Diffusion Policy

0. Abstract

1. Diffusion policy : Visuomotor policy 를 conditional denoising diffusion process 로 표현 → 로봇 동작 생성
   - 시각(vision) 입력을 받아서 로봇의 motor action을 내놓는 policy 함수

⊕ DPM : 확산 확률 모델 (마코프 확산) (2015)
↓ +
DDPM : MSE (2020)
↓ +
DDIM : 비마코프 확산 (2021)
↓
Diffusion policy : DDPM 학습 + DDIM 추론 + Policy (C를 넣었을 때 x를 뱉는 함수 (확률 분포))

<샘플(행동) x_0 + 조건 C + 임의의 노이즈 단계 k> 인 때의 (임의 k ↓ $1, \dots, K_{train}$) → 모든 샘플마다 다름

실제 노이즈 ε 가 섞인 x_k 생성

↳ $x_k = \sqrt{\bar{\alpha}_k}\, x_0 + \sqrt{1-\bar{\alpha}_k}\, \varepsilon$ ($\sqrt{1-\bar{\alpha}_k} = \sigma_k$)

($\varepsilon \sim N(0, I)$ / $\bar{\alpha}_k$ = hyperparameter)

→ $\langle x_k, k, C \rangle$ 를 통해 $L = \|\varepsilon - \varepsilon_\theta(x_k, k, C)\|^2$ (MSE) 줄이도록 학습, 즉 ε가 무엇인지 학습

= 기울기 학습

↳ $\nabla_{x_k} \log P(x_k|C) = -\frac{1}{\sigma_k}\varepsilon_\theta(x_k, k, C)$ (θ = parameter)

⇒ denoising 직접 X, (x_k, k, C) 일 때 어디가 노이즈인지 규칙 학습

2. 행동 분포의 점수 함수 기울기 학습 (DDPM 학습) → 학습 과정

확률적 Langevin 동역학을 통해 기울기에 따라 x 반복 최적화 (DDIM 추론) (기울기는 학습 시 얻은 것 사용 / denoising 진행)

$x_{k-1} = x_k + \alpha_k \cdot S_\theta(x_k, k, C) + \sigma_k \xi_k$

x_k ↓ 행동시퀀스 (현재 반복 k의)

$S_\theta(x_k, k, C) = \nabla_{x_k} \log P(x_k|C) = -\frac{1}{\sigma_k}\varepsilon_\theta(x_k, k, C)$ (C: 조건)

→ k : 확산(denoising) step 인덱스 (→0으로 역진행 / step = denoising 반복 횟수) (K_{infer})

α_k : step 크기 (k마다 다름) = gradient 적용 정도

σ_k : 주입할 noise의 표준편차

(둘다 학습 시 사용한 $\bar{\alpha}_k$를 이용해서 공식에 의해 값 정해짐)

↳ $\alpha_k = \frac{\bar{\alpha}_{k-1} - \bar{\alpha}_k}{\sqrt{\bar{\alpha}_{k-1}\bar{\alpha}_k}}$, $\sigma_k = \sqrt{\frac{1-\bar{\alpha}_{k-1}}{1-\bar{\alpha}_k} \times \frac{\bar{\alpha}_{k-1} - \bar{\alpha}_k}{\bar{\alpha}_{k-1}}}$

ξ_k : step마다 뽑는 표준 gaussian noise ($N(0, I)$)

ε : 훈련 때 쓰는 forward noise

ξ : 추론 때 쓰는 backward noise

⇒

	학습	추론
입력)	x_k, k, C	x_k (이미 noise 섞여진 상태), k, C
	$x_k = \sqrt{\bar{\alpha}_k}x_0 + \sqrt{1-\bar{\alpha}_k}\,\varepsilon$	↓
	↓ MSE	$\hat{\varepsilon} = \varepsilon_\theta(x_k, k, C)$ 예측
	noise ε 맞추도록	

1. Introduction

1.

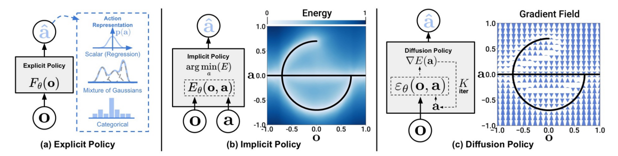


Figure 1. Policy Representations. a) Explicit policy with different types of action representations. b) Implicit policy learns an energy function conditioned on both action and observation and optimizes for actions that minimize the energy landscape c) Diffusion policy refines noise into actions via a learned gradient field. This formulation provides stable training, allows the learned policy to accurately model multimodal action distributions, and accommodates high-dimensional action sequences.

