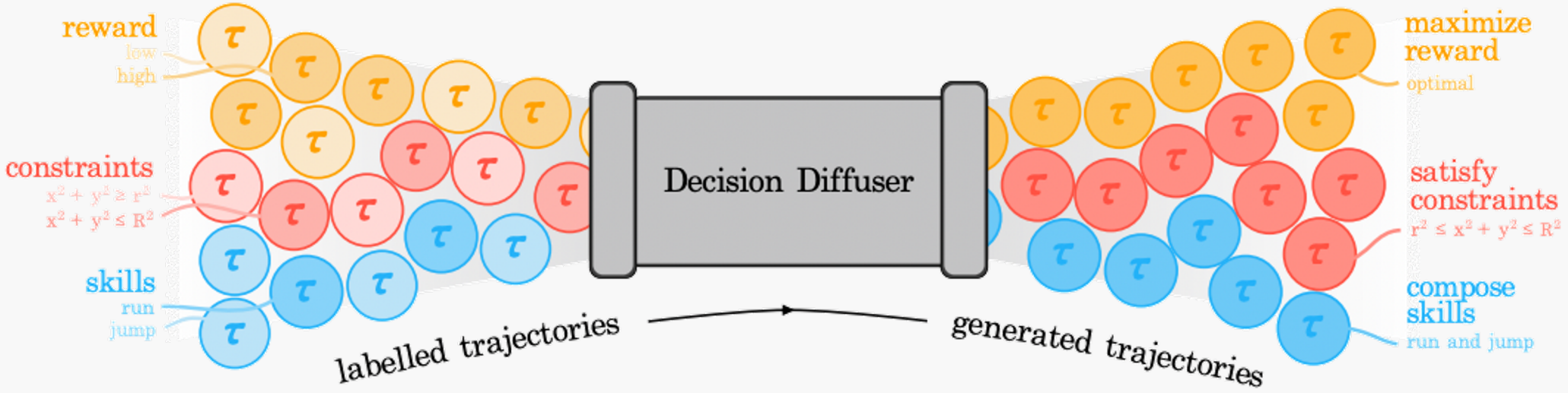
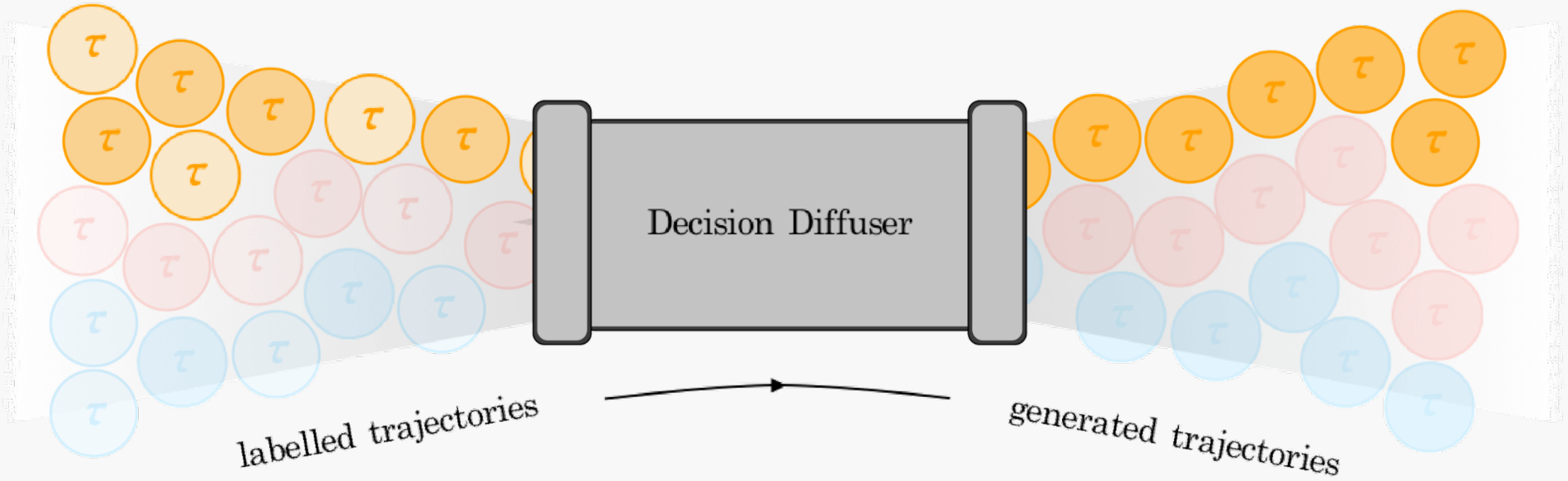


Figure 1: **Decision Making using Conditional Generative Modeling.** Framing decision making as a conditional generative modeling problem allows us to maximize rewards, satisfy constraints and compose skills.

01 Decision Diffuser - 서론

서론

given trajectories with sub-optimal reward, we can generate trajectories with maximum reward



1. Make Objective
2. Merge multiple Constraints
3. Compose Skills

03 Generative modeling with Decision Diffuser

모델 관련

Problem Definition $\max_{\theta} \mathbb{E}_{\tau \sim \mathcal{D}}[\log p_\theta(\boldsymbol{x}_0(\tau)|\boldsymbol{y}(\tau))]$

Noising process: $q(\boldsymbol{x}_{k+1}(\tau)|\boldsymbol{x}_k(\tau)),$

Denoising process: $p_\theta(\boldsymbol{x}_{k-1}(\tau)|\boldsymbol{x}_k(\tau), \boldsymbol{y}(\tau))$

