Diffusion Policy: Visuomotor Policy Learning via Action Diffusion

By: Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, Shuran Song

Motivation

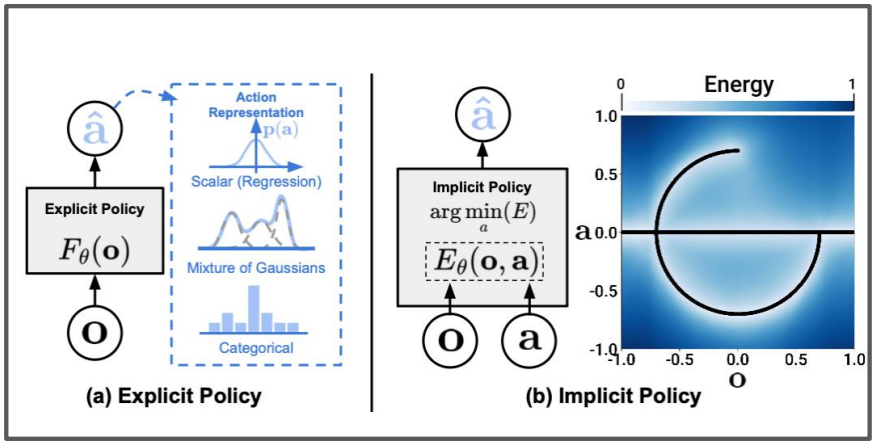
- **Goal:** Learn robot behaviors from demonstrations by mapping observations to actions.
- **Challenge:** Unlike ordinary prediction tasks, robot actions can have multiple valid solutions, must be temporally consistent, and often require high precision.

Multimodality

Temporal consistency

Precision

Prior work

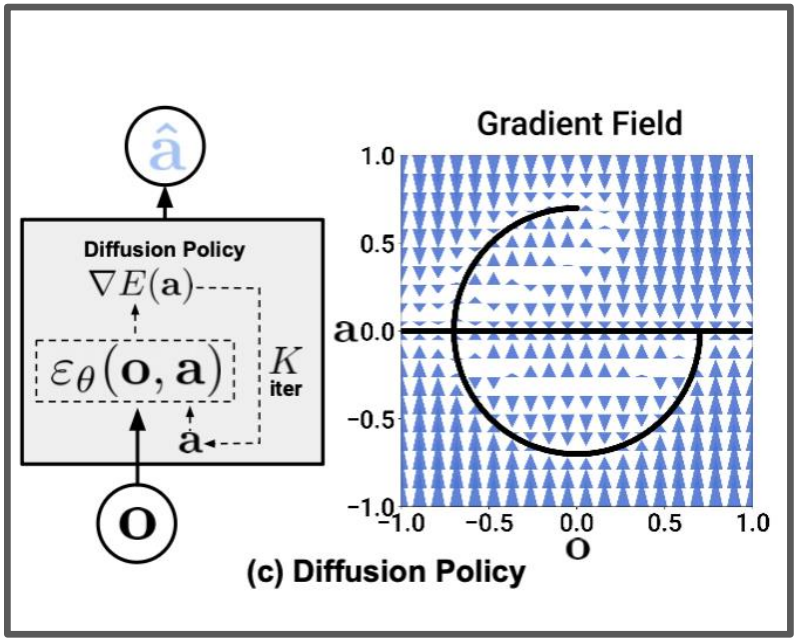


Challenge: Modeling multimodal action distributions while maintaining temporal consistency and precision

- **(a) Explicit policies:** Use different action representations (regression, Gaussian mixtures, categorical distributions) to predict actions. Can struggle with complex, high-dimensional action distributions.
- **(b) Implicit policies:** Learn an energy function to represent possible actions. Can capture multimodality but suffer from training instability due to negative sampling.