(a) Explicit policy

→ 관찰 o (조건 C) 가 들어오면 바로 행동 $\hat{a}$을 보내는 $F_\theta(o)$

(b) Implicit policy

→ 에너지 함수 $E_\theta(o, a)$ 학습

↓

주어진 o 에서 최소 $\hat{a}$ 찾기

(c) Diffusion policy

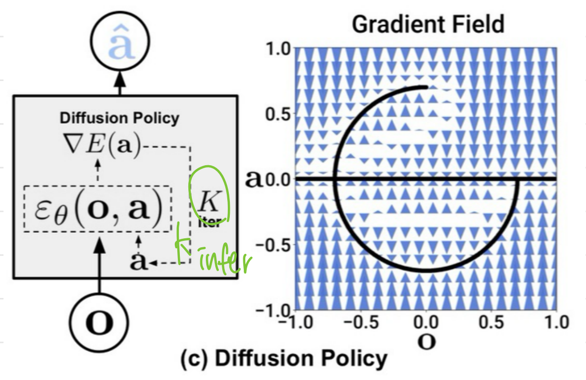
→ $\hat{a}$을 바로 뽑지 X

→ noise로 시작한 $\hat{a}$을 $\nabla E(a)$를 따라 K번 정제하도록 건너뛰면서 최종 행동 $\hat{a}$

(K_infer) denoising

전체: K_{train}

→ E 함수 자체 학습X 기울기를 학습 (기울기 = ε 이용)



2.
- 다중모달 행동분포의 표현 O
  - → 기울기 학습 + Langevin 추론 ⇒ 여러 정답 path O
- 고차원 출력 공간 (행동 sequence)
  - → diffusion은 고차원 공간, 즉 한번에 미래 seq 예측 O (시간 일관성 O)
- 안정적인 학습
  - → E 기반 정책 학습 (그림 b) : 정규화 상수 추정이 학습 불안정 ⇒ E 기울기 학습

3. Closed-loop action sequences
→ 한 번에 이만큼 예측하고, 그 중 앞의 몇 스텝만 실행 후 새로운 O_t(=C) = 이미지 로 재예측 ⇒ Receding Horizon
관찰 O_t에서 T_p 길이 예측 / T_p 中 T_a 길이만 사용 / 시점 $t+T_a$에서 새 관찰 O_{t+T_a}에서
sliding window (마스트든 고정) / 사용한 T_a 뒤부터 예측을 T_p 길이만큼 하고 또 앞 일부만 사용

Visual conditioning
→ O_t (관찰) = 이미지는 C(조건)로 하여 출력에 포함 X → 계산량↓ (행동만 denoising)
→ 이미지는 noise X, 단순 embedding 해서 조건으로 사용

Time-series diffusion transformer
→ vs CNN (시계열을 over-smoothing : 과도한 변화 뭉개기 (평활화))
→ token으로 보고 transformer

2. Diffusion Policy Formulation

2.1 Denoising Diffusion Probabilistic Models

1. (식1) $\mathbf{x}^{k-1} = \alpha(\mathbf{x}^k - \gamma\varepsilon_\theta(\mathbf{x}^k, k) + \mathcal{N}(0, \sigma^2 I))$: 역방향 (Denoising) ⇒ K_{infer} 만큼 반복함
누적계수 (1보다 약간↓, 안정성) / 예측 / 추론 시 추가되는 noise

(=) $x_{k-1} = x_k + \alpha_k \cdot S_\theta(x_k, k, c) + \sigma_k \xi_k$
$= \nabla_{x_k} \log P(x_k|c)$
$= -\frac{1}{\sigma_k}\varepsilon_\theta(x_k, k, c)$
$= x_k - \frac{\alpha_k}{\sigma_k}\varepsilon_\theta(x_k, k, c) + \sigma_k \xi_k$
$= \gamma$ ($\xi_k \sim N(0, I)$)

$\sim N(0, \sigma^2 I)$: 추론 시 더해지는 noise ← "아주 작은 미세한 noise" ∵ MSE

↳ "Stochastic Langevin Dynamics"
: 잡음 제거 과정 (생성모델)
가우시안 잡음 샘플된 x_K —(K_{infer}번 잡음제거)→ x_0 (무잡음)

↳ denoising 결과가 평균 이므로 예측하지 않은 해의 해를 막기위해 추가 (평균 주변의 해 선정)

↳ step 거듭될수록 감소하므로 상관X + 평균=0인 gaussian noise
(∵ 수축 계수) 이므로 방향 모멸 X

⇓ 더해지지 않은 식

(식2) $\mathbf{x}' = \mathbf{x} - \gamma\nabla E(\mathbf{x})$

2. hyper parameter

→ 학습: $\overline{\alpha}_k$ (noise scheduler), K_{train}

추론: K_{infer} (, 스킵 계수 α)

