1.General Understanding

1.1 What is policy?

According to the policy in CS285笔记 , most policies there are defined by an explict function: policy π_θ is a function that maps a given state s to an action a .For example, BC (Behavioral Cloning) , in actor-critic, the actor is a explict policy. Instead of these, we also got implict policy, IBC (implict Behavioral Cloning) is a fantastic example. DQN (deep Q-Learning) also plays a role which implict policy is greedy policy: $a = \arg\max_a Q(s,a)$.
Apart from these policies, we now got Diffusion Policy, which we directly optimize the field of action space which will be proved in math principle session.

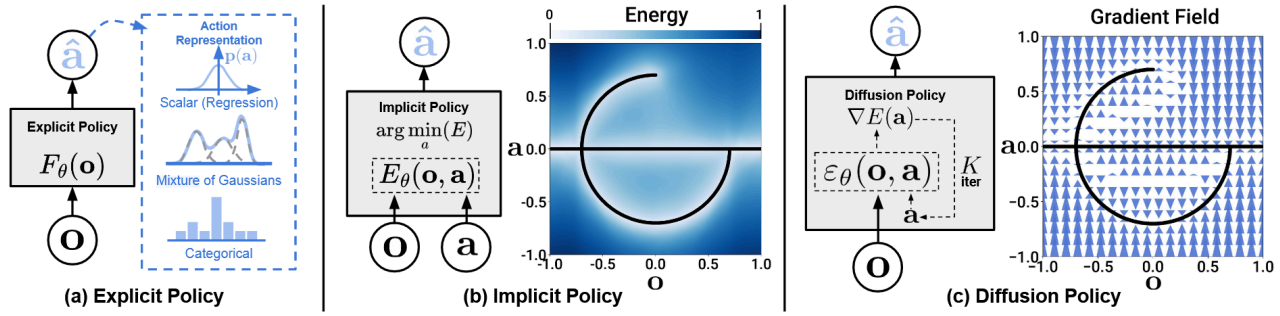


Figure 1. Policy Representations. a) Explicit policy with different types of action representations. b) Implicit policy learns an energy function conditioned on both action and observation and optimizes for actions that minimize the energy landscape c) Diffusion policy refines noise into actions via 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.

2.Math Principle

2.1 Diffusion equals to optimize gradient field

why this is improtant? solve the **Multimodality** problem.
different from implict policy which we learn **one** energy function to find the best action should be taken, learning a whole gradient feild helps that whatever randomly initializing noise in the global trajectory distribution and exploring through different denoising paths, the algorithm can effectively **cover various potential modes** within a multimodal policy distribution.
our goal is : The trained noise predictor $\epsilon_\theta(x_t, t)$ is proportional to the negative gradient of the log data distribution $\nabla x_t \log p(x_t)$
in formal,

$$\epsilon_\theta(x_t, t) \propto -\nabla_{x_t} \log p(x_t)$$

we want : build connection between our trained denoise ϵ_θ and $p(x_t)$
we have :

1.forward pass : $x_0 = \frac{x_t - \sqrt{1-\bar{\alpha}_t}\epsilon}{\sqrt{\bar{\alpha}_t}}$.
2.Denoise : $\epsilon_\theta \approx \epsilon$, ϵ is the true noise.
3.Bayes : $p(x_0|x_t) = \frac{q(x_t|x_0)p(x_0)}{p(x_t)}$
we want $\nabla_{x_t} \log p(x_t)$, so let's take the logarithm and then compute the gradient:

$$\nabla_{x_t} \log p(x_0|x_t) = \nabla_{x_t} \log q(x_t|x_0) + \nabla_{x_t} \log p(x_0) - \nabla_{x_t} \log p(x_t)$$

due to $\nabla_{x_t} \log p(x_0) = 0$
we finally got,

$$\nabla_{x_t} \log p(x_t) = \nabla_{x_t} \log q(x_t|x_0) - \nabla_{x_t} \log p(x_0|x_t)$$