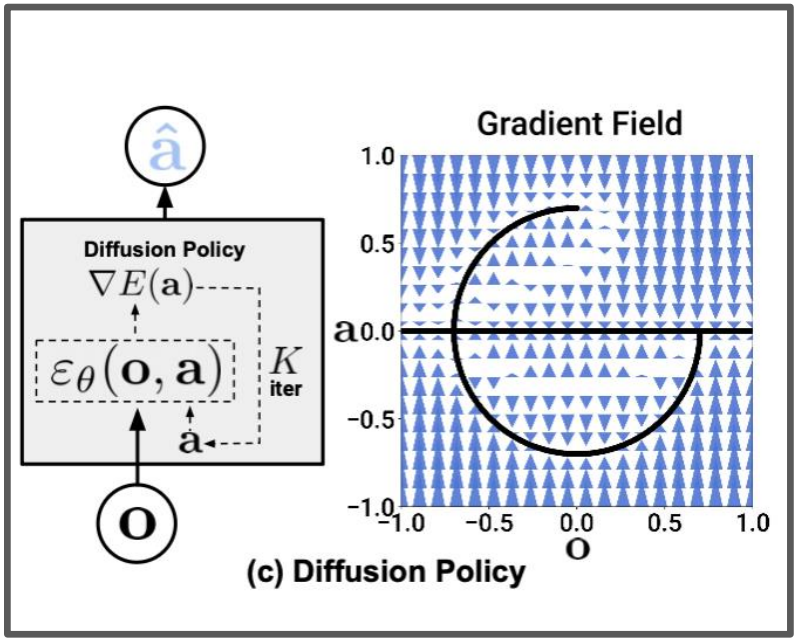
Research gap: How can we model complex, multimodal action sequences while maintaining stable training?

Diffusion policy



- **Represents a visuomotor policy as a conditional diffusion model over robot action sequences.**
  - Learns a noise-prediction network related to the score of the action distribution, then uses iterative denoising to generate actions conditioned on visual observations.
- **Advantages:** Models multimodal action distributions, supports high-dimensional action spaces, and offers more stable training than existing energy-based approaches.

Diffusion policy: contributions



- **Closed-loop action sequences:** Combines action-sequence prediction with receding-horizon control, allowing the robot to replan while maintaining temporal consistency.
- **Visual conditioning:** Conditions action generation on visual observations, reusing encoded features to reduce computation.
- **Time-series diffusion Transformer:** Reduces the over-smoothing seen in CNN-based models, improving predictions for rapidly changing actions.

DDPM: changes in process

$$x^{k-1} = \alpha(x^k - \gamma\,\varepsilon_\theta(x^k, k) + \mathcal{N}(0, \sigma^2 I))$$

DDPM denoising update

$$A_t^{k-1} = \alpha(A_t^k - \gamma\,\varepsilon_\theta(O_t, A_t^k, k) + \mathcal{N}(0, \sigma^2 I))$$

Conditional denoising

$$\mathcal{L} = \mathrm{MSE}(\varepsilon^k, \varepsilon_\theta(x^0 + \varepsilon^k, k))$$

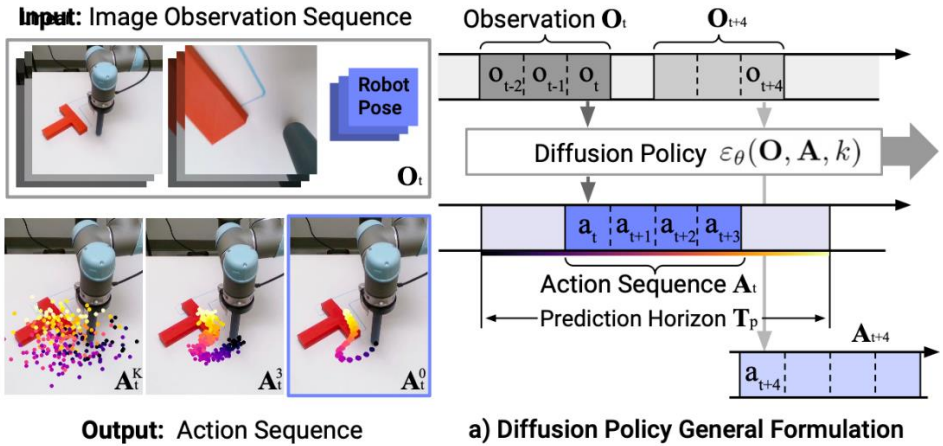
DDPM training loss

$$\mathcal{L} = \mathrm{MSE}(\varepsilon^k, \varepsilon_\theta(O_t, A_t^0 + \varepsilon^k, k))$$

Conditional training loss

- **DDPM:** Iteratively denoises Gaussian noise over K steps (Eq. 1).
- **Training:** Predicts added noise using an MSE loss (Eq. 3).
- **Robotics adaptation:** Generates action sequences conditioned on observations, modeling $p(A_t \mid O_t)$ (Eqs. 4–5).
    - Reuses visual features across denoising steps to reduce computation.