2.2 DDPM Training

1. 손실함수: $\mathcal{L} = MSE(\varepsilon^k, \varepsilon_\theta(\mathbf{x}^0 + \varepsilon^k, k))$ (식3)

실제 노이즈 ↓ ↳ 예측 노이즈

→ denoising 과정 샘플 분포 $q(x_0)$ 와 원본 데이터 분포 $p(x_0)$ 사이의 KL 발산의 변분 하한 최소화

2.3 Diffusion for Visumotor Policy Learning

1. 기존 DDPM의 x = 이미지 (이미지 생성)

↓

로봇의 Visuomotor : (x = 로봇 행동, O_t (관찰)을 C(조건)로 하여 denoising

2.

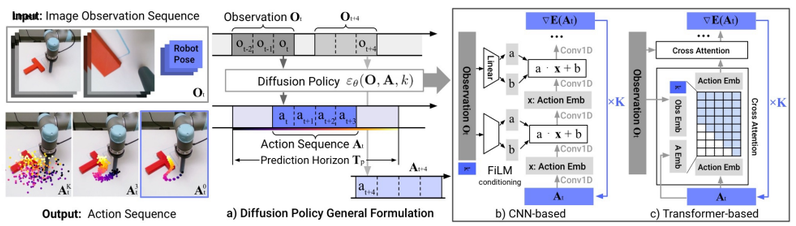


Figure 2. Diffusion Policy Overview a) General formulation. At time step t, the policy takes the latest T_o steps of observation data O_t as input and outputs T_a steps of actions A_t. b) In the CNN-based Diffusion Policy, FiLM (Feature-wise Linear Modulation) Perez et al. (2018) conditioning of the observation feature O_t is applied to every convolution layer, channel-wise. Starting from $\mathbf{A}_t^K$ drawn from Gaussian noise, the output of noise-prediction network ε_θ is subtracted, repeating K times to get $\mathbf{A}_t^0$, the denoised action sequence. c) In the Transformer-based Vaswani et al. (2017) Diffusion Policy, the embedding of observation $\mathbf{O}_t$ is passed into a multi-head cross-attention layer of each transformer decoder block. Each action embedding is constrained to only attend to itself and previous action embeddings (causal attention) using the attention mask illustrated.

<Diffusion Policy>

(a) 일반 정식화

→ 시점 t 에서 T_o 길이의 관찰 데이터 (조건화된 이미지 임베딩) O_t 를 입력으로 받아서 T_a 길이의 행동 A_t 예측

(b) CNN 기반

→ O_t 특징의 FiLM (Feature-wise Linear Modulation) 를 조건화하고 모든 Conv 층의 channel을 적용

→ gaussian noise 행동 A_t^k

↓

noise network ε_θ을 K_{infer} 반복마다 예측

↓

denoised 행동 A_t^0

동일

(c) Transformer 기반

→ O_t의 임베딩을 CNN(over-smoothing) 대신

각 transformer decoder 블록의 multi-head

cross-attention으로 전달

↓

attention mask → causal attention

: 자신과 이전행동 임베딩만

attention

3. "Closed-loop action-sequence prediction"

→ 시점 t에서 T_o 길이의 관찰 데이터 (조건화된 이미지 임베딩) O_t를

입력으로 받아서 T_p 길이의 행동 A_t 예측

↳ 사용은 앞의 T_a 길이 행동만 사용

(T_o: 관찰 지평, T_p: 행동 예측 지평, T_a: 행동 실행 지평)

→ 다음 추론 실행이 warming-start (매끄러움)

4. "Visual observation conditioning"

→ 결합 분포 $P(A_t, O_t)$ 대신 조건부 분포 $P(A_t|O_t)$를 근사

→ 관찰 측은 비용X, 조건화 → 추론 속도↑, 실시간 제어↑, Vision encoder의 end-to-end 학습 가능

→ 변경

식(1) → $\mathbf{A}_t^{k-1} = \alpha(\mathbf{A}_t^k - \gamma\varepsilon_\theta(\mathbf{O}_t, \mathbf{A}_t^k, k) + \mathcal{N}(0, \sigma^2 I))$ ← (식) $\mathbf{x}^{k-1} = \alpha(\mathbf{x}^k - \gamma\varepsilon_\theta(\mathbf{x}^k, k) + \mathcal{N}(0, \sigma^2 I))$

식(3)(Loss) → $\mathscr{L} = MSE(\varepsilon^k, \varepsilon_\theta(\mathbf{O}_t, \mathbf{A}_t^0 + \varepsilon^k, k))$ ← (식3) $\mathscr{L} = MSE(\varepsilon^k, \varepsilon_\theta(\mathbf{x}^0 + \varepsilon^k, k))$

