AI-STUDY 4기 · #2 MANIPULATION

ACT를 넘어 Diffusion Policy로

2026년 9월 22일 · 김재원

참고 논문 ALOHA(2023) · Diffusion Policy(2023, 확장판 2024)

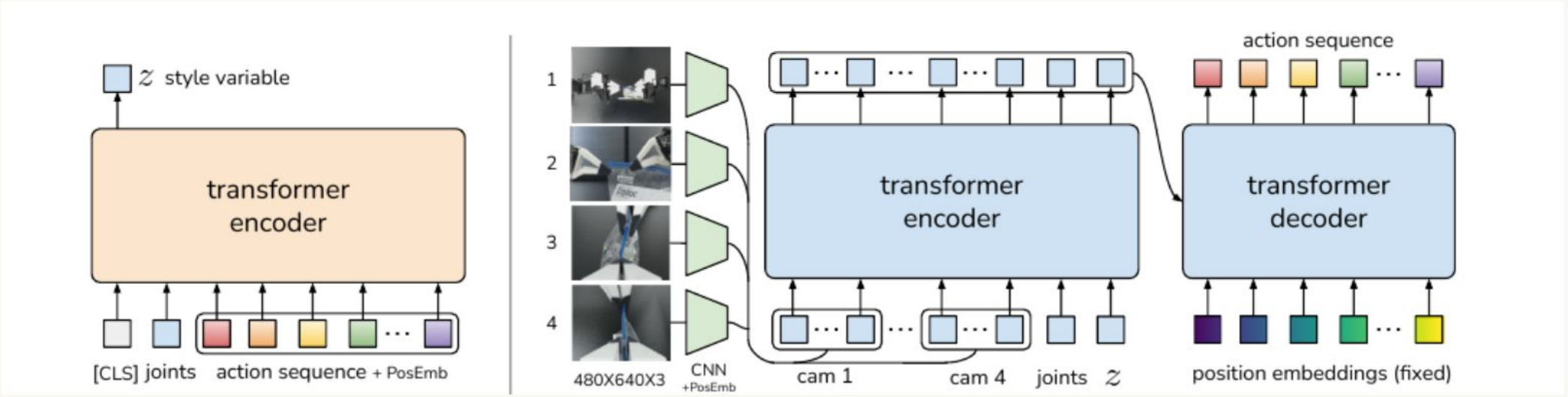
두 논문은 같은 두 문제를 푼다

두 논문 모두 behavior cloning을 바탕으로 한다. 사람이 보여 준 demonstration을 정답지 삼아 「이 observation이면 이 action」을 배우는 가장 단순한 방법이다.

단계	무엇을 하나
① demonstration 수집	사람이 로봇을 조종해 과제를 해 보이고 observation과 action을 기록한다
② training	그 짝을 지도학습으로 배운다
③ inference	새 observation을 받아 action을 낸다

하지만 이 방법을 로봇에 적용하면 영상 분류에는 없던 두 문제가 불거진다. **작은 오차가 눈덩이처럼 쌓이는 문제**와, **하나의 상황에 정답이 여러 갈래로 나뉘는 문제**다.