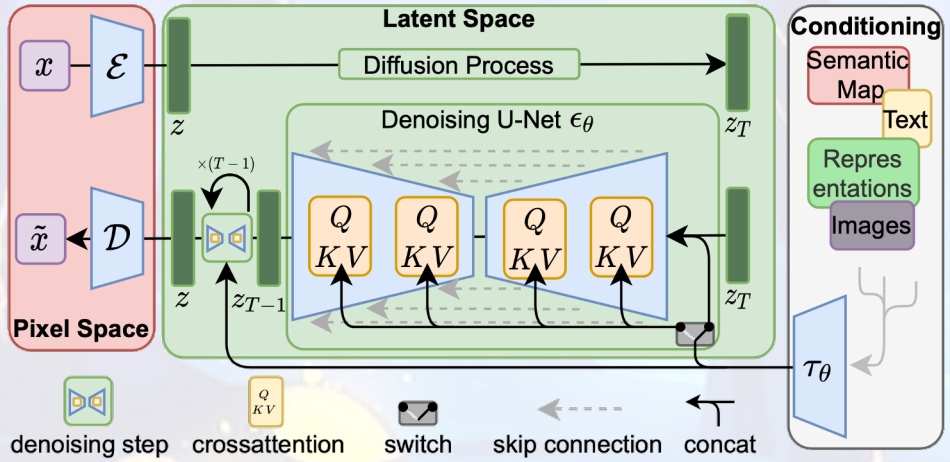
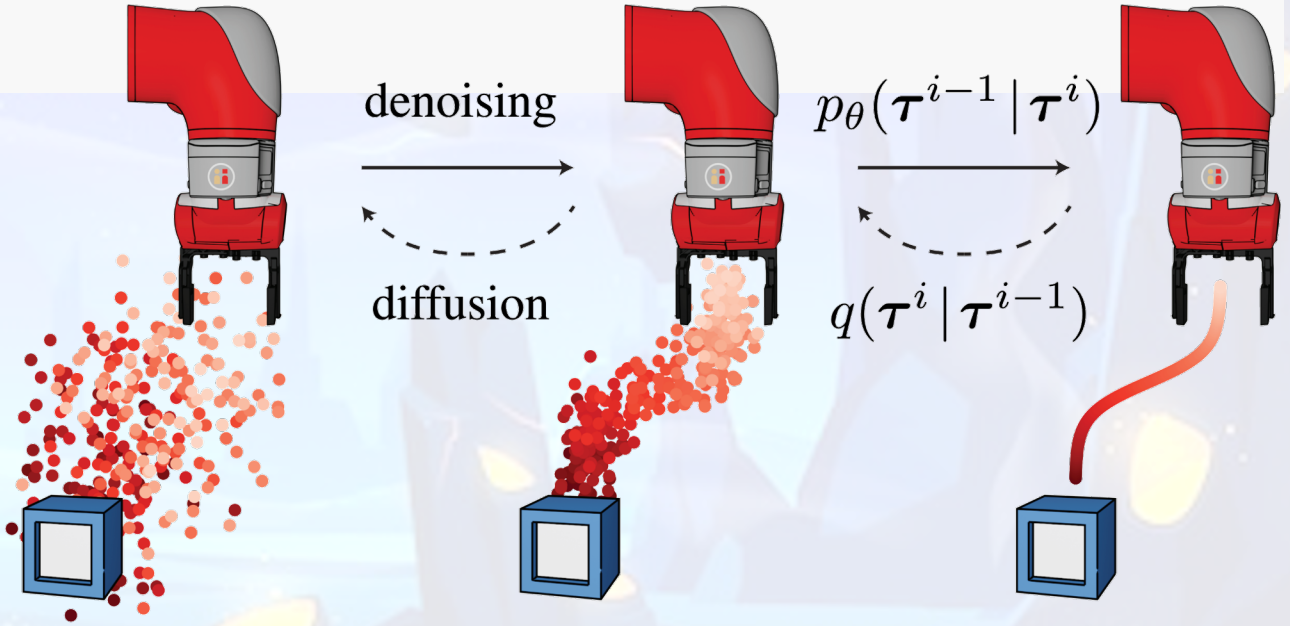
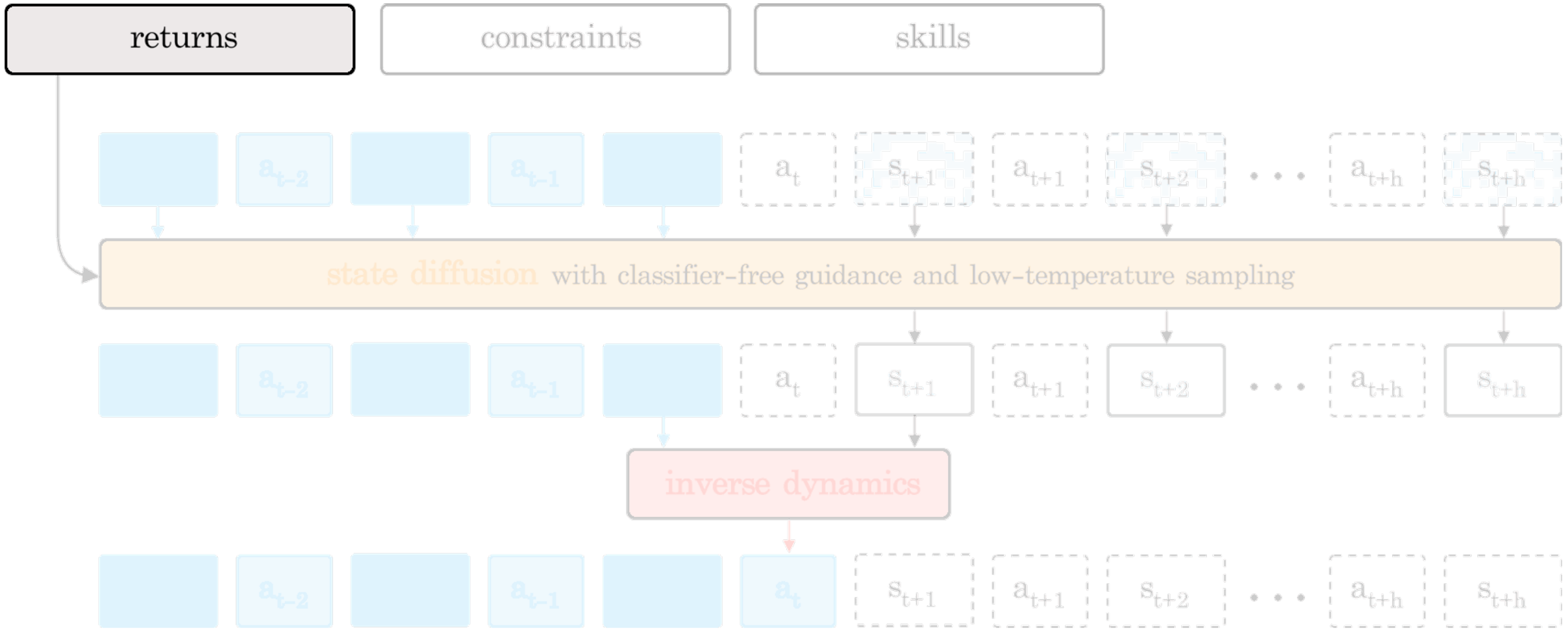