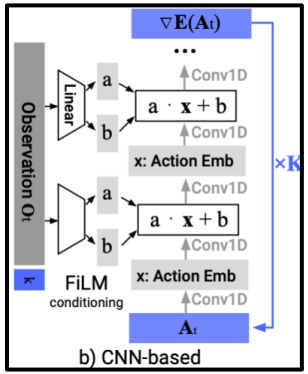
Closed loop action-sequence prediction



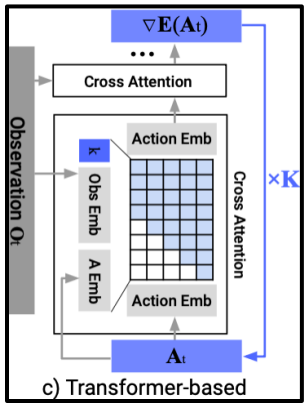
- **Goal:** Maintain temporal consistency while allowing the robot to respond to new observations.
- **Prediction:** Use the latest T_o observations to generate T_p future actions.
- **Execution:** Execute only T_a actions before observing and replanning (receding-horizon control).

Design: neural network architecture for εθ

	CNN-Based policy	**Time-series diffusion transformer**
Architecture	1D temporal CNN with FiLM conditioning	Causal self-attention and cross-attention
Strengths	Works well on most tasks with minimal tuning	Better for complex tasks and rapidly changing actions
Limitations	Can over-smooth rapid action changes	More sensitive to hyperparameter tuning



b) CNN-based



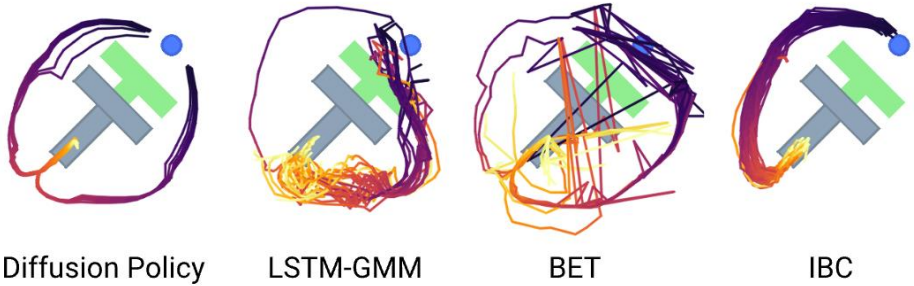
c) Transformer-based

Recommendation: Start with the CNN-based policy; switch to the Transformer for more complex tasks or rapidly changing actions.

Design: implementation details

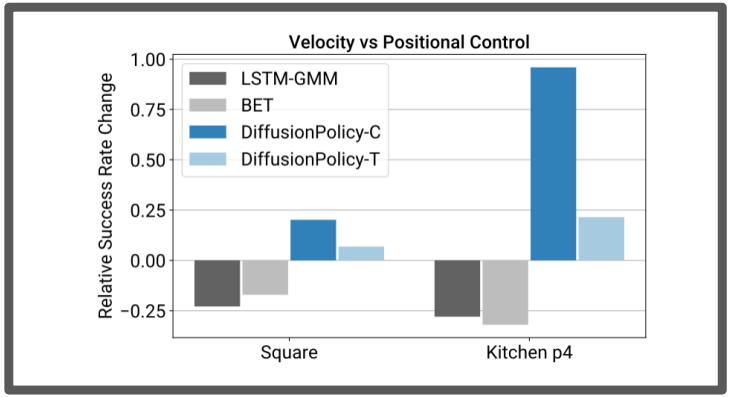
- **Visual Encoder:** Uses an 18-layer Residual Network (ResNet-18), trained end-to-end, with spatial softmax to preserve spatial information and Group Normalization for stable training.
- **Noise Schedule:** Uses a square cosine schedule to control how much noise is added or removed at each diffusion step.
- **Real-Time Inference:** Uses Denoising Diffusion Implicit Models (DDIM) to reduce inference from 100 to 10 denoising steps, achieving approximately 0.1-second latency on an NVIDIA 3080 graphics card.