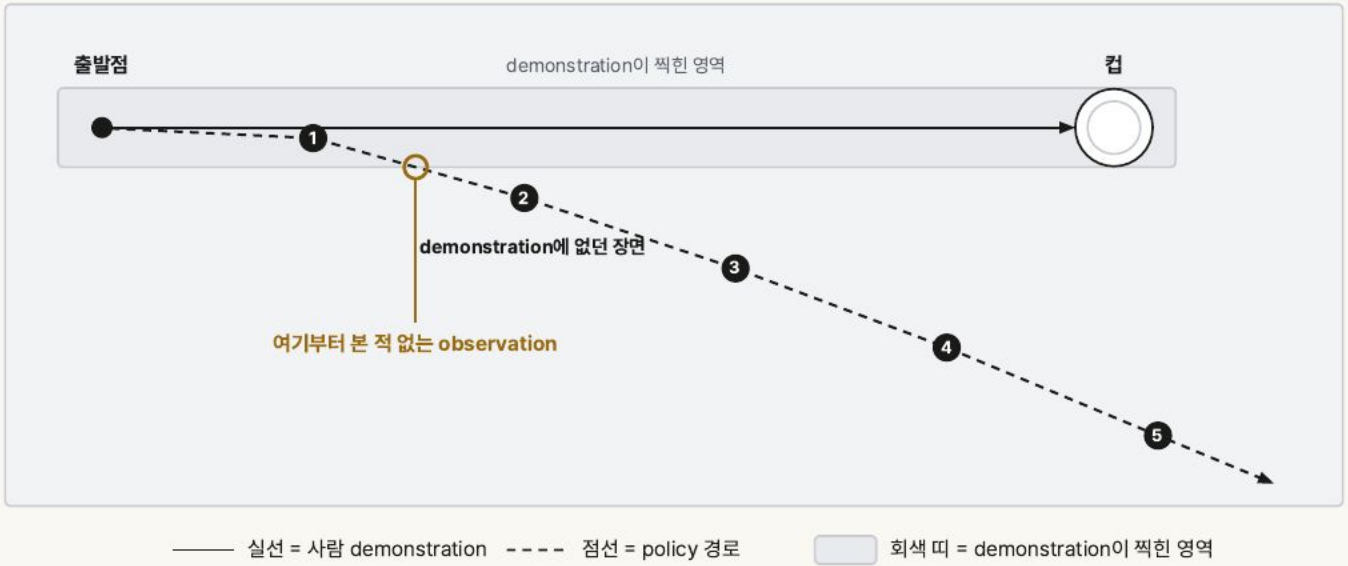
ACT는 inference할 때 z를 0으로 고정한다



왼쪽 encoder가 demonstration의 다양한 스타일을 z에 담아 두지만, inference할 때는 이 encoder를 버리고 z를 0으로 고정한다. 결국 같은 observation을 받아도 늘 똑같은 action 하나만 내놓는 한계에 갇힌다. 그런데 이 encoder 자체를 빼면 성공률이 **35.3%에서 2%**로 떨어진다.

Zhao et al., ALOHA, 2023 · Fig. 4 (원문 그림 인용) · §IV-B · §VI-B

내가 낸 action이 다음 observation을 바꾼다

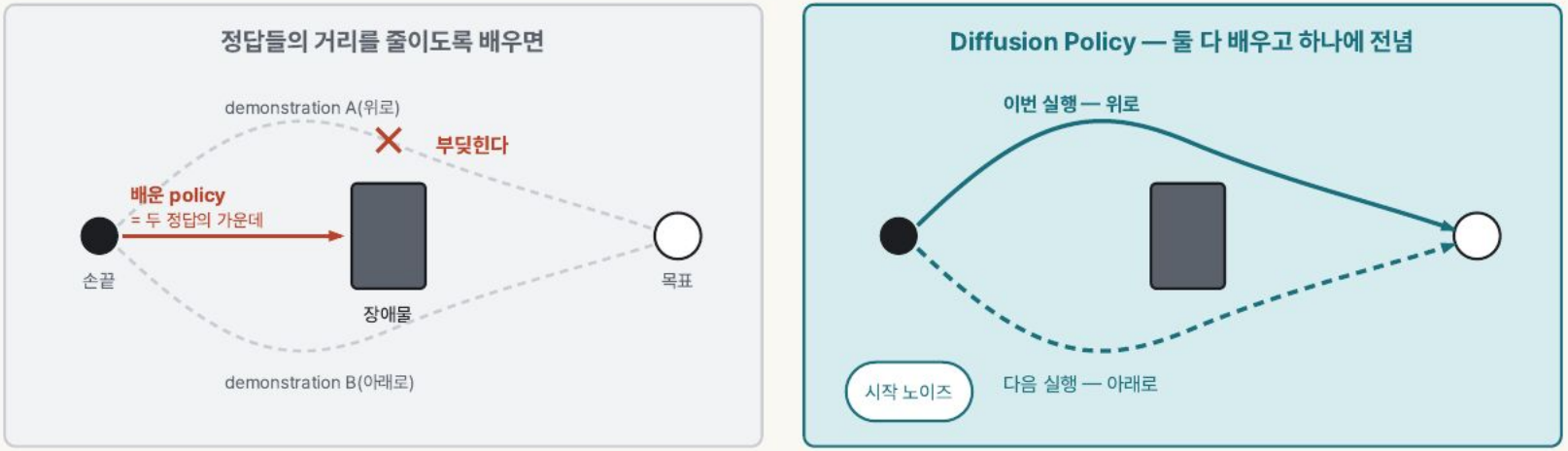


한 번 궤적이 어긋나면 demonstration에 없던 낯선 상태로 밀려나고, 오차는 더 크게 벌어진다. 영상 분류는 한 장을 틀려도 다음 사진이 제자리에 들어오지만, 로봇은 내가 방금 낸 동작이 다음 입력을 통째로 바꿔 버린다.