Figure. Diffusion model in Vision

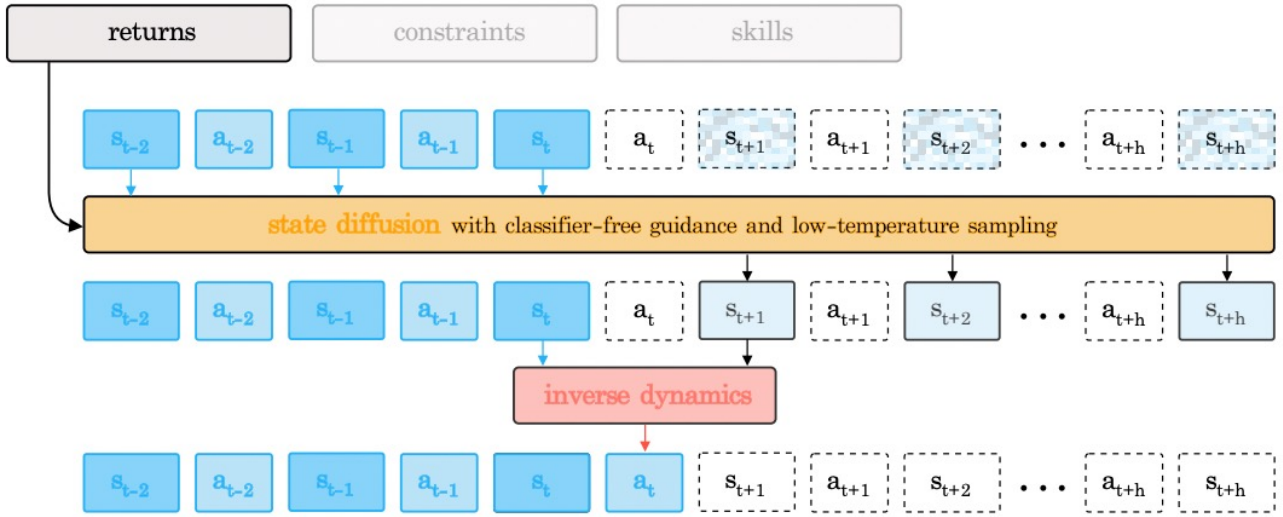


02 Sampling

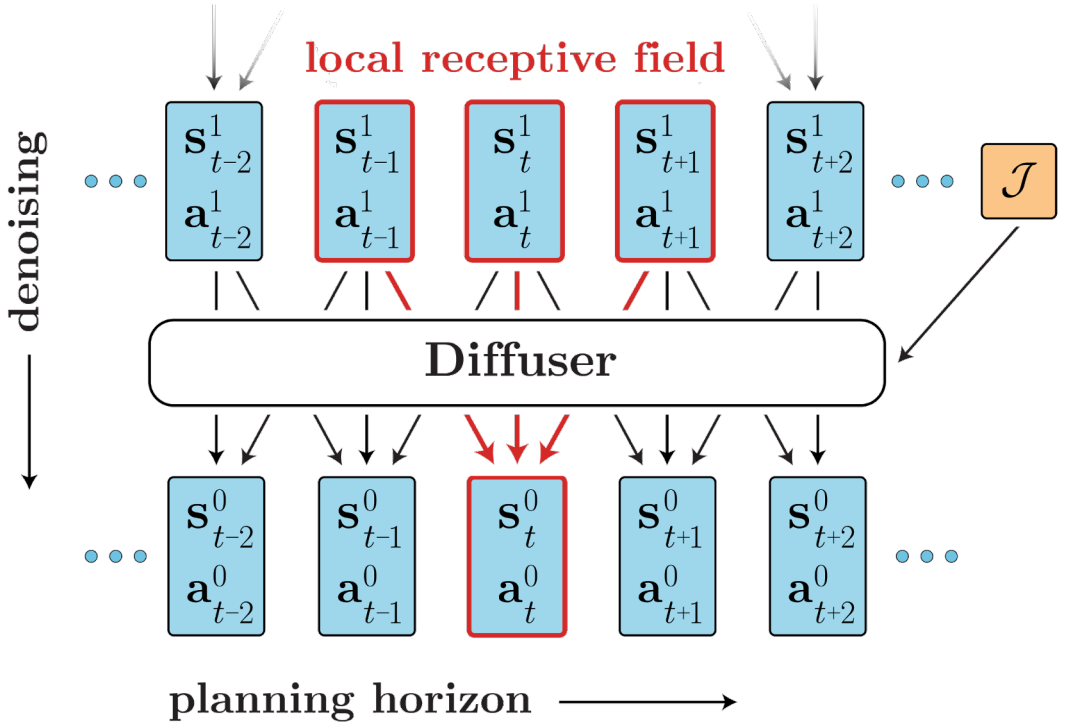
Sampling은 다음과 같이 할 수 있습니다.



02 Sampling



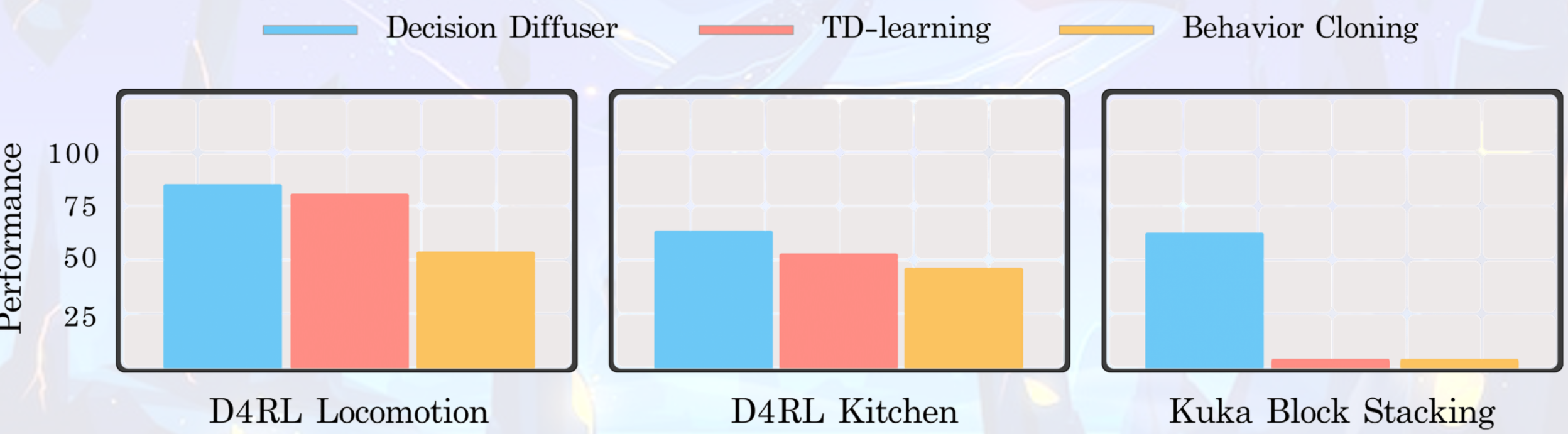
Decision diffuser



Diffuser planner

Compare with Offline RL

Decision Diffuser가 성능이 좋았다고 합니다.



