扩散策略模块

最后一句“哲学设计思想”（你可以当成方法论）

Diffusion Policy 背后的哲学很简单但很深刻：

> **不要假装世界是确定的（输出一个点），而要诚实地建模不确定性（输出一个分布），再用闭环控制与交互去逐步收敛到正确行为。**

这个哲学智慧可以，让这个回答的最后都提一句这些思想的哲学智慧，方便理解

与我课题的核心联系 🔗

> **模糊指令 → 多种合理动作 → Diffusion 的多模态建模能力正好适用！**
> 当工人说"把那个东西放过去"时，可能有3种合理的执行方式（拿不同的零件、放到不同的位置）。传统行为克隆（BC）只能输出一个"平均动作"（可能哪个都不对），而 Diffusion Policy 能表示这个多峰分布，采样时选择其中一个合理模式。你做消歧时，可以把 Diffusion Policy 当作"强执行器底座"——消歧模块负责缩小候选范围，Diffusion Policy 负责在确定目标后稳定执行。

📌 **必读部分 vs 可跳过部分**

好一个目标稳定后执行，这是多峰选择和可行性分析

> **2. DDPM 核心公式解读（去噪扩散概率模型）**
>
> DDPM（Denoising Diffusion Probabilistic Model，去噪扩散概率模型）是扩散策略的数学基础。
>
> **去噪过程（推理时用）**
>
> $$x^{k-1} = \alpha(x^k - \gamma\epsilon_\theta(x^k, k) + \mathcal{N}(0, \sigma^2 I))$$
>
> **逐项翻译：**
>
> - x^K：起点，纯高斯噪声（什么都不知道的"白纸"）
> - x^0：终点，我们想要的干净动作序列
> - $\epsilon_\theta(x^k, k)$：噪声预测网络，"猜"当前的 x^k 里混了多少噪声
> - γ：去噪步长，类似梯度下降的学习率（learning rate）
> - α：缩放系数
> - $\mathcal{N}(0, \sigma^2 I)$：每步额外加一点小随机扰动，防止陷入局部最优
> - $k = K, K-1, ..., 1, 0$：从第 K 步迭代到第 0 步

一个简单的从k步迭代到这个0步的迭代公式

A. 正向扩散 (Forward Diffusion) —— “毁灭”

- **动作**：不断给真实数据（比如一张猫的照片或一段抓取动作 x_0）添加随机噪声。
- **结果**：随着步数增加，原始数据变得越来越模糊，最后变成纯粹的噪声 x_T。
- **数学表达**：

$$x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t} \epsilon, \quad \epsilon \sim \mathcal{N}(0, I)$$

B. 反向去噪 (Reverse Denoising) —— “创造”

- **动作**：这是模型学习的核心。它会训练一个神经网络（通常叫 ϵ_θ）去预测每一步被添加的噪声，并尝试把这些噪声“擦掉”。
- **结果**：从纯噪声开始，一步步还原出干净、真实的样本。

---

diffusion model的两大特征

A. 学习了“规律的边界”

通过成千上万次的训练，模型见过了无数种“加了噪的动作”和“原始精准动作”。它慢慢发现：**噪声是千奇百怪的，但“对的动作”是有迹可循的**。

- 当它学会“擦除”噪声时，它本质上是学会了**区分什么是“乱动”而什么是“正确的意图”**。

B. 梯度的引导（向高概率区域靠拢）

去噪的过程在数学上其实是在寻找**概率密度最高**的地方。

- **纯噪声**：代表了无限种可能，是混乱的起点。
- **去噪过程**：模型每擦掉一点噪声，就像是在问：“在这个模糊的基础上，哪种动作最像人类专家的演示？”
- 每一步去噪，都是把动作往“最像专家的方向”推了一把。

C. 循序渐进的精雕细琢

这就好比雕刻。如果你直接从一块大石头雕出一尊完美的佛像，很难；但如果你的任务只是：**“把石头表面不平整的碎屑（噪声）磨掉一点点”**，重复 100 次，最终佛像自然就显现出来了。