한 step에 1% 틀리는 policy가 100 step	틀리는 횟수
쌓이지 않는다면	**1번**
상한 $T^2\varepsilon$	**100번**

Ross et al., DAgger, 2011 · ALOHA §II

같은 observation에 정답 action이 여럿이다



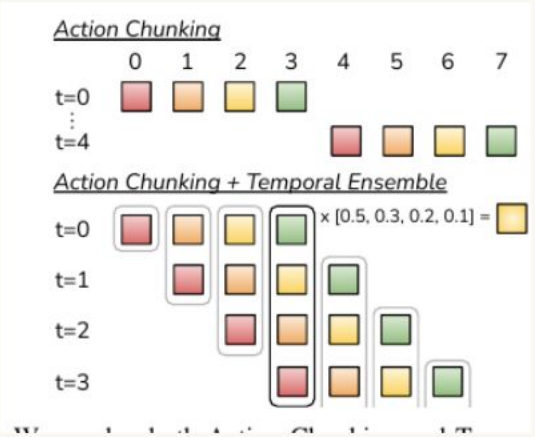
여러 갈래가 나오는 이유 둘 — ① 출발 노이즈가 매번 다르다 ② 반복마다 노이즈를 더해 한 봉우리에 빨려 들지 않는다 (§4.1)

왼쪽으로 돌아도 오른쪽으로 돌아도 정답이다. 하지만 두 경로를 단순 평균하면 곧장 장애물을 들이받는 가운데로 향한다.

ALOHA 논문은 demonstration을 **한 사람이 혼자 다 해 보였는데도** 테이프를 넘겨주는 위치가 매번 달랐다고 적었다. 로봇 제어에서 multimodality가 실재한다는 대표적인 예가 RT-1에 있다 — 256칸으로 나눠 고르던 출력을 가우스 분포 하나로 바꾸자 성공률이 **97에서 68**로 떨어졌다.

같은 두 문제를 서로 다른 도구로 풀었다

ACT는 action chunking과 CVAE로 풀었고, Diffusion Policy는 action chunking에 다시 계획하기를 더해 diffusion으로 답했다. 두 접근법의 공통점도 뚜렷하다. action을 묶어 예측하며, 묶음이 지나치게 길면 성능이 꺾이고, 상대 위치보다 절대 위치 명령이 유리하다.

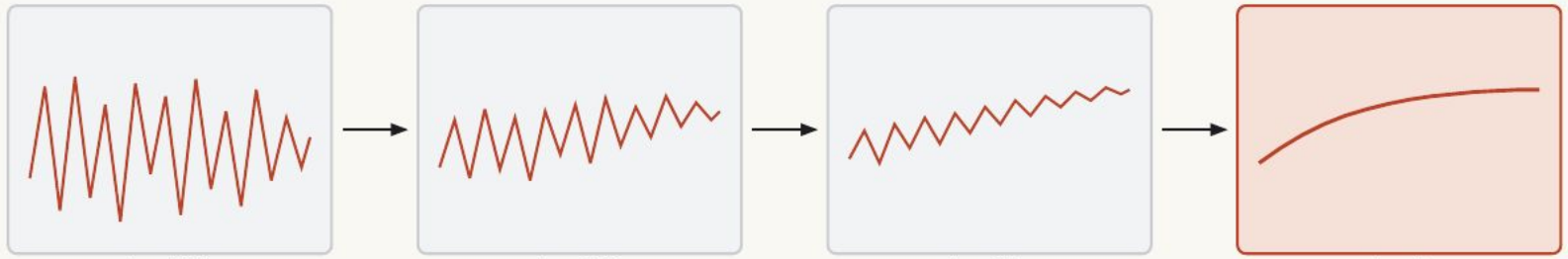


Zhao et al., ALOHA, 2023 · Fig. 5 (원문 그림 인용)