Dataset	Environment	BC	CQL	IQL	DT	TT	MOReL	Diffuser	DD
Med-Expert	HalfCheetah	55.2	91.6	86.7	86.8	**95**	53.3	79.8	90.6 ±1.3
Med-Expert	Hopper	52.5	105.4	91.5	107.6	**110.0**	108.7	107.2	**111.8** ±1.8
Med-Expert	Walker2d	**107.5**	**108.8**	**109.6**	**108.1**	101.9	95.6	**108.4**	**108.8** ±1.7
Medium	HalfCheetah	42.6	44.0	47.4	42.6	46.9	42.1	44.2	**49.1** ±1.0
Medium	Hopper	52.9	58.5	66.3	67.6	61.1	**95.4**	58.5	79.3 ±3.6
Medium	Walker2d	75.3	72.5	78.3	74.0	79	77.8	79.7	**82.5** ±1.4
Med-Replay	HalfCheetah	36.6	**45.5**	**44.2**	36.6	41.9	40.2	42.2	39.3 ±4.1
Med-Replay	Hopper	18.1	95	94.7	82.7	91.5	93.6	96.8	**100** ±0.7
Med-Replay	Walker2d	26.0	77.2	73.9	66.6	**82.6**	49.8	61.2	75 ±4.3
Average		51.9	77.6	77	74.7	78.9	72.9	75.3	**81.8**
Mixed	Kitchen	51.5	52.4	51	-	-	-	-	**65** ±2.8
Partial	Kitchen	38	50.1	46.3	-	-	-	-	**57** ±2.5
Average		44.8	51.2	48.7	-	-	-	-	**61**

Table 1: **Offline Reinforcement Learning Performance.** We show that Decision Diffuser (DD) either matches or outperforms current offline RL approaches on D4RL tasks in terms of normalized average returns (Fu et al., 2020). We report the mean and the standard error over 5 random seeds.

04 IMPORTANCE OF LOW TEMPERATURE SAMPLING

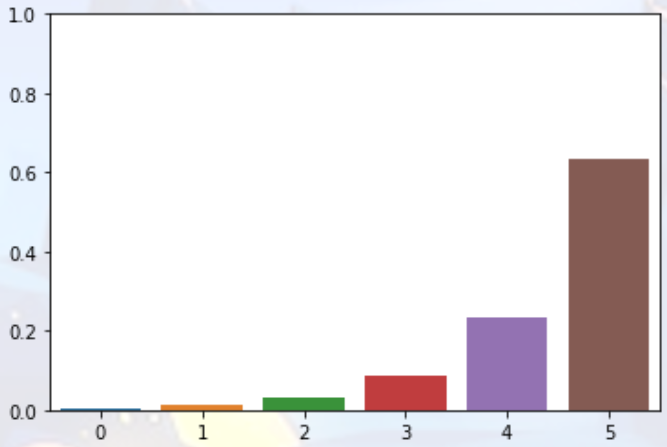
적당한 Temperature를 설정하는 것이 결과에 큰 영향을 줬다고 합니다.

Sampling : $x_{k-1} \sim N(\mu_{k-1}, \alpha\Sigma_{k-1})$

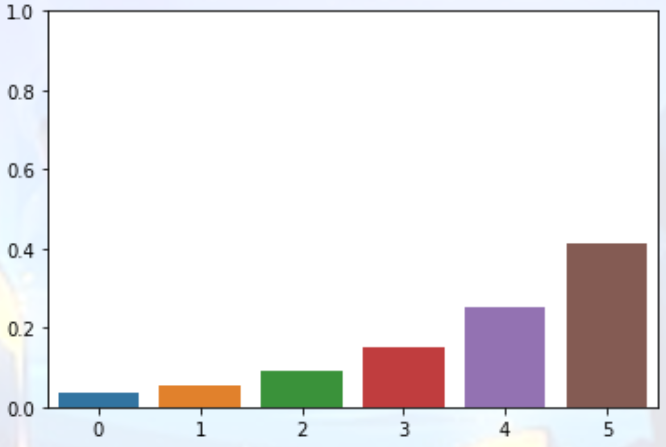
Results

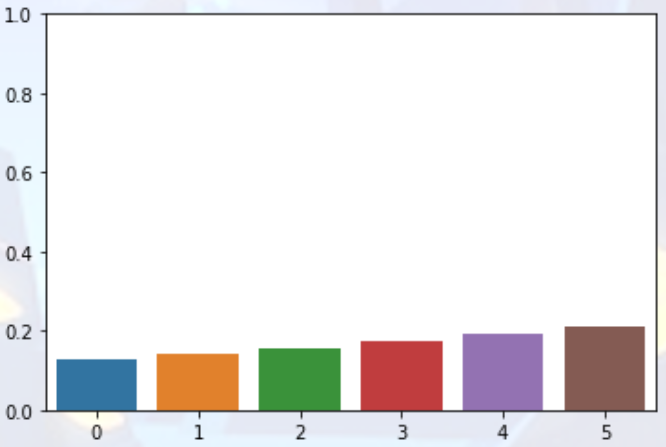
Decision Diffuser	Hopper-Medium-Expert
$\alpha = 0$	104.3 ± 0.7
$\alpha = 0.5$	**111.8** ± 1.6
$\alpha = 1.0$	107.1 ± 3.5