When t is large:
As the time step t increases, we progressively add more and more noise to the image, until x_t becomes pure random noise.
At this point, x_t contains almost no information about the original image x_0. Therefore, the posterior distribution $p(x_0 \mid x_t)$ becomes nearly uniform (flat), meaning that all possible values of x_0 are approximately equally likely.
A flat distribution has a gradient close to zero; hence,

$$\nabla_{x_t} \log p(x_0 \mid x_t) \approx 0.$$

so we only need compute $\nabla_{x_t} \log q(x_t|x_0)$
x_t is a Gaussian distribution, it's Probability density function

$$p(\mathbf{x}) = \frac{1}{\sqrt{(2\pi)^d |\mathbf{\Sigma}|}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^\top \mathbf{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right)$$

so,

$$q(x_t \mid x_0) \propto \exp\left(-\frac{1}{2} \cdot \frac{\|x_t - \sqrt{\bar{\alpha}_t}x_0\|^2}{1-\bar{\alpha}_t}\right)$$

$$\nabla_{x_t} \log q(x_t|x_0) = \nabla_{x_t}\left(-\frac{||x_t - \sqrt{\bar{\alpha}_t}x_0||^2}{2(1-\bar{\alpha}_t)}\right)$$

$$= -\frac{1}{2(1-\bar{\alpha}_t)} \cdot \nabla_{x_t}||x_t - \sqrt{\bar{\alpha}_t}x_0||^2$$

$$= -\frac{1}{2(1-\bar{\alpha}_t)} \cdot 2(x_t - \sqrt{\bar{\alpha}_t}x_0)$$

so,

$$\nabla_{x_t} \log q(x_t|x_0) = -\frac{\sqrt{1-\bar{\alpha}_t}\,\epsilon}{1-\bar{\alpha}_t} = -\frac{\epsilon}{\sqrt{1-\bar{\alpha}_t}}$$

and now we use $\epsilon_\theta \approx \epsilon$, finally got what we want $\epsilon_\theta(x_t, t) \propto -\nabla_{x_t} \log p(x_t)$

2.2 Diffusion Policy's train & sample

2.2.1 Forward pass

in diffusion policy, our data aren't image x_0 but an action sequence $a_{0:H}$ which H is the length of sequence. we usually flatten $a_{0:H}$ as a high dim vector. Similar to DDPM, the forward pass in diffusion policy is add noise in the sequence:

$$q(a_t|a_{t-1}) = \mathcal{N}(a_t; \sqrt{1-\beta_t}a_{t-1}, \beta_t I)$$

Also, we have the connection between a_t with a_0 .

$$q(a_t|a_0) = \mathcal{N}(a_t; \sqrt{\bar{\alpha}_t}a_0, (1-\bar{\alpha}_t)I \qquad \bar{\alpha}_t = \prod_{i=1}^{t}(1-\beta_i)$$

the forward pass here is unconditional because add noise is independent with observations

2.2.2 Train & Sampling

In Training:
Different from DDPM, here we want to learn a conditional diffusion $p_\theta(a_{t-1}|a_t, o)$
similar to DDPM,we also use Variational Lower Bound here to train the model.
the core is :

$$\mathbb{E}_{q(a_t|a_0)}\left[\mathrm{KL}\left(q(a_{t-1} \mid a_t, a_0) \,\|\, p_\theta(a_{t-1} \mid a_t, o)\right)\right]$$

we can also prove that this KL Divergence is equal to MSE between mean

$$L \propto \left\|\mu_q(a_t, a_0) - \mu_\theta(a_t, o, t)\right\|^2$$

we have a fixed μ_q

$$\mu_q(a_t, a_0) = \frac{1}{\sqrt{\alpha_t}}\left(a_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon\right)$$

$$L = \mathbb{E}_{t, a_0, \epsilon, o}\left[\left\|\epsilon - \epsilon_\theta(a_t, o, t)\right\|^2\right]$$