3. Key Design Decisions

3.1 Network Architecture Options

1. noise network ε_θ의 CNN vs Transformer

↳ CNN-based Diffusion Policy

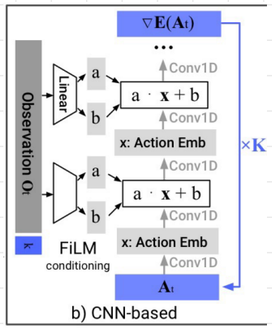
→ 1D 시계열 CNN

→ $P(A_t|O_t)$ 로 조건부 분포 (↔ 결합분포: $P(A_t, O_t)$)

→ 출력: 행동만 (관찰은 X; 조건화)

⇒ 행동 seq 급격하게 변화 = 고주파: 성능 저조 (over-smoothing)

(∵ 시계열 CNN의 귀납적 편향이 저주파 신호 선호)



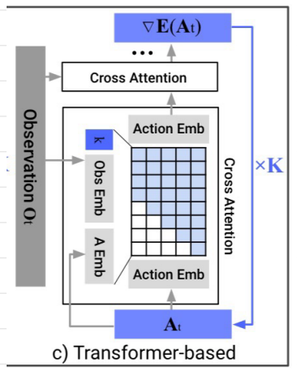
↳ Time-series diffusion transformer

→ CNN의 over-smoothing ⇒ "Transformer"

→ O_t: feature 추출 (Vision Encoder) + 임베딩 sequence (shared MLP)

K_{infer} 정보에 대한 사인파 임베딩 + A_t^K 행동 "token" ⟶ transformer decoder ⇒ $\varepsilon_\theta(O_t, A_t^K, K)$ (기울기 = ∇)

→ CNN보다 hyperparameter에 더 민감



3.2 Visual Encoder

1. 각 시점 이미지 encoding 후 모두 concatenate → 최종 O_t
2. Encoder: Resnet 18 (사전학습 X)
   → average pooling 대신 spatial softmax pooling (공간 정보 유지)
   → Batch Norm → Group Norm으로 교체
   DDPM에서 사용되는 EMA / 정규화 층과 매칭 ↑

3.3 Noise schedule

1. 제곱 코사인 스케줄

3.4 Accelerating Inference for Real-time Control

1. DDIM으로 파우프 실시간 제어를 위한 빠른 추론 속도

→ K_{train} ↑: 훈련이 배우고
K_{infer} ↓: 빠른 추론O

4. Intriguing Properties of Diffusion Policy

4.1 Model Multi-Modal Action Distributions

1. 다중 모달성 보장 : Stochastic sampling 결과 + Stochastic 초기화

= 확률적 경사하강의 $\nabla_k E_k$ 항을 통해 다양성 = A_t^K의 noise $\sim N(0,I)$ 의 랜덤 초기화

2. Push-T 예제의 다중 모달 행동

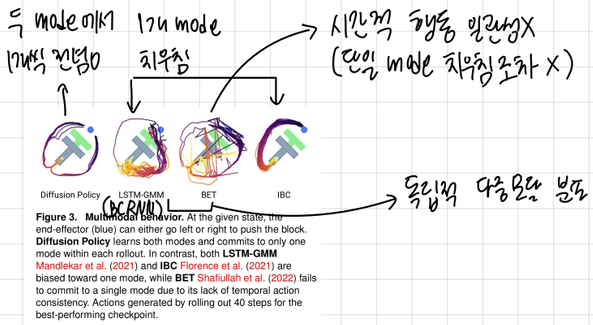


Figure 3. Multimodal behavior. At the given state, the end-effector (blue) can either go left or right to push the block. **Diffusion Policy** learns both modes and commits to only one mode within each rollout. In contrast, both **LSTM-GMM** Mandlekar et al. (2021) and **IBC** Florence et al. (2021) are biased toward one mode, while **BET** Shafiullah et al. (2022) fails to commit to a single mode due to its lack of temporal action consistency. Actions generated by rolling out 40 steps for the best-performing checkpoint.

→ 왼/오 고만 일관 O

독립적 다중모달 분포 : 샘플 1개에 대해 한 번의 seq 예측X, 매 t 마다 mode 변경

4.2 Synergy with Position Control

1. 행동 공간을 Velocity control X
Position control O (성능↑)

→ 다중 모달성 더 ↑

→ Compounding error (누적 오류) 가 더 ↓

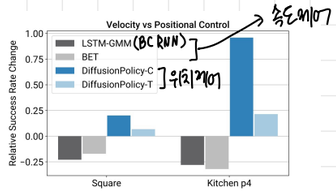


Figure 4. Velocity v.s. Position Control. The performance difference when switching from velocity to position control. While both BCRNN and BET performance decrease, Diffusion Policy is able to leverage the advantage of position and improve its performance.