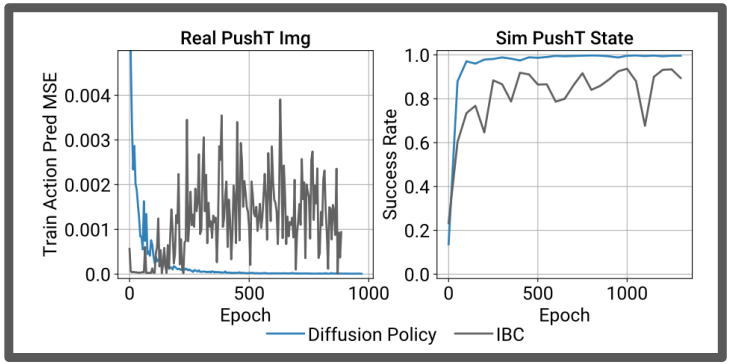
Multimodal action distributions and temporal consistency



- **Multimodal action distributions:** Diffusion Policy uses stochastic initialization and Langevin dynamics to generate different valid action trajectories.
- **Temporal action consistency:** By predicting entire action sequences, the policy can commit to one mode rather than alternate between different trajectories.
- **Robustness to idle actions:** Action-sequence prediction makes the policy less likely to get stuck when demonstrations contain pauses or repeated actions.

Position control and training stability



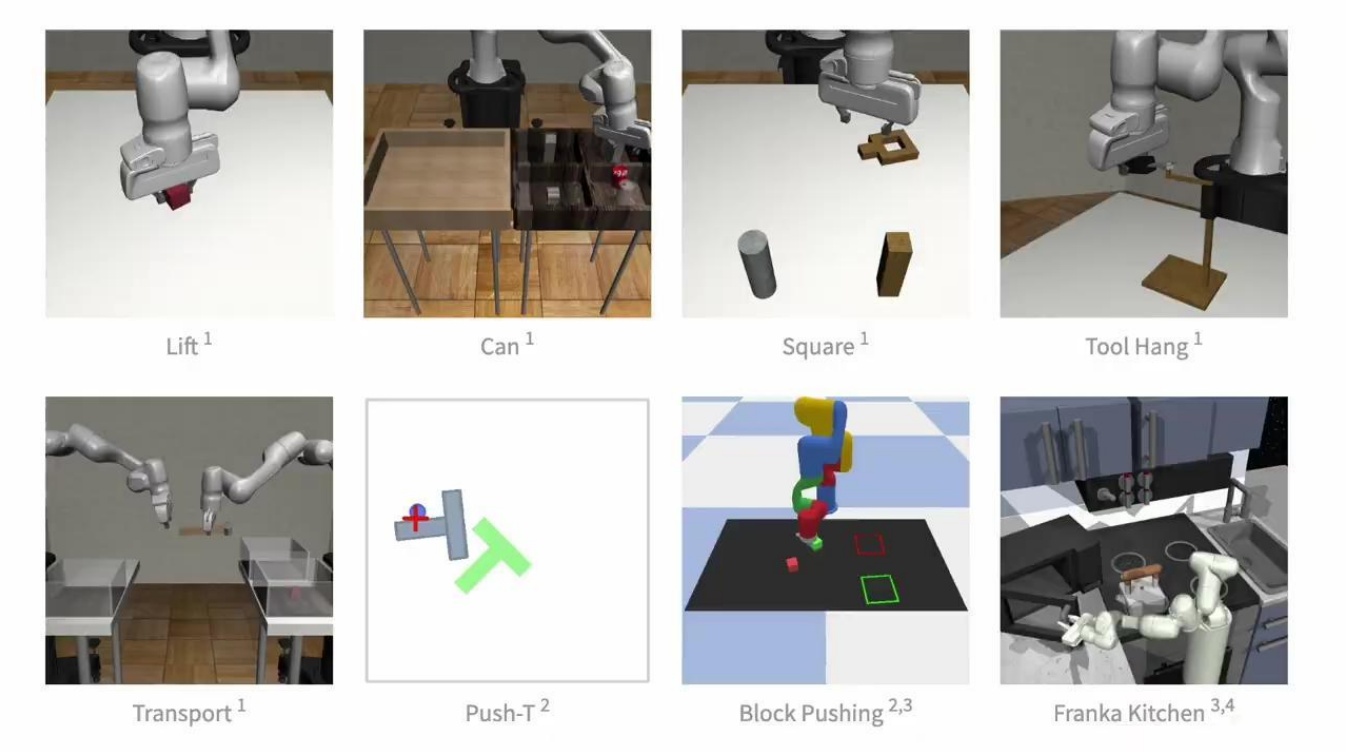


- **Position Control (top):** Diffusion Policy performs better with position commands than velocity commands, likely due to its ability to model multimodal action distributions and reduce compounding errors.
- **Training Stability (bottom):** Unlike energy-based methods such as IBC, Diffusion Policy avoids negative sampling by learning the action-distribution score through noise prediction, resulting in more stable training.