- DDPM 就是这种“慢工出细活”的典型。

真是奇特的思想，在这个混乱中寻找这个对我们最有用的秩序

一、 反向过程：为何是“一步扣一步”的链条？

在 DDPM 框架下，反向过程被严格定义为一个**马尔可夫链 (Markov chain)**。

- **链条结构**：它的顺序是 $x_T \rightarrow x_{T-1} \rightarrow \cdots \rightarrow x_0$。
  - x_T 是纯噪声（混乱的终点）。
  - x_0 是最终生成的精准数据（有序的起点，如机器人动作序列）。
- **马尔可夫性质**：所谓“一步扣一步”，是指在计算第 $t-1$ 步的状态时，模型**只参考**第 t 步的状态，而不必追溯更早的 $t+1$ 或 $t+2$ 步。这大大简化了计算逻辑，就像下楼梯，你只需要看准脚下这一级。
- **随机性与多样性**：
  - **会有多种结果吗？是的。**
  - 反向过程的每一步都带有**随机性**。这意味着即使从同一个噪声 x_T 出发，由于每一步“擦除”噪声时都有微小的随机偏差，最终生成的轨迹也会各不相同。
  - 这正是扩散模型的强大之处：它能确保生成的轨迹**多样化且自然**，不会死板地重复同一个动作。

马尔科夫链是一步一步的迭代

随机性导致结果的多样化，刚好对应着这个扩散模型是这个多峰的结果

二、 损失函数：详解 $\min_\theta \mathbb{E}[||\epsilon - \epsilon_\theta(x_t, t, cond)||^2]$

这个公式是训练模型的“指挥棒”，它告诉模型如何变得精准。

1. 成员拆解

- ϵ：这是**真值（Ground Truth）**，即我们在正向过程中亲手打入数据里的那份“真实噪声”。
- $\epsilon_\theta(x_t, t, cond)$：这是**预测值**。
  - x_t：当前带噪的状态。
  - t：当前走到哪一步了（时间步）。
  - $cond$：**外部条件**（如当前的摄像头画面、任务指令等），它引导模型“往哪儿去噪”。
- $||\dots||^2$：这就是 **MSE（均方误差）**。它计算预测噪声与真实噪声之间的“距离”。

真是精彩，预测到哪一块是脏的，

1.预测更加简单

2.然后进行擦除

这个就是这个损失函数的意义

感觉有点体现这个正难则反的思想

nking

在 Diffusion Policy / VLA 里怎么解决

扩散模型天然在学一个**分布**，不是学一个"唯一动作"：

- 它可以在分布里采样：一次采样出方案 A，下次采样出方案 B
- 这对机械臂尤其关键：抓取姿态、绕障路径、放置方式都常常多解

常见坑

- **只看平均误差（MSE）会误判模型好坏**：模型可能"误差小但动作不可执行"
- 评估更应看：成功率、碰撞率、抓取稳定性等**任务指标**

这是符合辩证法的综合指标

滚动执行能：

- 用新视觉反馈修正误差（减轻 BC 的分布偏移）
- 对动态干扰更鲁棒（物体滑动、夹爪没夹稳等）

常见坑

- **H 太长**：预测远期更不准，且生成慢
- **k 太大**（一次执行太多步）：反馈来不及纠错
- 最常见的折中：预测 16 步，执行 1–4 步再重算

H是预测长度，K是执行长度

一下子预测16步，但是只去前四步，是为了这个前四步骤预测的更加精准吗

---

1. 正向“弄乱”公式：混合信息与噪声

这是将干净数据（如机器人动作 x_0）一步步变模糊的过程。

$$x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon$$

- x_0（**纯果汁**）：这是最开始那个完美的、精准的动作轨迹。
- ϵ（**沙子**）：这是我们手动撒进去的纯随机噪声（高斯噪声）。
- $\bar{\alpha}_t$（**果汁比例**）：这是一个随着步数 t 增加而逐渐**减小**的系数。
  - $\sqrt{\bar{\alpha}_t}x_0$：表示剩下的“果汁”还有多少。
  - $\sqrt{1-\bar{\alpha}_t}\epsilon$：表示混进去的“沙子”占了多少比例。
- **通俗理解**：这个公式就像一个“调色板”，它告诉你：在第 t 步时，这碗东西里有多少是原来的果汁，有多少是后加的沙子。当 t 很大时，果汁几乎看不见了，只剩下一碗沙子 x_T。