$$\text{softmax}(x)_i = \frac{e^{\frac{y_i}{T}}}{\sum_j^N e^{\frac{y_j}{T}}}$$



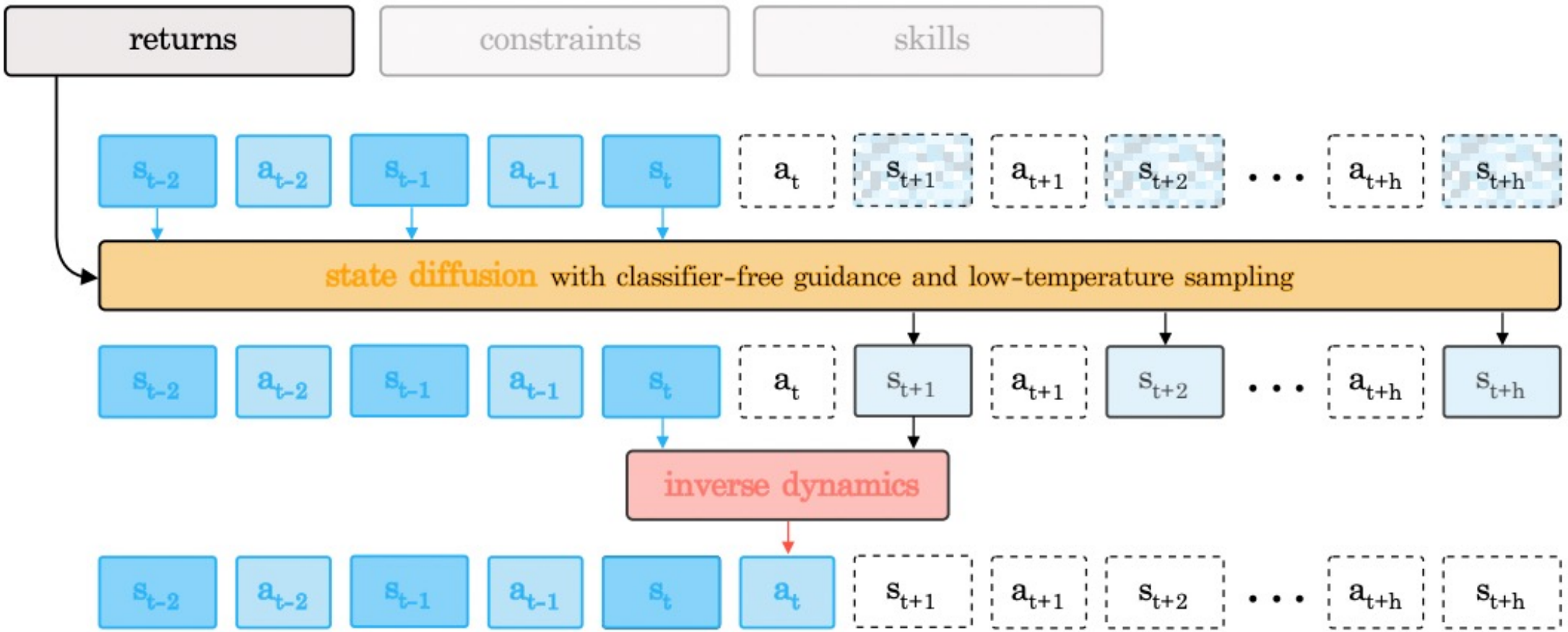
Low temperature





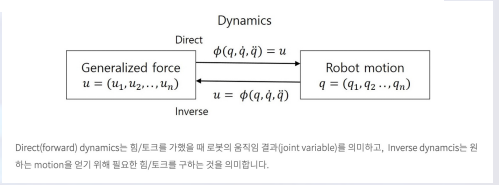
High temperature

04 INVERSE DYNAMICS

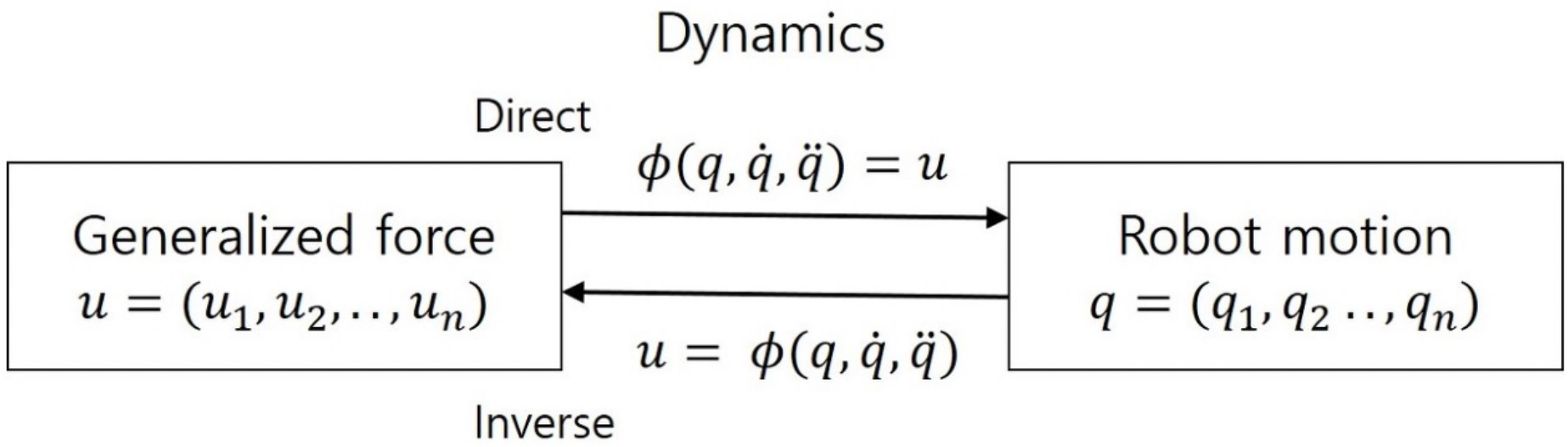


Inverse dynamics model : $a_t := f_\phi(s_t, s_{\{t+1\}})$

State Diffusion using Inverse dynamics



returns | constraints | skills



Direct(forward) dynamics는 힘/토크를 가했을 때 로봇의 움직임 결과(joint variable)를 의미하고, Inverse dynamcis는 원하는 motion을 얻기 위해 필요한 힘/토크를 구하는 것을 의미합니다.