PART 1

diffusion은 어떻게 동작하나

노이즈에서 action이 나온다

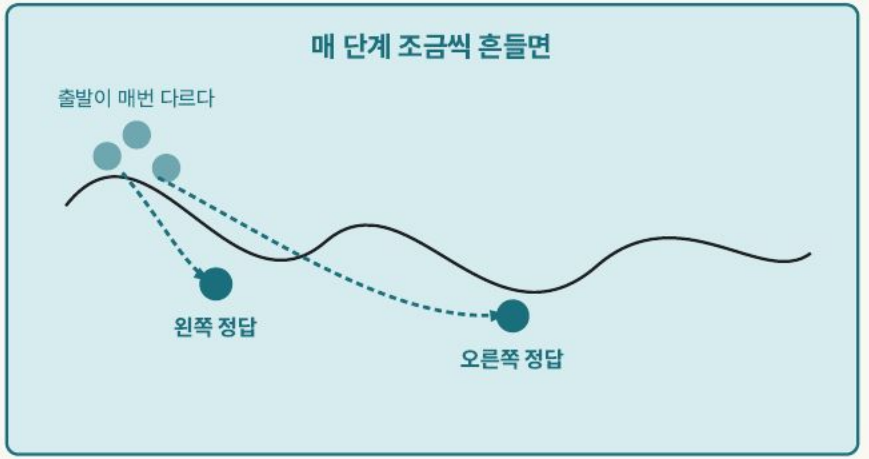


매 step: 섞인 노이즈를 신경망이 맞히고 그만큼 뺀 뒤 조금 다시 더한다

정상 데이터에 노이즈를 조금씩 섞어 망가뜨린 뒤, training 때 이를 되돌리는 역과정을 배운다. inference할 때는 무작위 숫자에서 출발해 노이즈를 열 번 걷어 낸다. 완성된 action 궤적을 한 번에 찍어내기는 어렵지만, 살짝 흐려진 것을 조금씩 또렷하게 다듬기는 쉽다.

denoising 하면서 노이즈를 다시 더한다





흔들림을 섞은 경사 하강법과 같은 모양이다. 적당히 흔들어 주어야만 한 골짜기에 갇히지 않는다. 이것이 05장에서 살펴본 「정답이 여럿」일 때 여러 갈래 경로를 온전히 살려내는 장치다.

training은 노이즈 맞히기 한 줄이다

데이터에 섞은 노이즈를 신경망이 예측하게 두고, 그 오차만 줄여 나간다.

$$\mathcal{L} = \mathrm{MSE}\left(\varepsilon^{k},\ \ \varepsilon_\theta(\mathbf{O}_t,\ \mathbf{A}_t^{0} + \varepsilon^{k},\ k)\right)$$

더한 노이즈 ↔ 신경망이 맞힌 노이즈

1. 깨끗한 action 궤적을 하나 꺼낸다
2. 걷어 낼 횟수 k를 무작위로 고른다
3. 그 k에 맞는 세기의 노이즈를 더한다
4. 신경망에 넣어 노이즈를 예측하게 한다
5. 예측과 실제 노이즈의 차이(MSE)를 줄인다

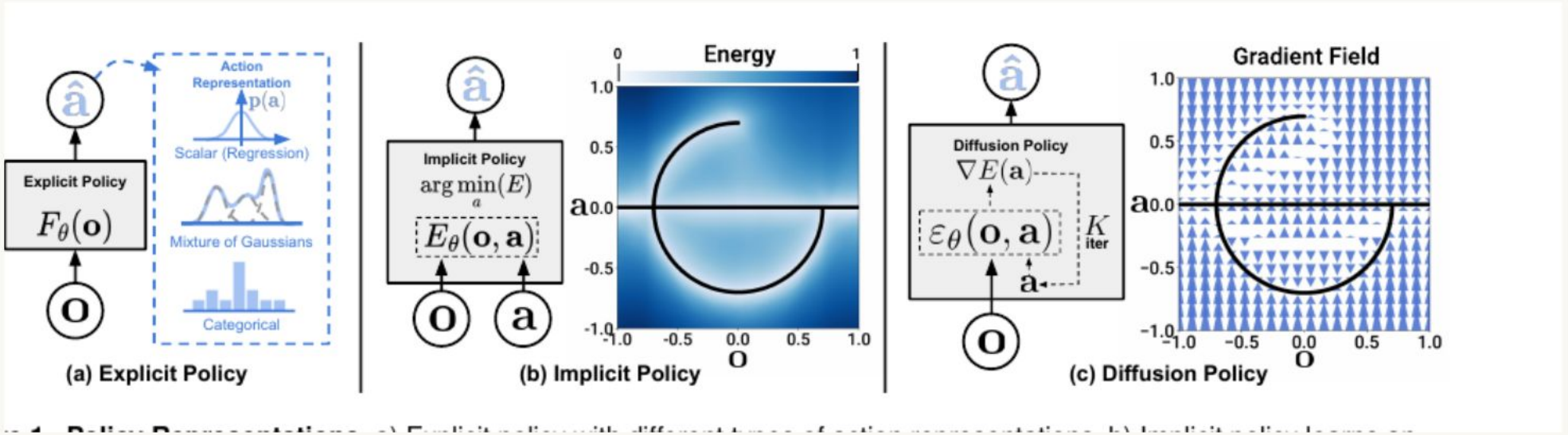
보통 diffusion 모델을 유도하려면 복잡한 ELBO를 거쳐야 하지만, 가우스 diffusion에서는 수식이 단순화되어 이 한 줄로 깔끔하게 떨어진다. 논문도 증명을 다시 적지 않고 기존 유도를 인용하기만 한다.