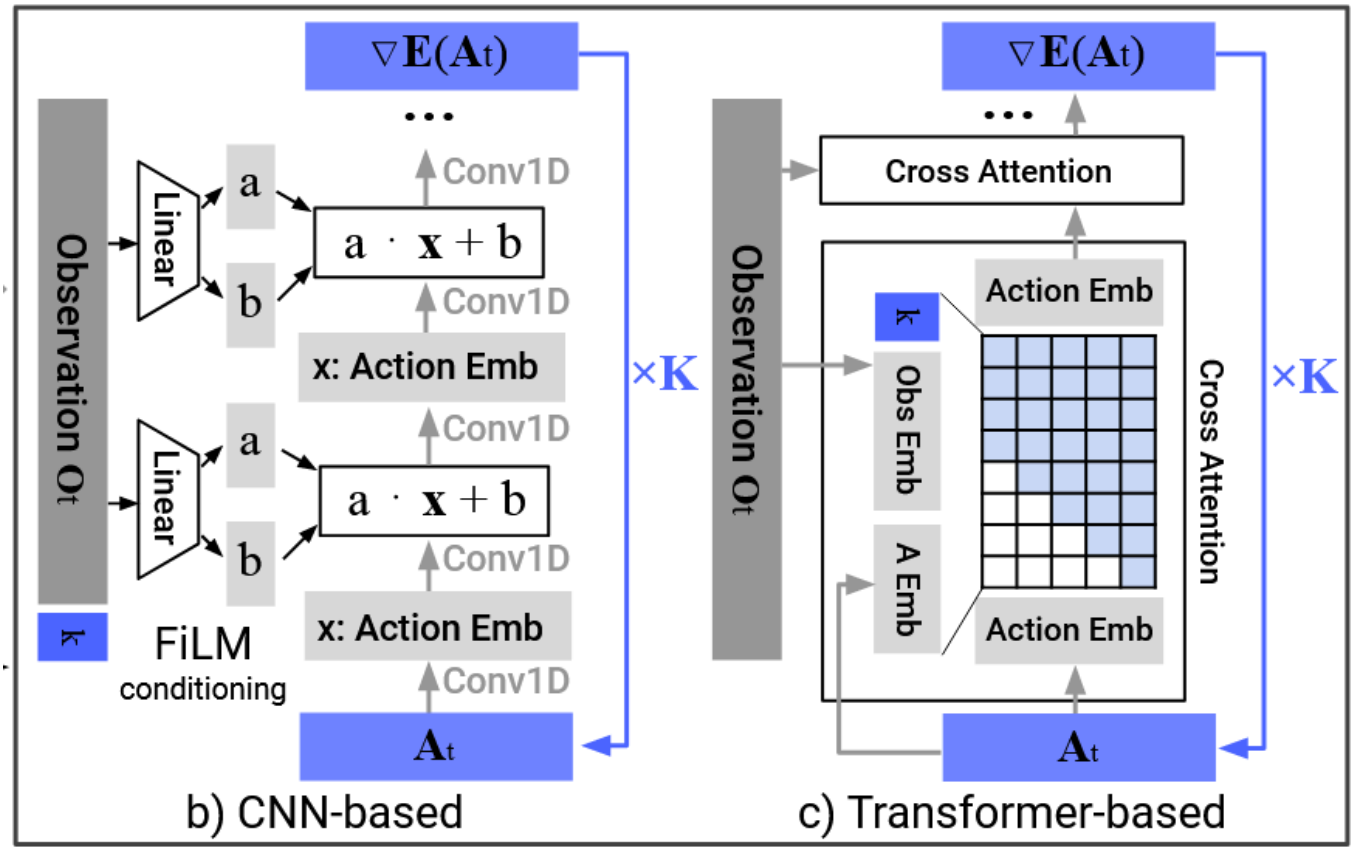
In Sampling:

$$a_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(a_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(a_t, o, t)\right) + \sigma_t z$$