State Diffusion using Inverse dynamics

서론

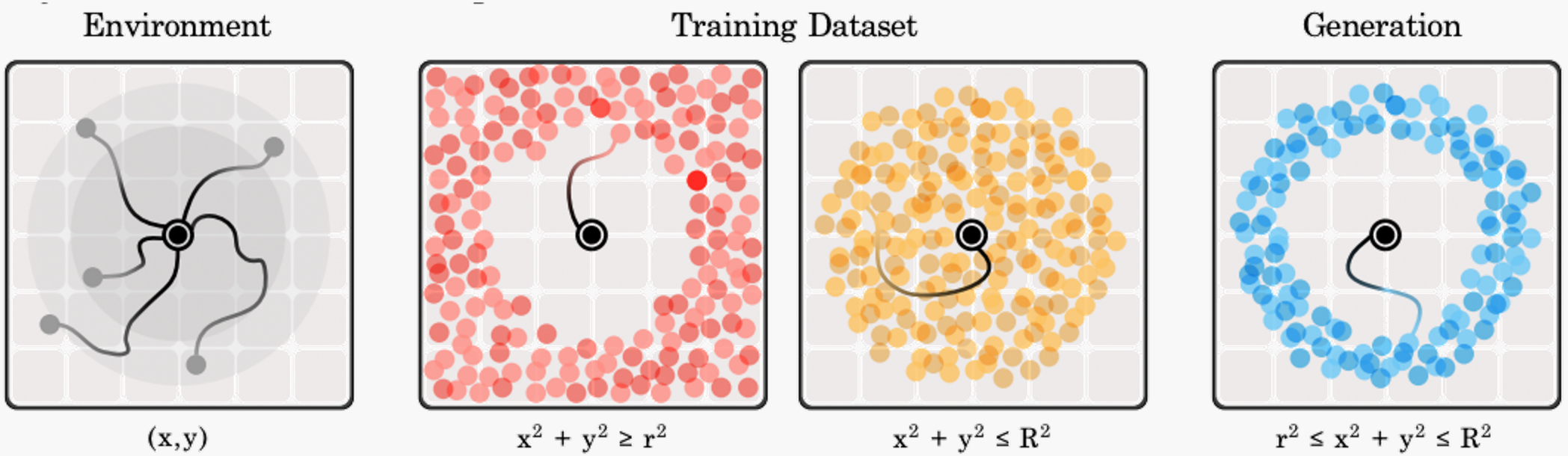
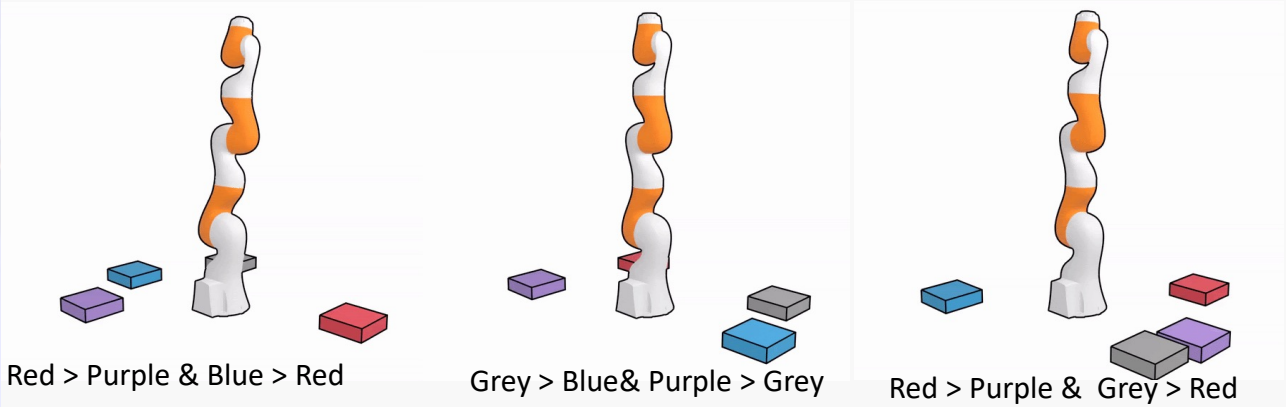


Figure 2: **Illustrative example.** We visualize the 2d robot navigation environment and the constraints satisfied by the trajectories in the dataset derived from the environment. We show the ability of the conditional diffusion model to generate trajectories that satisfy the combined constraints.

COMPOSING

Combining Stacking Constraints

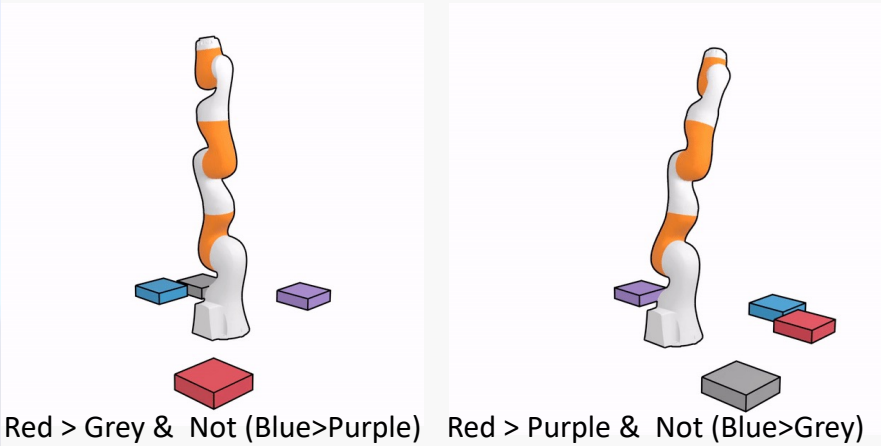


Red > Purple & Blue > Red

Grey > Blue& Purple > Grey

Red > Purple & Grey > Red

'NOT' constraints in Stacking and Rearrangement



Red > Grey & Not (Blue>Purple)

Red > Purple & Not (Blue>Grey)

$\epsilon_\theta(x_k(\tau), y(\tau), k) := \epsilon_\theta(x_k(\tau), \mathbb{I}(\tau \in C_i), k)$

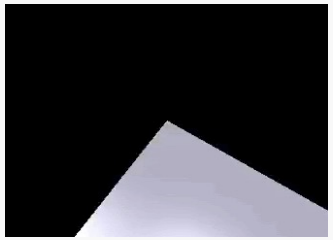
One hot encoding을 condition으로 줍니다.

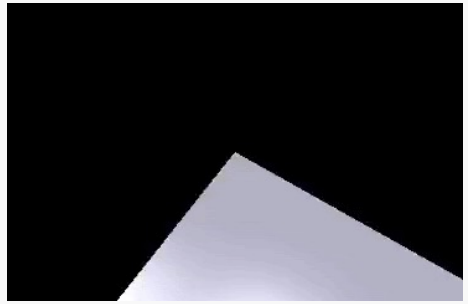
04 Skill Composition

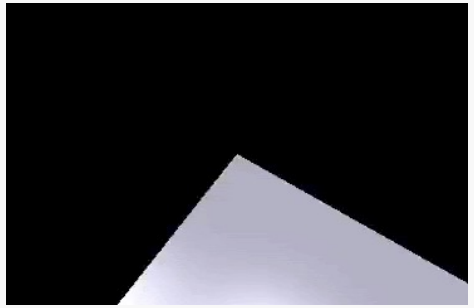
Decision diffuser에서는 여러가지 Skill을 결합할수 있습니다.