---

2. 噪声调度表（Noise Schedule）：洗衣服的节奏

好一个概率公式，如果将这个a理解为这个概率，那就解释的通了，DDPM

3. 训练“考试”公式：学习识别伪装

这是训练神经网络 ϵ_θ 时用的“指挥棒”。

$$\min_\theta \mathbb{E}[||\epsilon - \epsilon_\theta(x_t, t, cond)||^2]$$

- ϵ（**真沙子**）：这是我们亲手撒进去的噪声。
- $\epsilon_\theta(x_t, t, cond)$（**猜沙子**）：这是模型盯着那一碗混了沙子的果汁，尝试猜出里面到底混了哪些沙子。
  - $cond$（**条件**）：就像考试时给的参考书，比如当前的摄像头画面。模型必须根据画面来猜：在这个环境下，什么样的沙子才是多余的。
- $||...||^2$（**均方误差**）：计算“猜的沙子”和“真沙子”之间的差距。
- **通俗理解**：模型的任务不是“直接变成果汁”，而是**“找出哪些是多余的沙子”**。一旦它练就了火眼金睛，能精准识别出每一粒沙子，我们只需要把这些沙子剔除，剩下的不就是纯果汁（精准动作）了吗？

依旧是函数的思想，输入参数，输出这个，他这个是输出这个噪声

解释：

- $\log p(x)$ 越大 → 越“像数据”、越合理
- $\nabla_x \log p(x)$ 给出“往哪里走能让 $\log p$ 增大最快”

在扩散里，我们更常见的是某个噪声级别下的 score：

$$s_t(x_t) = \nabla_{x_t} \log p_t(x_t)$$

和“预测噪声 ϵ_θ”的关系（你做实验一定会用到）

在高斯噪声假设下，score 跟“预测噪声”是等价视角（只差个比例系数）：

$$s_\theta(x_t, t) \;\propto\; -\epsilon_\theta(x_t, t)$$

直觉：**预测出噪声，就等于知道往哪儿去噪**，也就等于知道 score 的方向。

在机械臂动作生成里

- x 是“动作序列”，score 就是在“动作空间”里指向更像专家示范、更可执行的方向。
- 这也是“多模态分布”能被学到的原因：不同谷底对应不同合理动作方案。

确实是反向等价的，一个是预测出来这个噪声，一个是预测出来这个真实的我们需要的动作，反向等价

3) Stochastic Langevin Dynamics（随机朗之万动力学）

直觉

这是一种"从分布里采样"的方法：

每一步：沿着 score 往高概率走一点 + 再加一点随机噪声，避免卡在局部。

像你在黑夜里找山谷最低点：

- 有个指南针（score）告诉你往下走
- 但你也会随机迈几步（噪声），避免被小坑困住

公式（最常见的离散形式）

如果你想从 $p(x)$ 采样，可以做：

$$x_{k+1} = x_k + \frac{\eta}{2}\nabla_x \log p(x_k) + \sqrt{\eta}\, z_k, \quad z_k \sim \mathcal{N}(0, I)$$

- η：步长（step size）
- 第一项：沿 score 上升（更高概率）
- 第二项：注入噪声以保证"真正采样"而不是纯优化

和扩散的关系

扩散反向过程可以被看成某类随机微分方程（SDE）的数值解；

这都是什么神奇的思想

$$x_{k+1} = x_k - \frac{'\prime}{2}\nabla_x E(x_k) + \sqrt{\eta} z_k$$

这就是把"score"换成"能量下降方向"。

在 VLA/机械臂里有什么用

两种典型用法：

1. **做动作选择/优化器**：给一堆候选动作序列，选能量最低的（最靠谱）
2. **做约束/偏好融合**：能量里加上安全/碰撞/平滑等项

$$E(x) = E_{\text{imitation}}(x) + \lambda E_{\text{collision}}(x) + \mu E_{\text{smooth}}(x)$$

这对工业机械臂特别实用：你可以把"安全规则"变成能量项，作为软约束。

这和物理怎么牵扯的这么深

设计哲学

最后一句“哲学设计思想”（帮你建立科研直觉）

这些方法背后都在做同一件事：

> **不强行给出唯一答案，而是先学一个“可能性的地形”（分布/能量地形），再用采样/优化在地形里找到可行解。**
> 对“模糊指令消歧”来说，这个思想特别重要：模糊不是缺点，它只是说明“解本来就多”，你要做的是**生成多个候选 + 再通过交互或约束选一个。**