Chi et al., Diffusion Policy, 2024 · §2.2

PART 2

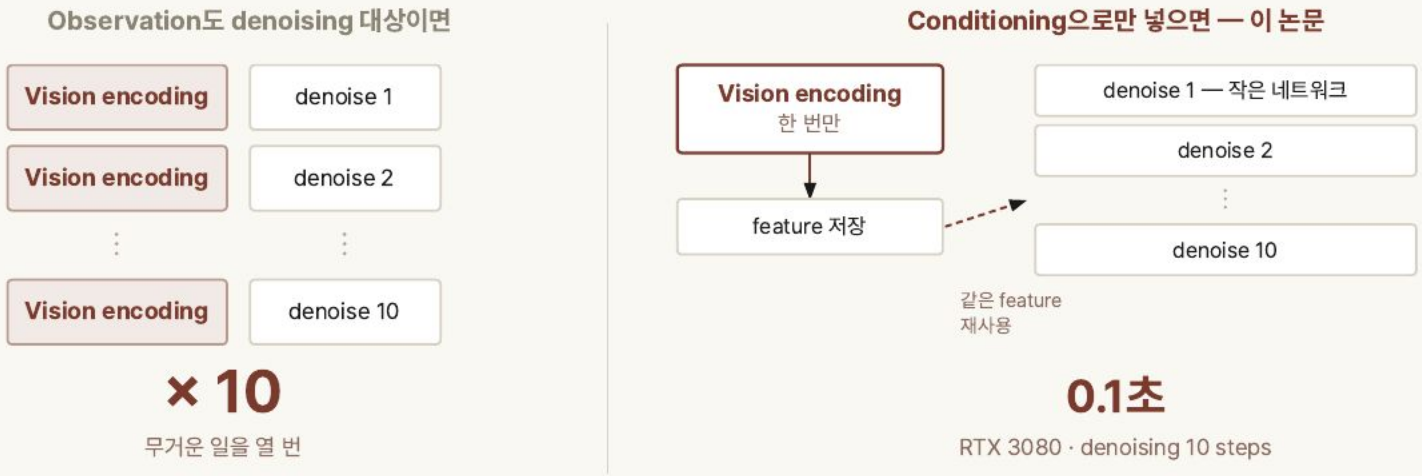
Diffusion Policy의 핵심 설계

IBC도 multimodality를 표현했다



explicit policy는 action을 단번에 출력한다. 반면 implicit policy(IBC)는 「이 action이 얼마나 안 어울리나」를 energy로 매긴 뒤 가장 낮은 값을 찾는다. 문제는 표현력이 아니라 불안정한 training 과정이었다. Diffusion Policy는 energy 함수 자체가 아니라 **기우는 방향**만 학습해 그 불안정을 비껴간다.

10번 반복해도 compute cost는 10배가 되지 않는다



매 denoising step마다 영상을 다시 encoding하지 않는다. 반복 루프 밖에서 observation을 한 번만 encoding해 두고, 그 feature를 열 번의 반복 내내 재사용한다.

16개 만들어 8개만 쓴다



길면 반응이 느려지고(열린 고리에 가까워짐), 짧으면 action이 들쭉날쭉해지고 계산이 는다 — 그 사이가 8

길게 실행하면 반응이 느려지고, 너무 짧게 끊으면 동작이 들쭉날쭉해진다. 예측한 16개 중 8개만 실행하고 다시 계획하는 방식이 대부분 과제에서 가장 안정적이었다.