4.3 Benefits of Action-sequence Prediction

1. 다른 policy 학습 방식: 고차원 출력에서 샘플링 잘 X

   → IBC, BCRNN, BET

   ↕

2. DDPM: 행동 seq 예측에 사용O (고차원 출력)

   - 시간적 행동 일관성 (**4.1.2** 참고)
   - Idle actions에 대한 강건성: 과적합X
     - ↳ 일시정지일때의 동일한 위치 행동의 seq 와 속도0의 seq의 data가 과적합

4.4 Training Stability

1. IBC (Implicit policy): 이론적으로 diffusion policy와 유사

   → but, "학습 불안정성"

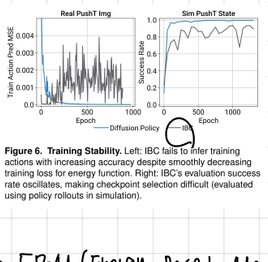


Figure 6. Training Stability. Left: IBC fails to infer training actions with increasing accuracy despite smoothly decreasing training loss for energy function. Right: IBC's evaluation success rate oscillates, making checkpoint selection difficult (evaluated using policy rollouts in simulation).

→ EBM (Energy-Based-Model) 로 행동 분포 P 표현

$$p_\theta(\mathbf{a}|\mathbf{o}) = \frac{e^{-E_\theta(\mathbf{o},\mathbf{a})}}{Z(\mathbf{o},\theta)}$$ ← 정규화 상수 : 추정 어려움

→ 손실함수 : 음의 log-likelihood

$$\mathscr{L}_{infoNCE} = -\log\left(\frac{e^{-E_\theta(\mathbf{o},\mathbf{a})}}{e^{-E_\theta(\mathbf{o},\mathbf{a})} + \sum_{j=1}^{N_{neg}} e^{-E_\theta(\mathbf{o},\tilde{\mathbf{a}}^j)}}\right)$$

정규화 상수 추정을 위한 negative sample set $\{\tilde{\mathbf{a}}^j\}_{j=1}^{N_{neg}}$ 사용 → 복잡학습성 (학습 불안정)

$$\nabla_{\mathbf{a}} \log p(\mathbf{a}|\mathbf{o}) = -\nabla_{\mathbf{a}} E_\theta(\mathbf{a},\mathbf{o}) - \underbrace{\nabla_{\mathbf{a}} \log Z(\mathbf{o},\theta)}_{=0} \approx -\varepsilon_\theta(\mathbf{a},\mathbf{o})$$

: Diffusion Policy DDPM

↳ $Z(\mathbf{o},\theta)$를 경사함수 = ∇를 모델링함으로써 우회

4.5 Connections to Control Theory

1. Linear 상태 공간: $\mathbf{s}_{t+1} = \mathbf{A}\mathbf{s}_t + \mathbf{B}\mathbf{a}_t + \mathbf{w}_t, \quad \mathbf{w}_t \sim \mathscr{N}(0, \Sigma_w).$ ($\mathbf{a}_t$: 행동)

일 때, (T_p: 예측길이)

- $T_p = 1$ (한 step): $\mathscr{L} = MSE(\varepsilon^k, \varepsilon_\theta(\mathbf{s}_t, -\mathbf{K}\mathbf{s}_t + \varepsilon^k, k))$ / $\varepsilon_\theta(\mathbf{s},\mathbf{a},k) = \frac{1}{\sigma_k}[\mathbf{a} + \mathbf{K}\mathbf{s}]$ / DDIM: $\mathbf{a} = -\mathbf{K}\mathbf{s}$ 에서 결정적 특징
  - ↳ $\sigma_k = 0$
  - K_{infer}의 불안정
- $T_p > 1$ (궤적): $\mathbf{a}_{t:t'} = -\mathbf{K}(\mathbf{A} - \mathbf{B}\mathbf{K})^{t'}\mathbf{s}_t$ DDM

2. 저의 예상이나 policy 中 어느것도 non-linear 이면 미래 행동 예측 X

5. Evaluation

5.1 Simulation Environments and dataset

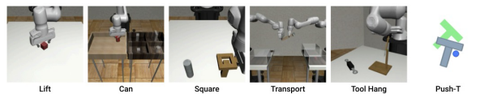
1. Robomimic benchmark 정리

로봇 수 객체수 행동 dim 숙련 인간 다름 인간

Task	# Rob	# Obj	ActD	#PH	#MH	Steps	Img?	HiPrec
Simulation Benchmark								
Lift	1	1	7	200	300	400	Yes	No
Can	1	1	7	200	300	400	Yes	No
Square	1	1	7	200	300	400	Yes	Yes
Transport	2	3	14	200	300	700	Yes	No
ToolHang	1	2	7	200	0	700	Yes	Yes
Push-T	1	1	2	200	0	300	Yes	Yes
BlockPush	1	2	2	0	0	350	No	No
Kitchen	1	7	9	656	0	280	No	No
Realworld Benchmark								
Push-T	1	1	2	136	0	600	Yes	Yes
6DoF Pour	1	liquid	6	90	0	600	Yes	No
Peri Spread	1	liquid	6	90	0	600	Yes	No
Mug Flip	1	1	7	250	0	600	Yes	No