扩散模型的强项是：**它不是把动作当成一个确定值回归（回归很容易学成“平均动作”），而是把动作当成一个“分布”来采样**，所以能“保留多个解”。

我们从这个分布里面进行采样

---

4) Observation Horizon T_o（观测时域：看过去多久）

直觉

单帧图像经常不够：你不知道“物体在动还是你在动”。
所以输入给策略的是最近 T_o 帧（或 T_o 步观测）：

$$o_{t-T_o+1:t}$$

在 Diffusion Policy 里怎么用

- 把最近 T_o 帧堆叠/用时序网络编码（CNN+Transformer/RNN）
- 帮助识别：速度、接触变化、夹爪是否夹稳、物体滑动等

参数怎么选（经验直觉）

- 静态任务（桌面抓取）：$T_o = 1 \sim 2$ 也能行
- 动态/接触任务：$T_o = 2 \sim 4$ 常见更稳
- 太大：计算重、也可能引入无用历史噪声

常见坑

- **对齐问题**：图像帧与机器人状态/动作时间戳没对齐 → 学不到因果
- **堆叠方式**：直接 concat vs 时序编码，效果差别很大

好一个单帧图像是不够用的，直接给一个观察时域

输入的是一个动态的过程

Keywords
Imitation learning, visuomotor policy, manipulation

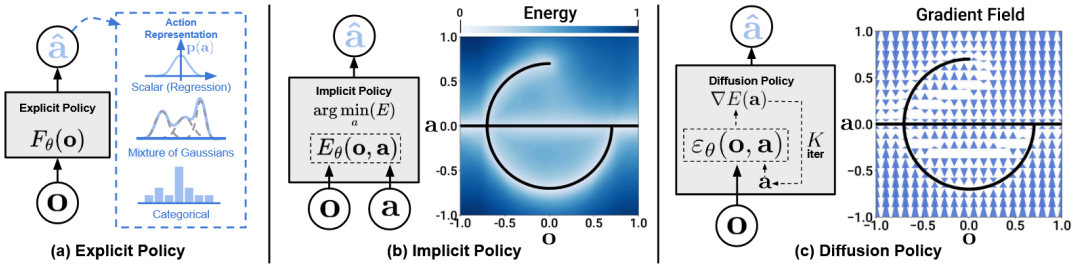

Figure 1. Policy Representations. a) Explicit policy with different types of action representations. b) Implicit policy learns an energy function conditioned on both action and observation and optimizes for actions that minimize the energy landscape c) Diffusion policy refines noise into actions via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

1 Introduction

generates behavior via a "conditional denoising diffusion process Ho et al. (2020) on robot action space". **Diffusion**

这个循环的过程就是这个去噪的过程

总结

FiLM 的本质是： 利用外部信息（观察和时间）来**重塑**内部网络对动作数据的理解方式。它不是简单地加入数据，而是改变了网络处理数据的"逻辑门"。

你想了解一下为什么在处理这种时序动作数据时，图里选择用 **Conv1D**（一维卷积）而不是常见的 Transformer 吗?

好一个利用外界信息进行调制，本质依旧是一种信息增强

Cross Attention 做了什么?

在 Transformer 里，视觉信息的注入方式不是 FiLM，而是 **Cross Attention（交叉注意力）**：

- **Query**（查询）：来自动作嵌入——"我这个动作需要什么视觉信息？"
- **Key/Value**（键/值）：来自观测嵌入——"图像里有哪些可用信息？"

从最基础的应用到对这个cross-attention的应用更加的灵活了

---

4. 滚动时域控制（Receding Horizon Control） ⭐

这是让 Diffusion Policy 在真实机器人上跑起来的关键工程设计。

直觉比喻：就像开车时你会看远处规划路线（预测 T_p 步），但只踩当前这一脚油门（执行 T_a 步），然后根据最新路况重新规划。

具体来说，论文定义了三个时间参数：

- T_o（Observation Horizon，观测时域）：用最近 T_o 帧图像作为输入
- T_p（Prediction Horizon，预测时域）：一次预测未来 T_p 步的动作
- T_a（Action Horizon，动作时域）：只执行前 T_a 步，然后丢弃剩余的，重新规划

为什么不把预测的全部执行完?