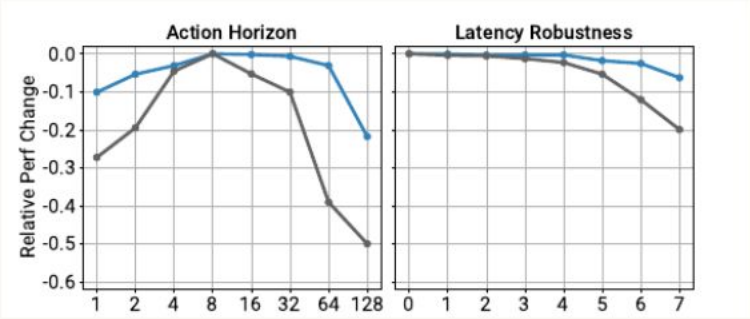
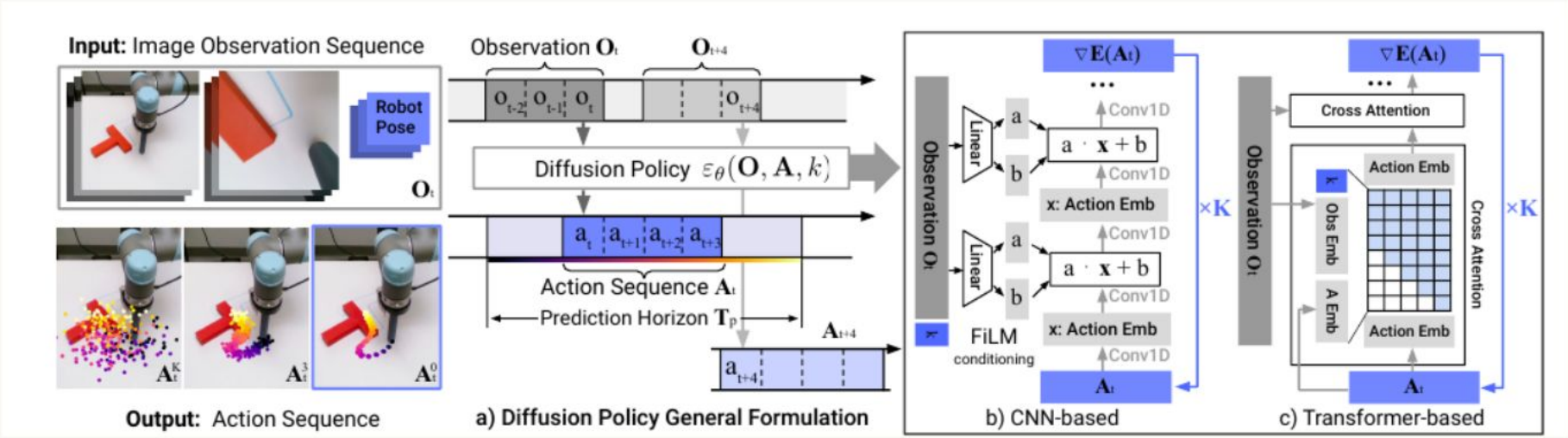


Figure 5 — action horizon을 1부터 128까지 바꿔 본 것. 8을 지나면 떨어진다.

새 과제는 CNN판부터 쓴다



구조는 둘이다. b)가 CNN판, c)가 Transformer판이다. 논문은 새 과제라면 CNN판부터 써 보라고 권한다. 실제 Push-T에서 CNN판이 **95%**, Transformer판이 65%였다. CNN판은 조정할 하이퍼파라미터가 적고, Transformer판은 사람이 정해 주는 값에 민감하다.

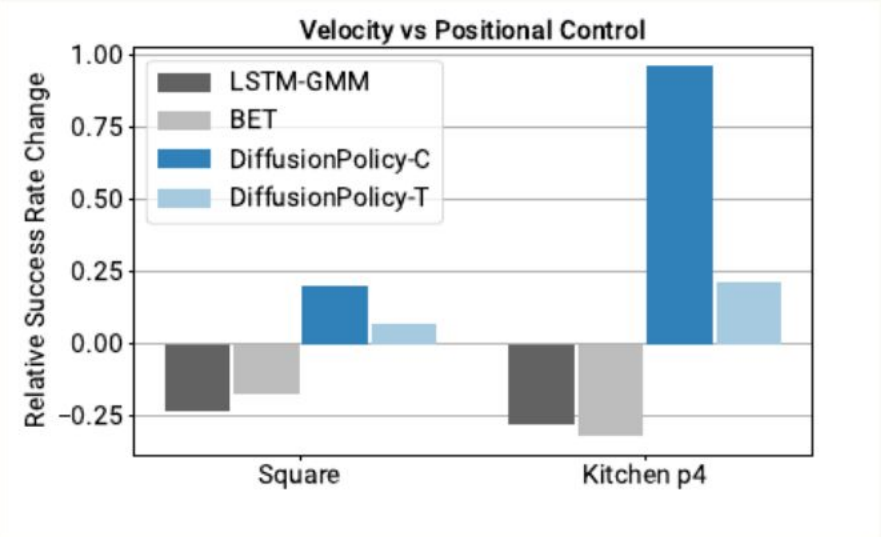
고정해 쓰는 것이 가장 나빴다

policy의 「눈」 역할을 할 encoder를 마련하는 방법은 셋이다. 밑바닥부터 같이 배우거나, 미리 학습한 가중치를 고정해 쓰거나, 작은 learning rate로 다듬는 선택지다.

encoder	밑바닥부터	고정	fine-tune
ResNet-18	**0.94**	0.58	0.92
ResNet-34	0.92	0.40	**0.94**
CLIP ViT-B/16	0.22	0.70	**0.98**

논문은 Diffusion Policy가 흔히 쓰는 시각 사전학습 표현과 사뭇 다른 특징을 선호한다고 해석했다. 0.22에 머물던 CLIP이 0.98로 뛴 까닭도 데이터가 적은 탓으로 보았다. 여기 쓰인 demonstration은 200개뿐이다.