3.Complete architecture

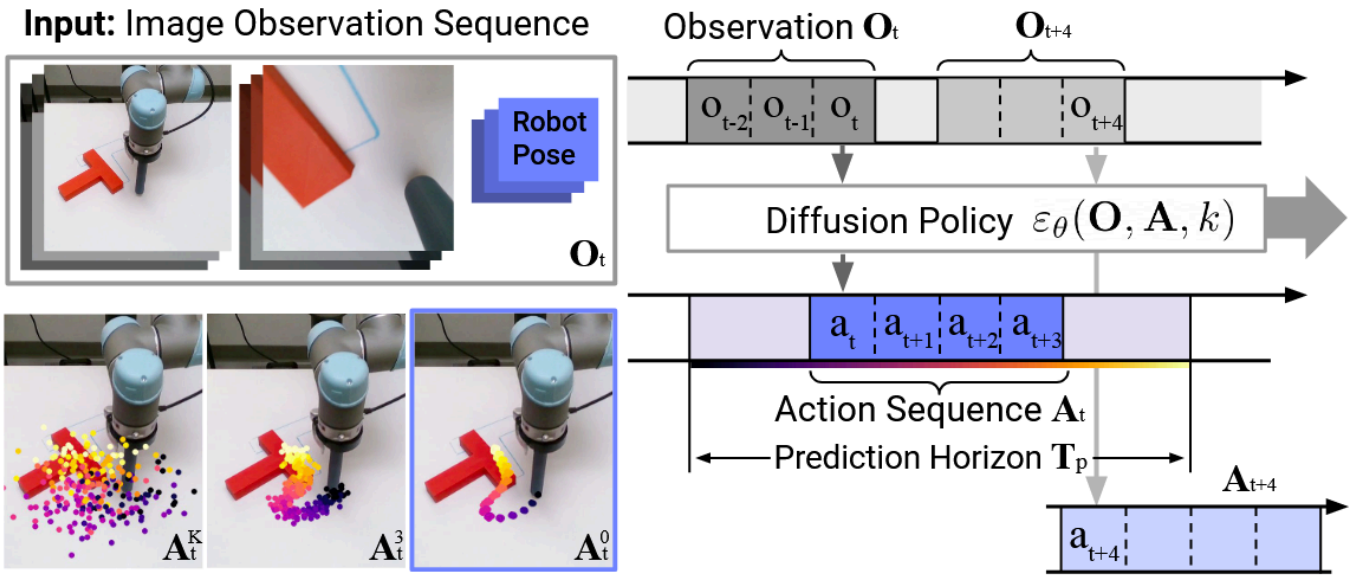
3.1 Denoise network

there are two denoise network can be chosen. It's important to mention that whatever in CNN-based or in Transformer-based , we both need a visual encoder to process the high dim observations, the encoder's output is the "Observation $\mathbf{O}_t$" here



3.2 General formulation

in Diffusion Policy, we have a sequence of Observation $\mathbf{O}_t$ and we generate a sequence of action $\mathbf{A}_t$.



Input: Observation Sequence

- **Observation** O_t: The model's input is the observation at time t. Crucially, this is not just a single static image, but an **observation sequence**. As shown in the figure, O_{t-2}, O_{t-1}, O_t represent several past frames of observations . A single frame lacks dynamic information such as object motion direction or velocity. By providing a temporal sequence, the model can infer environmental dynamics, enabling it to generate more accurate and temporally coherent actions.
  Output: Action Sequence
- **Action Sequence** A_t: The output is a **predicted sequence of future actions**. Unlike traditional Behavior Cloning (BC), which predicts only a single action, Diffusion Policy predicts a full sequence over a **prediction horizon** T_p . Many robotic tasks require a series of coordinated actions (e.g., reaching for and grasping a cup). Predicting an entire sequence enables the model to plan holistically, avoiding myopic decisions that focus solely on immediate rewards.

Diffusion Policy $\epsilon_\theta(O, A, k)$: The gray block in the center represents the core model — a **conditional diffusion model**.

- A: A noisy action sequence, which the model iteratively denoises.
- O: The observation sequence, serving as the **conditioning signal** that guides the denoising process toward contextually appropriate actions.
- k: The diffusion timestep, indicating the current stage of the denoising process (from high noise level K to clean action 0).

During training:

1. **Input**: Extract observation sequences O_t and corresponding clean expert action sequences A_t from demonstrations.
2. **Add noise**: Corrupt A_t with random Gaussian noise according to diffusion schedule k, resulting in a noisy version A_t^k.
3. **Learn noise prediction**: Train the network $\epsilon_\theta(O_t, A_t^k, k)$ to predict the injected noise, optimizing via mean squared error (MSE):

$$\mathcal{L} = \mathbb{E}_{O,A,k,\epsilon}\left[\|\epsilon - \epsilon_\theta(O, A_k, k)\|^2\right]$$

During inference:

1. **Input**: The robot receives the current observation sequence O_t.
2. **Initialization**: Start with a pure Gaussian noise action sequence A_t^K.
3. **Iterative denoising**: Feed A_t^K and O_t into ϵ_θ, progressively denoising over K steps to obtain a clean, feasible action sequence A_t^0.
4. **Execution**: Execute only the first action in A_t^0, then re-plan in the next step using updated observations — a strategy known as **Receding Horizon Control (RHC)**.

4.Reference

Diffusion Policy' paper: https://arxiv.org/abs/2303.04137v5
blogs: https://zhuanlan.zhihu.com/p/1948180866107941053

5.Links

works related to Diffusion Policy :

1. DiT , Diffusion with transformer
2. flow-match