Experimental setup

- **Evaluation:** 15 tasks across 4 benchmarks, including simulated and real-world robot manipulation.
- **Tasks:** Object pushing, grasping, assembly, pouring, and bimanual manipulation.
- **Baselines:** LSTM-GMM (BC-RNN), Implicit Behavior Cloning (IBC), and Behavior Transformer (BeT).
- **Metrics:** Task success rate and target coverage (Push-T).

Results: Simulation environments



Results: Real world evaluation



- **Evaluation:** Tested Diffusion Policy on 4 real-world manipulation tasks across 2 robot setups: Push-T, mug flipping, sauce pouring, and sauce spreading.
- **Push-T:** Achieved **95% success**, compared to 20% for LSTM-GMM and 0% for Implicit Behavioral Cloning.
- **Other tasks:** Achieved 90% success on mug flipping, 79% on sauce pouring, and 100% on sauce spreading.

Results: Real world bimanual tasks

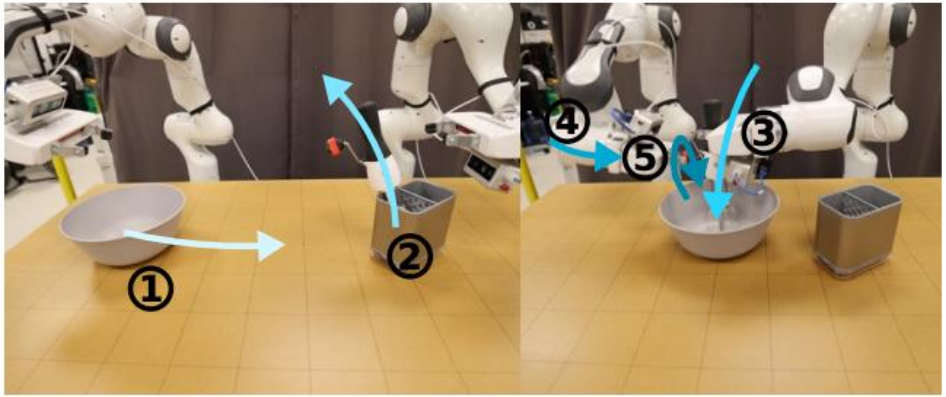


Figure 11. Bimanual Egg Beater Manipulation. The robot needs to ① push the bowl into position (only if too close to the left arm), ② approach and pick up the egg beater with the right arm, ③ place the egg beater in the bowl, ④ approach and grasp the egg beater crank handle, and ⑤ turn the crank handle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

- **Evaluation:** Three real-world bimanual tasks: egg beating, mat unrolling, and shirt folding.
- **Results:** 55% success on egg beating and 75% on both mat unrolling and shirt folding.

With deterministic DDIM sampling, does multimodality come only from the initial noise?

Yes, multimodality comes from the initial Gaussian noise because subsequent denoising steps are deterministic. However, the paper also uses stochastic DDPM sampling, where additional Gaussian noise is introduced during denoising.

Why does the CNN backbone fail on Block Push?

Temporal CNNs tend to over-smooth rapidly changing actions, which may make the transitions between subtasks difficult. I'm not sure though.

Questions: are the baselines too weak?

- The baselines were established methods at the time, so I wouldn't necessarily call them weak.
- Other competitive approaches, such as **Action Chunking with Transformers (ACT)**, emerged around the same time. ACT also predicts action sequences, but uses a Transformer rather than iterative diffusion.
    a. **ACT (Action Chunking with Transformers)** is faster, requires less training data, and performs better on tasks needing quick reactions. It generates action sequences in a single forward pass.
    b. **Diffusion Policy** is generally stronger on high-precision tasks with multiple valid solutions, such as peg insertion and cloth folding, but requires more computation and training data