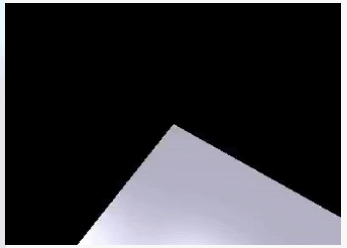
SKILL COMPOSITION

Pace

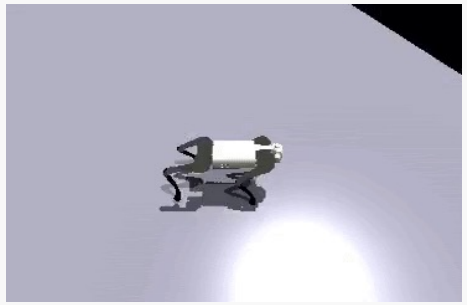


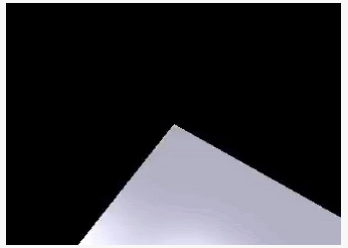






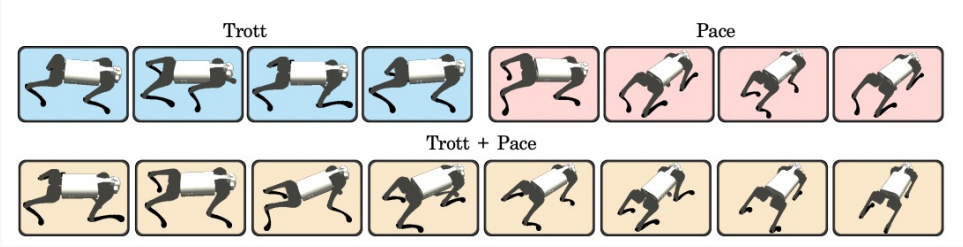
Trott

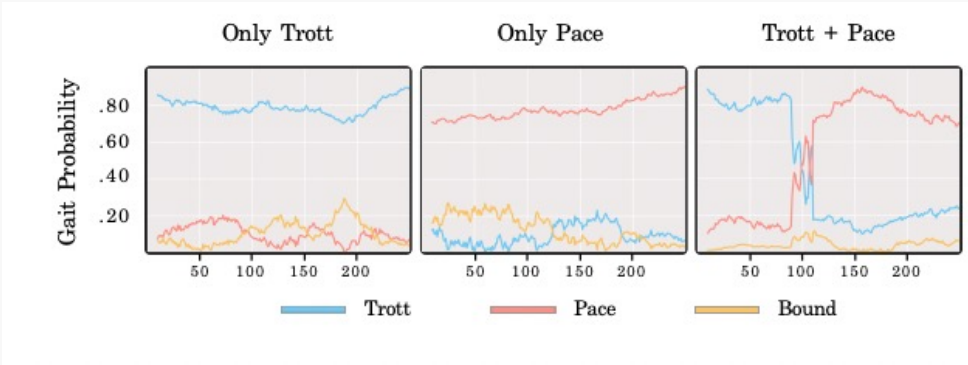




Bound

$$\hat{\epsilon} := \epsilon_\theta(\boldsymbol{x}_k(\tau), \emptyset, k) + \omega(\sum_{i \neq j}(\epsilon_\theta(\boldsymbol{x}_k(\tau), \boldsymbol{y}^i(\tau), k) - \epsilon_\theta(\boldsymbol{x}_k(\tau), \emptyset, k))$$
$$- (\epsilon_\theta(\boldsymbol{x}_k(\tau), \boldsymbol{y}^j(\tau), k) - \epsilon_\theta(\boldsymbol{x}_k(\tau), \emptyset, k)))$$





COMPOSING

Composing Skills, Constraints가 만들어지는 방법

$$q(\boldsymbol{x}_k(\tau)|\{\boldsymbol{y}^i(\tau)\}_{i=1}^n) \propto q(\boldsymbol{x}_k(\tau)) \prod_{i=1}^{n} \frac{q(\boldsymbol{x}_k(\tau)|\boldsymbol{y}^i(\tau))}{q(\boldsymbol{x}_k(\tau))} \quad \text{(Bayes Rule)}$$

$$\Rightarrow \log q(\boldsymbol{x}_k(\tau)|\{\boldsymbol{y}^i(\tau)\}_{i=1}^n) \propto \log q(\boldsymbol{x}_k(\tau)) + \sum_{i=1}^{n} (\log q(\boldsymbol{x}_k(\tau)|\boldsymbol{y}^i(\tau)) - \log q(\boldsymbol{x}_k(\tau)))$$

$$\Rightarrow \nabla_{\boldsymbol{x}_k(\tau)} \log q(\boldsymbol{x}_k(\tau)|\{\boldsymbol{y}^i(\tau)\}_{i=1}^n) = \nabla_{\boldsymbol{x}_k(\tau)} \log q(\boldsymbol{x}_k(\tau)) + \sum_{i=1}^{n} (\nabla_{\boldsymbol{x}_k(\tau)} \log q(\boldsymbol{x}_k(\tau)|\boldsymbol{y}^i(\tau)) - \nabla_{\boldsymbol{x}_k(\tau)} \log q(\boldsymbol{x}_k(\tau)))$$

$$\Rightarrow \epsilon_\theta(\boldsymbol{x}_k(\tau), \{\boldsymbol{y}^i(\tau)\}_{i=1}^n, k) = \epsilon_\theta(\boldsymbol{x}_k(\tau), \emptyset, k) + \sum_{i=1}^{n} (\epsilon_\theta(\boldsymbol{x}_k(\tau), \boldsymbol{y}^i(\tau), k) - \epsilon_\theta(\boldsymbol{x}_k(\tau), \emptyset, k))$$

결과적으로 나오는 식

$$\hat{\epsilon} := \epsilon_\theta(\boldsymbol{x}_k(\tau), \emptyset, k) + \omega(\epsilon_\theta(\boldsymbol{x}_k(\tau), \{\boldsymbol{y}^i(\tau)\}_{i=1}^n, k) - \epsilon_\theta(\boldsymbol{x}_k(\tau), \emptyset, k))$$

$$= \epsilon_\theta(\boldsymbol{x}_k(\tau), \emptyset, k) + \omega \sum_{i=1}^{n} (\epsilon_\theta(\boldsymbol{x}_k(\tau), \boldsymbol{y}^i(\tau), k) - \epsilon_\theta(\boldsymbol{x}_k(\tau), \emptyset, k))$$

Q-Function guided diffusion vs Classfier-free guided diffusion

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
    $\boldsymbol{\tau}^{i-1}_{\mathbf{s}_0} \leftarrow \boldsymbol{s}$
  Execute first action of plan $\boldsymbol{\tau}^0_{\mathbf{a}_0}$

