Probabilistic Action Diffusion: Foundations and Insights for Generative Policy Modeling

Abstract

In this paper I review the core theory related to the application of probabilistic generative modeling especially Diffusion Models in Visual-Motor manipulation and walk through the implementation of this technique. The main framework for evaluating and testing the outcomes of this paper is simulated PushT enviornment, however the discussed approach might be applied to any action generation task with little or no modification.

Introduction

Recent advances in hardware and perception are gradually shifting robotic systems from highly structured industrial environments into households, warehouses, and public spaces [3,]. In these unstructured settings, the distribution of possible actions is both high-dimensional and inherently multimodal: a single objective, such as opening a door, may allow for many valid motion sequences that differ in contact timing, grasp pose, or force profile [2, 5, 6]. Traditional planners and behavior cloning policies typically estimate only a single "best" trajectory, thereby collapsing this diversity into a brittle average behavior that quickly degrades when the environment deviates from its training manifold [4, 5]. For example, in behavior cloning tasks [10], models that predict only the next action (e.g., BC-RNN, BET, IBC) can produce erratic motions when multiple equally valid trajectories are present in the demonstrations, or become stuck when "idle" actions (pauses) are demonstrated. In contrast, these scenarios require a control policy that explicitly represents a probability distribution over future actions, enabling the robot to generate multiple plausible candidates, evaluate them in real time, and quickly adapt as external conditions change [4, 6, 7].

In this context, Diffusion Models (DMs) have emerged as a powerful and versatile technology [8, 5, 9] that offers flexible modeling of high-dimensional data. They possess an exceptional ability to represent multimodal distributions, which is a key advantage in many manipulation tasks where multiple equally acceptable solutions exist. Modeling all of these "modes" (distribution peaks representing different possible outcomes) enhances generalization and robustness in robotic behavior [1, 2, 4, 5, 5, 7, 8, 9]. Diffusion models also exhibit robustness to high-dimensional input and output spaces and show impressive training stability [7].

Applications of diffusion models in robotics include parameterizing control/RL policies for complex tasks such as robot manipulation and human behavior imitation [1, 2, 7]. In these cases, the goal is to learn a conditional probability distribution over actions given the current system state.

Theory

Diffusion models is the family of probabilistic generative models originally came from the thermodynamics domain. The idea of these models is to progeresively destruct data by adding gausian noise, then learn the reverse process for sample generation. after the pretrain the model represent the probabilistic distribution of data which might be navigated with various conditioning stategies. The current research on diffusion models mostly based on three variations: denoising diffusion probabilistic models (DDPMs) [11], score-based generative models (SGMs) [12, 13], and stochastic differential equations (Score SDEs) [14], where I will focus on DDPMs.

DDPM

A Denoising Diffusion Probabilistic Model (DDPM) is defined by two Markov chains: a **forward process**, which gradually perturbs input data into pure noise, and a **reverse process**, which learns to transform noise back into data through learned transitions. The goal of the forward chain is to convert a complex data distribution $q(x_0)$ into a tractable prior—usually a standard Gaussian $\mathcal{N}(0, I)$. Then, the reverse chain, parameterized by deep neural networks, attempts to learn the inverse mapping by iteratively denoising a sampled noise vector.

Forward Process and Reparameterization

Formally, we define the forward diffusion process as a sequence of latent variables $x_1, x_2, \ldots, x_T$ generated from an initial data point $x_0 \sim q(x_0)$, via Gaussian transitions:

$$q(x_t \mid x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t} \cdot x_{t-1}, \beta_t I)$$

where $beta_t \in (0, 1)$ is a small positive scalar controlling the amount of noise injected at each step, often called the **noise scheduler**. These transitions are handcrafted and fixed unlike the reverse process, which is learned.

Using the Markov assumption, the full joint distribution of the forward process can be factorized as:

$$q(x_{1:T} \mid x_0) = \prod_{t=1}^{T} q(x_t \mid x_{t-1})$$

Each step slightly corrupts the sample, and as $t \to T$, the sample becomes indistinguishable from isotropic Gaussian noise.

Cumulative Noise and Reparameterization

For efficiency and training stability, it is useful to derive a closed-form expression for the marginal distribution $q(x_t \mid x_0)$, allowing us to compute x_t from x_0 in a single step without sequential sampling. This is possible because the Gaussian perturbations accumulate in a structured way.

Define:

$$\alpha_t = 1 - \beta_t, \quad \bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s$$

Then, by unrolling the composition of Gaussian noise over (t) steps, we obtain:

$$q(x_t \mid x_0) = \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t} \cdot x_0, (1 - \bar{\alpha}_t)I)$$

This allows us to efficiently sample noisy inputs using the **reparameterization trick**:

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

The reparameterization trick enables us to write a stochastic sample x_t as a deterministic function of the model input x_0, timestep t, and random noise ϵ. This is essential for optimizing the model via gradient-based learning, as it separates the stochasticity from the learnable parameters and makes the entire sampling process differentiable.