위치 제어 명령과 더 잘 맞는다



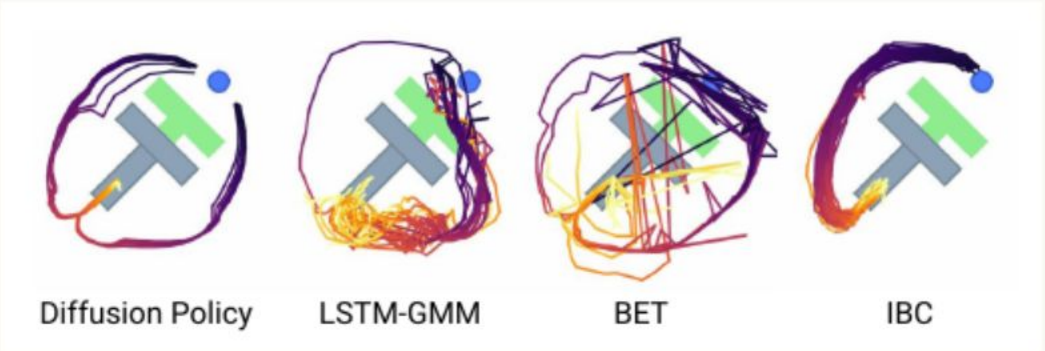
속도 제어에서 위치 제어로 바꿨을 때 나타나는 변화다. LSTM-GMM과 BET는 성공률이 오히려 떨어지는데 Diffusion Policy만 올라간다. 위치 명령은 도달할 절대 좌표를 지정하므로 앞 step이 조금 어긋나도 다음 명령이 스스로 오차를 보정한다.

Chi et al., Diffusion Policy, 2024 · Figure 4 (원문 그림 인용) · §4.2 · §5.3

PART 3

실제 성능은 어떻게 달라졌나

두 갈래를 모두 배우고 한쪽에 전념한다



40 step을 돌린 실제 궤적이다. LSTM-GMM과 IBC는 한쪽으로 치우치고, BET는 두 갈래 사이에서 갈팡질팡 오간다. Diffusion Policy만 두 갈래를 모두 배운 뒤, 실제 실행할 때는 한쪽 경로를 끝까지 따른다.

Chi et al., Diffusion Policy, 2024 · Figure 3 (원문 그림 인용) · §4.1

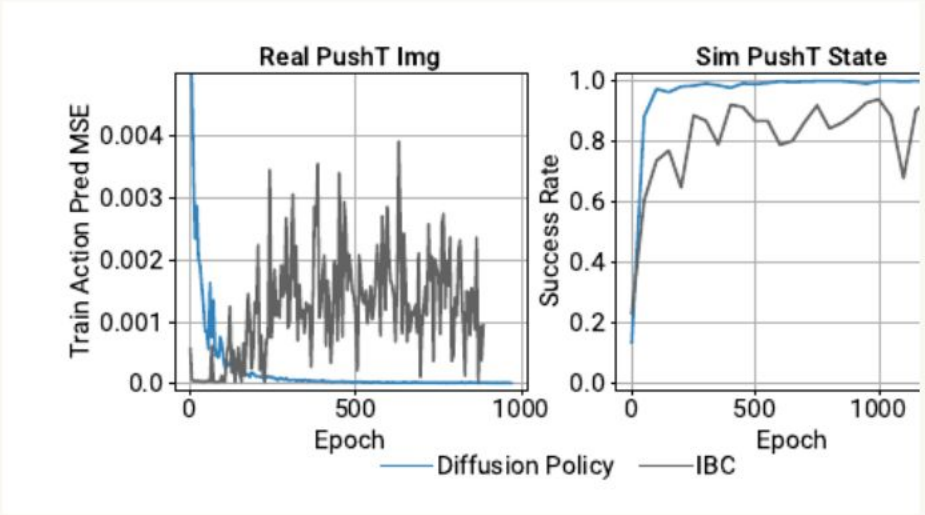
사람이 멈춘 구간에 강하다

사람이 로봇을 조종하다 보면 잠깐 멈추는 구간이 생긴다. 액체를 따를 때처럼 의도적으로 멈추기도 한다.