Table 3. Tasks Summary. # Rob: number of robots, #Obj: number of objects, ActD: action dimension, PH: proficient-human demonstration, MH: multi-human demonstration, Steps: max number of rollout steps, HiPrec: whether the task has a high precision requirement. BlockPush uses 1000 episodes of scripted demonstrations.

5.2 Evaluation Methodology

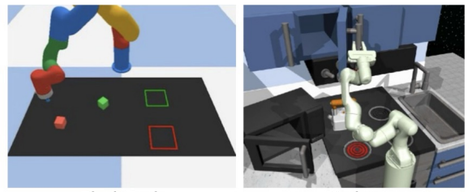


	Lift		Can		Square		Transport		ToolHang	Push-T
	ph	mh	ph	mh	ph	mh	ph	mh	ph	ph
LSTM-GMM	**1.00**/0.96	**1.00**/0.93	**1.00**/0.91	**1.00**/0.81	0.95/0.73	0.86/0.59	0.76/0.47	0.62/0.20	0.67/0.31	0.67/0.61
IBC	0.79/0.41	0.15/0.02	0.00/0.00	0.01/0.01	0.00/0.00	0.00/0.00	0.00/0.00	0.00/0.00	0.00/0.00	0.90/0.84
BET	**1.00**/0.96	**1.00**/0.99	**1.00**/0.89	**1.00**/0.90	0.76/0.52	0.68/0.43	0.38/0.14	0.21/0.06	0.58/0.20	0.79/0.70
DiffusionPolicy-C	**1.00**/0.98	**1.00**/0.97	**1.00**/0.96	**1.00/0.96**	**1.00/0.93**	**0.97/0.82**	0.94/0.82	**0.68/0.46**	0.50/0.30	0.95/**0.91**
DiffusionPolicy-T	**1.00/1.00**	**1.00/1.00**	**1.00/1.00**	**1.00**/0.94	**1.00**/0.89	0.95/0.81	**1.00/0.84**	0.62/0.35	**1.00/0.87**	**0.95**/0.79

Table 1. Behavior Cloning Benchmark (State Policy) We present success rates with different checkpoint selection methods in the format of (max performance) / (average of last 10 checkpoints), with each averaged across 3 training seeds and 50 different environment initial conditions (150 in total). LSTM-GMM corresponds to BC-RNN in RoboMimicMandlekar et al. (2021), which we reproduced and obtained slightly better results than the original paper. Our results show that Diffusion Policy significantly improves state-of-the-art performance across the board.

	Lift		Can		Square		Transport		ToolHang	Push-T
	ph	mh	ph	mh	ph	mh	ph	mh	ph	ph
LSTM-GMM	**1.00**/0.96	**1.00**/0.95	**1.00**/0.88	0.98/0.90	0.82/0.59	0.64/0.38	0.88/0.62	0.44/0.24	0.68/0.49	0.69/0.54
IBC	0.94/0.73	0.39/0.05	0.08/0.01	0.00/0.00	0.03/0.00	0.00/0.00	0.00/0.00	0.00/0.00	0.00/0.00	0.75/0.64
DiffusionPolicy-C	**1.00/1.00**	**1.00/1.00**	**1.00**/0.97	**1.00**/0.96	0.98/**0.92**	**0.98/0.84**	**1.00/0.93**	**0.89/0.69**	**0.95/0.73**	**0.91/0.84**
DiffusionPolicy-T	**1.00/1.00**	**1.00**/0.99	**1.00/0.98**	**1.00/0.98**	**1.00**/0.90	0.94/0.80	0.98/0.81	0.73/0.50	0.76/0.47	0.78/0.66

Table 2. Behavior Cloning Benchmark (Visual Policy) Performance are reported in the same format as in Tab 1. LSTM-GMM numbers were reproduced to get a complete evaluation in addition to the best checkpoint performance reported. Diffusion Policy shows consistent performance improvement, especially for complex tasks like Transport and ToolHang.