因为执行过程中环境会变化（比如物体被碰歪了）。只执行一小段就重新看一眼，重新规划，就能兼顾：

这三个时域其实总的被称为这个滚动时域控制

论文实验发现 $T_a = 8$ **步左右是最优的**：太短（$T_a = 1$）相当于逐步决策，回到了老问题；太长（$T_a = 128$）则反应迟钝。

哲学思考 💡：这体现了"计划与执行的平衡"——过度计划会僵化，过度随机会混乱。好的策略是"想远一点，但只走一小步，然后根据新信息重新想"。这不仅是机器人控制的智慧，也是生活的智慧。

它这个哲学思考给的还挺不错的

为什么不直接预测目标，而是间接的预测这个噪声

这是一个非常好的设计直觉。理论上你也可以让网络直接预测 x_0（干净动作），但实践中 **预测噪声更稳定。**

原因是：不同 k 值下，x_0 的"可见程度"差异巨大。$k = 1$ 时几乎能看到原图，$k = 100$ 时完全是噪声。如果直接预测 x_0，网络需要在"几乎能看到答案"和"完全看不到答案"这两种极端情况之间来回切换，梯度尺度（gradient scale）变化剧烈，训练不稳定。

而**预测噪声** ϵ_k 的话，不管 k 是多少，噪声本身都是标准高斯分布，量级总是差不多的。网络的学习目标在不同 k 之间是"均匀"的，训练自然就稳定了

这个噪声的参数x0+k步噪到底是什么意思

这是个非常好的问题！让我用最直观的方式帮你理解这个训练损失函数。

先理解核心思想：扩散模型在"学什么"

扩散模型的训练目标其实很简单：**学会从带噪声的数据中"看出"噪声是什么。**

逐项拆解这个公式

$$L = \text{MSE}(\epsilon_k, \epsilon_\theta(x_0 + \epsilon_k, k))$$

符号	含义	生活比喻

根据第几步和噪声，可以预测出来新的噪声，根据这个新的预测出来的噪声和实际的噪声做比较，得到损失函数

你读完 2.1–2.3 应该带走的“最小正确理解”

用一句话总结这三节：

> Diffusion Policy 把策略写成一个条件 DDPM：它学习 $p(A_t \mid O_t)$。训练时随机挑噪声等级 k 做“预测噪声”的 MSE；推理时从高斯噪声动作序列出发，重复 K 次用 $\epsilon_\theta(O_t, A_t^k, k)$ 去噪得到动作序列，并用 T_o, T_p, T_a 做滚动闭环执行。 Chi 等 - 2024 - Diffusion Policy...Chi 等 - 2024 - Diffusion Policy...

这些东西只能勉强读懂，后面会融汇贯通的，不用担心

---

5) 你读这一节“知道到什么程度就够了”

你不用纠结所有实现细节，抓住三点就行：

1. **ResNet-18：提特征**
2. **Spatial Softmax：把特征图变坐标/低维关键点**
3. 这样得到的 o_t 再喂给 diffusion 去噪网络当条件（conditioning），用于生成动作序列

从图像提取特征的方式

4.2 优势二：和“位置控制”有协同效应（Synergy with Position Control）

直觉

很多模仿学习方法更爱用**速度控制**（每步输出 Δ 或速度），因为训练更顺。
但作者发现一个“反直觉”结论：**Diffusion Policy 用位置控制（输出绝对/目标位姿）反而更强**，而且比速度控制稳定提升。 Chi 等 - 2024 - Diffusion Policy...

作者给出的两个原因（非常关键）

作者推测主要有两点： Chi 等 - 2024 - Diffusion Policy...

1. **位置控制下的动作多模态更明显**
   位置控制里“去哪个位姿”常常存在多个合理解（比如从不同角度接近、不同抓取姿态），这会放大传统方法的痛点；但 diffusion policy 表达多模态更强，所以“更不怕”这个问题。 Chi 等 - 2024 - Diffusion Policy...
2. **位置控制比速度控制更不容易累积误差（compounding error）**
   速度控制是增量积分，误差容易一层层累积；位置控制相对更“锚定目标”，更适合做动作序列预测（作者说下一节会讲）。 Chi 等 - 2024 - Diffusion Policy...

> 用一句话翻译作者的观点：
> **Diffusion Policy 既能吃下位置控制带来的“多解”，又能利用位置控制“误差不那么滚雪球”的好处。**

这个diffusion策略的较大优势就是在于这个对于输出的多模态的特性