Continuous-Time Diffusion Policies for Visuomotor Control: A Stochastic Calculus Perspective

Angikar Ghosal , Nils Kuhn
Professor: Jose Blanchet
Management Science and Engineering 322 – Stochastic Calculus and Control

Abstract

Recent advancements in visuomotor control have been driven by Diffusion Policies, which model the action distribution as a discrete-time denoising process. In this project, we reinterpret these methods through the rigorous lens of continuous-time stochastic calculus, formulating them as controlled stochastic differential equations (SDEs). We rigorously define, analyze, and compare three distinct classes of SDEs—Variance Preserving (VP), Variance Exploding (VE), and Critically Damped Langevin Dynamics (CLD)—and their corresponding numerical solvers. For each process, we derive the forward and reverse-time SDEs, the probability flow ODE, and the associated score-matching objective for training a neural network policy. Our experiments on the PushT manipulation benchmark demonstrate that the choice of stochastic process fundamentally alters control performance. We find that the VP-SDE significantly outperforms VE and CLD formulations, achieving higher average goal coverage and smoother trajectories.

1 Mathematical Framework

Let $(\Omega, \mathcal{F}, \mathbb{P})$ be a probability space with a filtration $\{\mathcal{F}_t\}_{t\in[0,T]}$. A visuomotor policy is a conditional probability distribution $\pi(\mathbf{a} \mid \mathbf{o})$ over an action space $\mathcal{A} \subset \mathbb{R}^D$ conditioned on an observation $\mathbf{o} \in \mathcal{O}$. In this work, we model this distribution using a generative process defined by a controlled stochastic differential equation. The core idea is to transform a complex, unknown data distribution (expert actions) into a simple, tractable prior distribution (e.g., Gaussian noise) via a forward SDE, and then learn to reverse this process to generate new data. The equation of the forward process in our action space is given as:

$$d\mathbf{a}_t = f(\mathbf{a}_t, t)\, dt + g(t)\, d\mathbf{W}_t, \quad \mathbf{a}_0 \sim \pi(\cdot \mid \mathbf{o}), \tag{1}$$

The backwards process can then be derived as:

$$d\mathbf{a}_t = \left[f(\mathbf{a}_t, t) - g(t)^2 \nabla_{\mathbf{a}_t} \log p_t(\mathbf{a}_t)\right]\, dt + g(t)\, d\bar{\mathbf{W}}_t, \tag{2}$$

The definition of the Forward and Reverse Processes as Itô diffusion is given in Appendix A. The reverse SDE (2) can be interpreted from a control theory perspective. (Tzen and Raginsky [2019], Berner et al. [2022]) Consider the uncontrolled process $d\mathbf{x}_t = f(\mathbf{x}_t, t)\, dt + g(t)\, d\bar{\mathbf{W}}_t$. The term $u(\mathbf{a}_t, t) = -g(t)^2 \nabla_{\mathbf{a}_t} \log p_t(\mathbf{a}_t)$ acts as a feedback control law that steers the state $\mathbf{a}_t$ towards regions of high probability under the target distribution. Learning the score function $s_\theta = \nabla_{\mathbf{a}_t} \log p_t(\mathbf{a}_t)$ is thus equivalent to learning an optimal control policy that minimizes the Kullback–Leibler (KL) divergence between the trajectory distribution of the controlled process and that of the time-reversed expert data process. Furthermore, the evolution of the density $p_t(\mathbf{a})$ is governed by the Fokker-Planck equation, and the reverse SDE provides the control required to shape this density from a simple prior p_T into the complex data distribution p_0.

2 Stochastic Processes for Control

We formulate the visuomotor policy as a time-reversal of a forward diffusion process. We investigate three specific SDE classes. For each, we define the forward dynamics, the reverse generative process, and the specific loss function $\mathcal{L}(\theta)$ derived from score matching. (Hyvärinen [2005], Vincent [2011])

2.1 Variance Preserving (VP) SDE

This process is the continuous-time limit of the DDPM schedule. It creates a mean-reverting process that keeps the data variance bounded, ensuring stable training gradients. The detailed mathematical derivations are in in Appendix B.

Definition 1 (VP Forward SDE)**.** *Let $\beta(t)$ be a positive noise schedule. The forward process is an Ornstein-Uhlenbeck process:*

$$d\mathbf{a}_t = -\frac{1}{2}\beta(t)\mathbf{a}_t\, dt + \sqrt{\beta(t)}\, d\mathbf{W}_t \tag{3}$$

The transition kernel $p_{0t}(\mathbf{a}_t|\mathbf{a}_0)$ is Gaussian with mean $\boldsymbol{\mu}_t = \mathbf{a}_0 e^{-\frac{1}{2}\int_0^t \beta(s)ds}$ and variance $\boldsymbol{\Sigma}_t = (1 - e^{-\int_0^t \beta(s)ds})\mathbf{I}$.

Training Objective: By parameterizing the score network to predict the added noise $\boldsymbol{\epsilon}$, the loss function simplifies to a weighted mean-squared error:

$$\mathcal{L}_{VP}(\theta) = \mathbb{E}_{t,\mathbf{a}_0,\boldsymbol{\epsilon}}\left[\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{a}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t, \mathbf{o})\|^2\right] \tag{4}$$

Noise Schedule: In our experiments, we use the cosine squared noise schedule for $\beta(t)$ as formulated by Nichol and Dhariwal [2021].