To summarize:

- The forward process corrupts the data with Gaussian noise according to a fixed schedule.
- The marginal distribution $q(x_t \mid x_0)$ is Gaussian with closed-form mean and variance.
- The reparameterization trick allows us to sample x_t directly and backpropagate gradients through the sampling step.

This formulation sets the foundation for training the reverse Markov chain to undo the forward noise process. In the next section, we will define the parameterized reverse transitions and the objective used to learn them.

Reverse Process and Training Objective

While the forward process $q(x_t \mid x_{t-1})$ is fixed and designed to gradually corrupt the data, the **reverse process** aims to learn how to denoise — that is, to recover the data distribution by inverting the noise trajectory. This is done by learning a reverse Markov chain of the form:

$$p_\theta(x_{t-1} \mid x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t))$$

Here, the mean μ_θ and covariance Σ_θ are predicted by neural networks, and θ denotes the model parameters. The prior at the final step is set to a standard normal distribution:

$$p(x_T) = \mathcal{N}(x_T; 0, I)$$

A complete sample is generated by drawing $x_T \sim p(x_T)$ and applying the learned transitions repeatedly until reaching x_0.

To learn this reverse chain, we train the model so that the reverse trajectory matches the true posterior of the forward process. Formally, this is done by minimizing the Kullback–Leibler divergence between the joint forward and reverse distributions:

$$\mathcal{KL}(q(x_{0:T}) \,\|\, p_\theta(x_{0:T})) = \mathbb{E}_q \left[\log \frac{q(x_{0:T})}{p_\theta(x_{0:T})}\right]$$

By applying the Markov factorization for both q and p_θ, we get:

$$\mathcal{KL}(q(x_{0:T}) \,\|\, p_\theta(x_{0:T})) = \mathbb{E}_q \left[-\log p(x_T) - \sum_{t=1}^{T} \log p_\theta(x_{t-1} \mid x_t)\right] + \text{const}$$

This is equivalent (up to constants) to minimizing the negative **variational lower bound (VLB)** of the data log-likelihood:

$$-\log p_\theta(x_0) \le \mathcal{KL}(q(x_{0:T}) \,\|\, p_\theta(x_{0:T})) =: \mathcal{L}_{\text{VLB}}$$

However, directly optimizing this bound is cumbersome, since it requires learning $\Sigma_\theta(x_t, t)$ and evaluating log-likelihood terms. Instead we could simplify the objective by letting the model predict the added noise ϵ in the forward process and optimize the following simplified loss[15]:

$$\mathcal{L}_{\text{simple}} = \mathbb{E}_{x_0, t, \epsilon} \left[\lambda(t) \cdot \|\epsilon - \epsilon_\theta(x_t, t)\|^2\right]$$

where:

- $t \sim \mathcal{U}(\{1, \ldots, T\})$ is a uniformly sampled timestep,
- $x_t = \sqrt{\bar{\alpha}_t} \cdot x_0 + \sqrt{1 - \bar{\alpha}_t} \cdot \epsilon$ with $\epsilon \sim \mathcal{N}(0, I)$,
- $\lambda(t)$ is a weighting function that controls how much each timestep contributes to the loss.

This formulation is equivalent (up to scaling) to denoising score matching over multiple noise levels, and it allows efficient optimization with stochastic gradient descent. In practice, $\lambda(t)$ is often chosen to balance training across early and late timesteps.

The reverse process is trained to approximate the time-reversal of the forward diffusion. A variational lower bound is used to derive a tractable training objective. In practice, the loss reduces to predicting the forward noise ϵ, making the training efficient and stable.

Implementation Details

coming in next iteration

Literature

1. Diffusion Models: A Comprehensive Survey of Methods and Applications
2. Diffusion Models for Robotic Manipulation: A Survey
3. manipulation.pdf
4. diffusion_policy_2023.pdf
5. diffusion models for robotic manipulation:asurvey - arXiv
6. [research] Diffusion Policy: Visuomotor Policy Learning via Action Diffusion
7. Visuomotor Policy Learning via Action Diffusion
8. An Overview of Diffusion Models: Applications, Guided Generation, Statistical Rates and OptimizationEmails: {minshuochen, jqfan, mengdiw}@princeton.edu, songmei@berkeley.edu - arXiv
9. Diffusion Models for Robotics
10. Is Behavior Cloning All You Need? Understanding Horizon in Imitation Learning
11. Denoising diffusion probabilistic models.
12. Yang Song and Stefano Ermon. 2019. Generative modeling by estimating gradients of the data distribution.
13. Yang Song and Stefano Ermon. 2020. Improved techniques for training score-based generative models.
14. Maximum likelihood training of score-based diffusion models.