	BlockPush		Kitchen			
	p1	p2	p1	p2	p3	p4
LSTM-GMM	0.03	0.01	**1.00**	0.90	0.74	0.34
IBC	0.01	0.00	0.99	0.87	0.61	0.24
BET	0.96	0.71	0.99	0.93	0.71	0.44
DiffusionPolicy-C	0.36	0.11	**1.00**	**1.00**	**1.00**	**0.99**
DiffusionPolicy-T	**0.99**	**0.94**	**1.00**	0.99	0.99	0.96

Table 4. Multi-Stage Tasks (State Observation). For PushBlock, px is the frequency of pushing x blocks into the targets. For Kitchen, px is the frequency of interacting with x or more objects (e.g. bottom burner). Diffusion Policy performs better, especially for difficult metrics such as $p2$ for Block Pushing and $p4$ for Kitchen, as demonstrated by our results.

5.3 key Findings

1. Diffusion policy can express short-horizon multimodality (4.1.2)
   예측 길이

2. Diffusion policy can express long-horizon multimodality

3. Diffusion policy can better leverage position control (4.2.1)

4. The tradeoff in action horizon
   → 4.3 but, horizon (예측 길이)
   ↑ : 성능↓ (trade off)
   → 8 steps 적절

5. Robustness against latency
   → ∵ Receding Horizon (1.3)

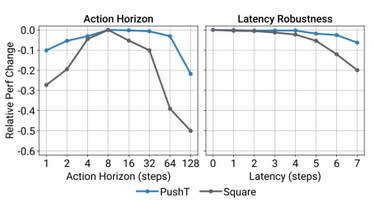


Figure 5. Diffusion Policy Ablation Study. Change (difference) in success rate relative to the maximum for each task is shown on the Y-axis. **Left**: trade-off between temporal consistency and responsiveness when selecting the action horizon. **Right**: Diffusion Policy with position control is robust against latency. Latency is defined as the number of steps between the last frame of observations to the first action that can be executed.

6. Diffusion policy is stable to train (4.4.1)

5.4 Ablation Study

1. Visual Encoder

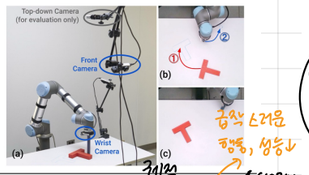
Archicture & Prertain Datset	From Scatch	Pretrained frozen	Pretrained finetuning
Resnet18 (in21)	0.94	0.58	0.92
Resnet34 (in21)	0.92	0.40	0.94
ViT-base (clip)	0.22	0.70	0.98

Table 5. Vision Encoder Comparison All models are trained on the robomimic square (ph) task using CNN-based diffusion policy. Each model is trained for 500 epochs and evaluated every 50 epochs under 50 different environment initial conditions.

6. Realworld Evaluation

6.1 Realworld Push-T Task

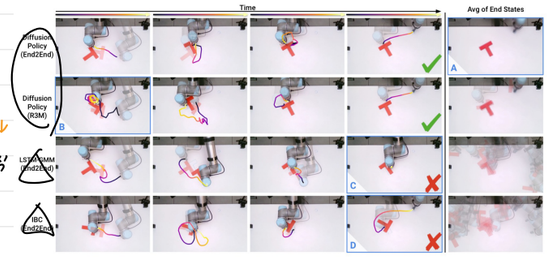
1. End-to-End vs Pre-trained vision encoders



	Human Demo	IBC pos	IBC vel	LSTM-GMM pos	LSTM-GMM vel	Diffusion Policy E2E	Diffusion Policy ImgNet	Diffusion Policy R3M	Diffusion Policy E2E
IoU	0.84	0.14	0.19	0.24	0.25	0.53	0.74	0.66	**0.80**
Succ%	1.00	0.00	0.00	0.20	0.10	0.61	0.15	0.39	**0.95**
Dur.	20.3	56.3	41.6	47.3	51.7	57.5	55.8	31.7	**22.9**

Table 6. Realworld Push-T Experiment. a) Hardware setup. b) Illustration of the task. The robot needs to ① precisely push the T-shaped block into the target region, **and** ② move the end-effector to the end-zone. c) The ground truth end state used to calculate IoU metrics used in this table. Table: Success is defined by the end-state IoU greater than the minimum IoU in the demonstration dataset. Average episode duration presented in seconds. T-E2E stands for end-to-end trained Transformer-based Diffusion Policy

End-to-End < R3M but, R3M : Jittery



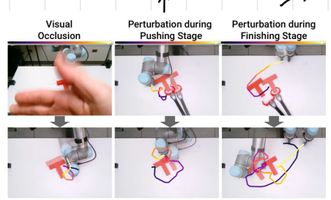
→ Jittery 함

Figure 7. Realworld Push-T Comparisons. Columns 1-4 show action trajectories based on key events. The last column shows averaged images of the end state. **A**: Diffusion policy (End2End) achieves more accurate and consistent end states. **B**: Diffusion Policy (R3M) gets stuck initially but later recovers and finishes the task. **C**: LSTM-GMM fails to reach the end zone while adjusting the T block, blocking the eval camera view. **D**: IBC prematurely ends the pushing stage.