그때 생기는 것	무슨 일이 일어나나
idle action	「제자리 정지」 상태가 여러 step 이어져 demonstration에 남는다
한 step씩 내는 policy	그 구간에 과적합되어 **실제로 돌리면 로봇이 멈춰 선다**
action을 묶어 내면	미래의 여러 step을 한 번에 내다보므로 일시적인 멈춤에 갇히지 않는다

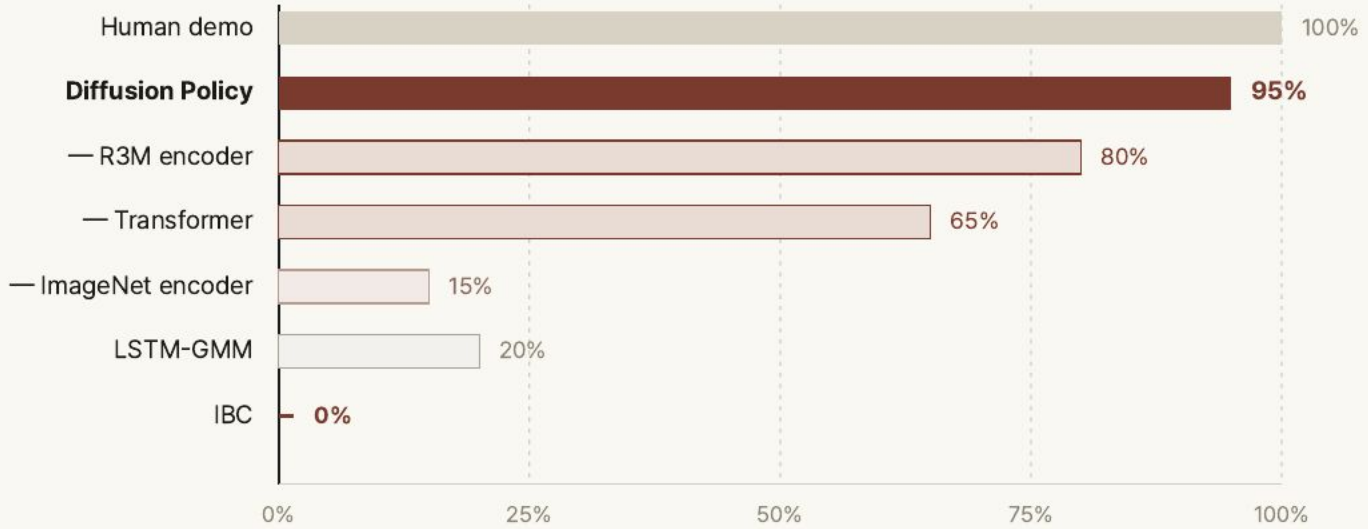
논문은 정지 구간을 데이터에서 지우지 않으면 **BC-RNN과 IBC가 실제 실험에서 자주 멈춰 섰다**고 적었다. demonstration을 어떻게 모았는지가 policy 구조 선택에 큰 영향을 준다는 좋은 예다.

training이 흔들리지 않는다



왼쪽은 training 오차, 오른쪽은 성공률이다. IBC(회색)는 오차가 줄어드는 흐름 속에서도 중간중간 크게 튀고, 성공률이 계속 출렁인다. 어느 checkpoint를 고르느냐에 따라 평가 결과가 요동친다. 실제 Push-T에서 IBC는 위치 제어와 속도 제어 모두 **0%** 였다.

성공률 95%와 80%는 성공 횟수 세 번 차이다



각 policy를 20회씩 평가하므로 1회 성공이 곧 5%p 차이다. 눈여겨볼 것은 맨 아래 0%다 — IBC는 같은 demonstration을 쓰고도 단 한 번도 성공하지 못했다.

속도를 내주고 표현력을 얻었다

논문 저자들이 스스로 밝힌 한계는 둘이다.

한계	논문이 적은 것
behavior cloning의 구조적 한계를 물려받는다	demonstration이 모자라면 성능이 낮다. 저자들은 강화학습 등으로 영역을 넓힐 수 있다고 적었다
compute cost와 지연 시간	단순한 방법보다 연산 부담이 크다. action chunking이 짐을 덜어 주지만 **아주 빠른 제어가 필요한 과제에는 모자랄 수 있다**

ACT가 한 번 답하는 데 0.01초, Diffusion Policy가 0.1초다. ⚠️ 다만 **그래픽카드가 달라서**(RTX 2080 Ti ↔ 3080) 그대로 견주면 안 된다. 방향만 맞다.