Algorithm 1 Conditional Planning with the Decision Diffuser

Input: Noise model ϵ_θ, inverse dynamics f_ϕ, guidance scale ω, history length C, condition $\boldsymbol{y}$
Initialize $h \leftarrow \texttt{Queue}(\texttt{length} = C), t \leftarrow 0$ // Maintain a history of length C
while not done **do**
  Observe state s; $h.\texttt{insert}(s)$; Initialize $\boldsymbol{x}_K(\tau) \sim \mathcal{N}(0, \alpha I)$
  for $k = K \ldots 1$ **do**
    $\boldsymbol{x}_k(\tau)[:\texttt{length}(h)] \leftarrow h$ // Constrain plan to be consistent with history
    $\hat{\epsilon} \leftarrow \epsilon_\theta(\boldsymbol{x}_k(\tau), k) + \omega(\epsilon_\theta(\boldsymbol{x}_k(\tau), \boldsymbol{y}, k) - \epsilon_\theta(\boldsymbol{x}_k(\tau), k))$ // Classifier-free guidance
    $(\mu_{k-1}, \Sigma_{k-1}) \leftarrow \texttt{Denoise}(\boldsymbol{x}_k(\tau), \hat{\epsilon})$
    $\boldsymbol{x}_{k-1} \sim \mathcal{N}(\mu_{k-1}, \alpha\Sigma_{k-1})$
  end for
  Extract (s_t, s_{t+1}) from $x_0(\tau)$
  Execute $a_t = f_\phi(s_t, s_{t+1}); t \leftarrow t + 1$
end while

1. Train에서는 Classifier guided가 유리하지만, Testtime에서 Classifier free가 더 성능이 좋았다고 합니다.

2. Offline RL에서 Q-function은 Q-value를 높게 평가하는 경향이 있습니다. 대신에 Offline RL 세팅을 사용하는 diffuser에서 Q-function은 이런 error가 생길 수 있습니다. Classifier는 Q-function을 사용하지 않기에 이 문제가 발생하지 않습니다 .

Limatations of Decision Diffuser

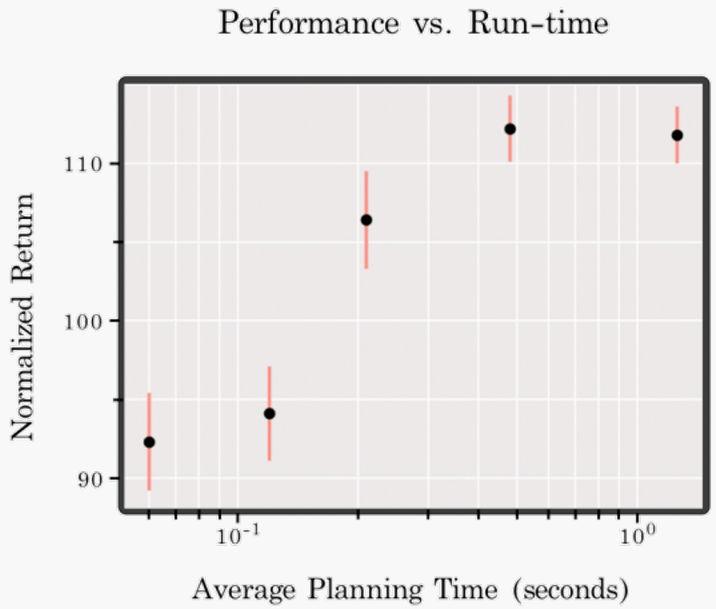


Figure A1: **Performance vs planning time.** We visualize the trade-off between performance, measured by normalized average return achieved in the environment, and planning time, measured in wall-clock time after warm-starting the reverse diffusion process.

5.4 Warm-Starting Diffusion for Faster Planning

A limitation of Diffuser is that individual plans are slow to generate (due to iterative generation). Naïvely, as we execute plans open loop, a new plan must be regenerated at each step of execution. To improve execution speed of Diffuser, we may further reuse previously generated plans to warm-start generations of subsequent plans.

To warm-start planning, we may run a limited number of forward diffusion steps from a previously generated plan and then run a corresponding number of denoising steps from this partially noised trajectory to regenerate an updated plan. In Figure 7, we illustrate the trade-off between performance and runtime budget as we vary the underlying number of denoising steps used to regenerate each a new plan from 2 to 100. We find that we may reduce the planning budget of our approach markedly with only modest drop in performance.

- Janner

run forward diffusion for a fixed number of steps (100 -> 40), and finally run the same number of reverse diffusion steps from the partially noised state sequence to generate another state sequence

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
    $\boldsymbol{\tau}^{i-1}_{\mathbf{s}_0} \leftarrow \boldsymbol{s}$
  Execute first action of plan $\boldsymbol{\tau}^0_{\mathbf{a}_0}$

05 한계점 (Limitations)

Limatations of Decision Diffuser

No partial observability (Offline RL)

Partial Observable environment에서는 offline RL에서 self-delusion 문제가 발생할 수 있습니다.

Inability to explore the environment and update itself in online setting (DT, DF 공통)

Online RL에서 사용하기에는 아직은 한계가 있습니다.

Experiments on only state-based environments

본 논문에서는 State 기반의 환경에서만 실행했습니다.

Only AND and NOT compositions are supported

Constraints나 Skill을 OR 조건으로 결합하는 것은 아직 못했다고 합니다.

Performance degradation in environments with stochastic dynamics → **Latent model**

Stochastic Dynamics를 갖는 환경에서는 성능이 떨어진다고 합니다.

Performance in limited data regime : Diffusion model처럼 Overfitting 문제가 발생합니다.

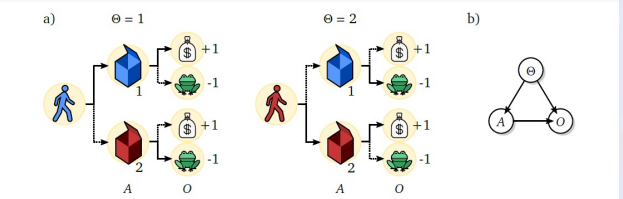


Figure 1 | The prize-or-frog problem. The objective is to choose the box containing the prize. Two problem instances exist: either the prize is in box 1 (Θ = 1) or box 2 (Θ = 2). There is also an expert who knows which box to open provided they are told the value of Θ. Panel (a) depicts the two configurations. Solid transitions are deterministic (taken with probability one) and dotted ones are possible transitions that aren't taken. Panel (b) shows the causal Bayesian network of the problem.