2. Robustness against Perturbation

이동중 위치교정

end-zone 으로 갈때 위치 조정

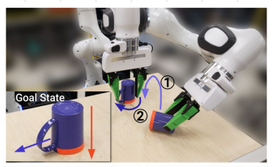


손 흔들기로 가리기

Figure 8. Robustness Test for Diffusion Policy. Left: A waving hand in front of the camera for 3 seconds causes slight jitter, but the predicted actions still function as expected. **Middle**: Diffusion Policy immediately corrects shifted block position to the goal state during the pushing stage. **Right**: Policy immediately aborts heading to the end zone, returning the block to goal state upon detecting block shift. This novel behavior was never demonstrated. Please check the videos in the supplementary material.

6.2 Mug Flipping Task

1.

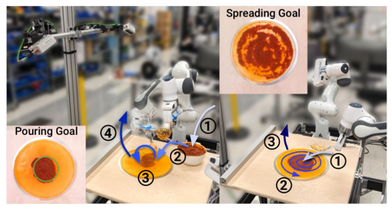


	Human	LSTM-GMM	Diffusion Policy
Succ %	1.0	0.0	0.9

Figure 9. 6DoF Mug Flipping Task. The robot needs to ① Pickup a randomly placed mug and place it lip down (marked orange). ② Rotate the mug such that its handle is pointing left.

6.3 Sauce Pouring and Spreading

1.



	Pour		Spread	
	IoU	Succ	Coverage	Succ %
Human	0.79	1.00	0.79	1.00
LSTM-GMM	0.06	0.00	0.27	0.00
Diffusion Policy	**0.74**	**0.79**	**0.77**	**1.00**

Figure 10. Realworld Sauce Manipulation. [Left] **6DoF pouring Task.** The robot needs to ① dip the ladle to scoop sauce from the bowl, ② approach the center of the pizza dough, ③ pour sauce, and ④ lift the ladle to finish the task. [Right] **Periodic spreading Task** The robot needs to ① approach the center of the sauce with a grasped spoon, ② spread the sauce to cover pizza in a spiral pattern, and ③ lift the spoon to finish the task.

7. Realworld Bimanual Tasks

1. Bimanual : 양손

7.1 observation and Action spaces

7.2 Teleoperation

7.3 Bimanual Egg Beater

1.

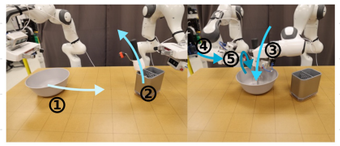


Figure 11. Bimanual Egg Beater Manipulation. The robot needs to ① push the bowl into position (only if too close to the left arm), ② approach and pick up the egg beater with the right arm, ③ place the egg beater in the bowl, ④ approach and grasp the egg beater crank handle, and ⑤ turn the crank handle 3 or more times.

7.4 Bimanual Mat Unrolling

1.

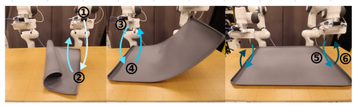


Figure 12. Bimanual Mat Unrolling. The robot needs to ① pick up one side of the mat (if needed), using the left or right arm, ② lift and unroll the mat (if needed), ③ ensure that both sides of the mat are grasped, ④ lift the mat, ⑤ place the mat oriented with the table, mostly centered, and ⑥ release the mat.

7.5 Bimanual Shirt Folding

1.

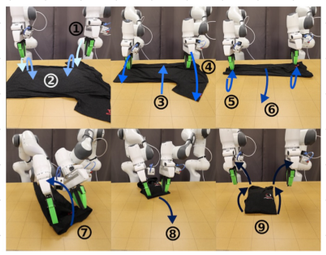


Figure 13. Bimanual Shirt Folding. The robot needs to ① approach and grasp the closest sleeve with both arms, ② fold the sleeve and release, ③ drag the shirt closer (if needed), ④ approach and grasp the other sleeve with both arms, ⑤ fold the sleeve and release, ⑥ drag the shirt to a orientation for folding, ⑦ grasp and fold the shirt in half by its collar, ⑧ drag the shirt to the center, and ⑨ smooth out the shirt and move the arms away.

8. Related Work

1. Explicit policy
2. Implicit policy
3. diffusion models

9. Limitations and Future Work

1. LSTM-GMM 보다 계산비용·속도 저면 ↑

10. Conclusions