2.2 Variance Exploding (VE) SDE

Common in Score-Based Generative Models (SGM), this process has no drift term. The variance grows ("explodes") as $t \to T$. The detailed mathematical derivations are in in Appendix C.

Definition 2 (VE Forward SDE)**.** *Let $\sigma(t)$ be an increasing function. The process is defined as:*

$$d\mathbf{a}_t = \sqrt{\frac{d[\sigma^2(t)]}{dt}}\, d\mathbf{W}_t, \quad \mathbf{a}_0 \sim \pi_{data} \tag{5}$$

Conditioned on $\mathbf{a}_0$, the state is $\mathbf{a}_t \sim \mathcal{N}(\mathbf{a}_0, \sigma^2(t)\mathbf{I})$.

Training Objective: Due to the unbounded variance, the score-matching loss requires a weighting function $\lambda(t) = \sigma^2(t)$ to stabilize optimization:

$$\mathcal{L}_{VE}(\theta) = \mathbb{E}_{t,\mathbf{a}_0,\mathbf{a}_t}\left[\sigma^2(t)\,\|s_\theta(\mathbf{a}_t, t, \mathbf{o}) - \nabla_{\mathbf{a}_t} \log p_t(\mathbf{a}_t \mid \mathbf{a}_0)\|^2\right]. \tag{6}$$

The VE SDE just adds noise to the expert (target) actions $\mathbf{a}_0$. At inference time, however, $\mathbf{a}_0$ is unknown. In practice, we normalize expert actions so that $\mathbb{E}[\mathbf{a}_0] \approx \mathbf{0}$, and choose $\sigma(T)$ large compared to the typical action magnitude, i.e. $\|\mathbf{a}_0\| \ll \sigma(T)$ for most samples. Under these assumptions, the marginal $p_T(\mathbf{a})$ is well approximated by a zero-mean Gaussian,

$$p_T(\mathbf{a}) \approx \mathcal{N}\big(\mathbf{0}, \sigma^2(T)\mathbf{I}\big),$$

and we therefore initialize the reverse-time process by sampling $\mathbf{a}_T \sim \mathcal{N}\big(\mathbf{0}, \sigma^2(T)\mathbf{I}\big)$ as a simple prior.

Noise Schedule: In our experiments, we use the logarithmic noise schedule for $\sigma(t)$ as formulated by Song et al. [2021b].

2.3 Critically Damped Langevin Dynamics (CLD)

This second-order process introduces momentum $\mathbf{p} \in \mathbb{R}^D$, acting on the phase space $(\mathbf{a}, \mathbf{p})$. It models physical concepts like inertia and friction, providing a potentially powerful inductive bias for generating smooth robot motions. The detailed mathematical derivations are in in Appendix D.

Definition 3 (CLD Forward SDE)**.** *The system is governed by a coupled SDE in phase space:*

$$d\mathbf{a}_t = \mathbf{p}_t\, dt \tag{7}$$

$$d\mathbf{p}_t = -\gamma \mathbf{p}_t\, dt + \sqrt{2\gamma}\, d\mathbf{W}_t \tag{8}$$

Training Objective: We derived the marginal transition kernel for position $\mathbf{a}_t$ (see Appendix D) to be Gaussian with variance $\sigma^2_{\text{CLD}}(t)$. This allows us to use a score-prediction objective similar to VE:

$$\mathcal{L}_{CLD}(\theta) = \mathbb{E}_{t,\mathbf{a}_0,\mathbf{a}_t}\left[\sigma^2(t)\,\|s_\theta(\mathbf{a}_t, t, \mathbf{o}) - \nabla_{\mathbf{a}_t} \log p_t(\mathbf{a}_t \mid \mathbf{a}_0)\|^2\right]. \tag{9}$$

The generative process is different. Here, the learned score s_θ acts as a control force that steers the particle.

$$d\mathbf{a}_t = \mathbf{p}_t\, dt, \tag{10}$$

$$d\mathbf{p}_t = [-\gamma \mathbf{p}_t + \lambda(t) s_\theta(\mathbf{a}_t, t, \mathbf{o})]\, dt + \sqrt{2\gamma}\, d\mathbf{W}_t, \tag{11}$$

where $\lambda(t)$ is a time-dependent scaling factor for the control input. In practice, as mentioned by Dockhorn et al. [2022], the virtual time is flipped such that the system can be integrated forward in time from $t = 0$ to $t = T$, starting from prior samples $\mathbf{a}_0 \sim \mathcal{N}(0, \sigma_a^2\mathbf{I})$ and $\mathbf{p}_0 \sim \mathcal{N}(0, \sigma_p^2\mathbf{I})$.

Noise Schedule: The noise schedule of the CLD SDE is driven by γ. In our experiments, we chose the constant $\gamma = 3$.

3 Experimental Protocol

3.1 Diffusion Policies on PushT

We evaluate our methods on the PushT tabletop manipulation benchmark, where a planar end-effector must push an object into a target "T"-shaped configuration. The task involves contact-rich interactions, in which small changes in actions can produce qualitatively different object motions (e.g., sticking vs. sliding, different rotation outcomes). This makes the underlying dynamics highly non-linear and

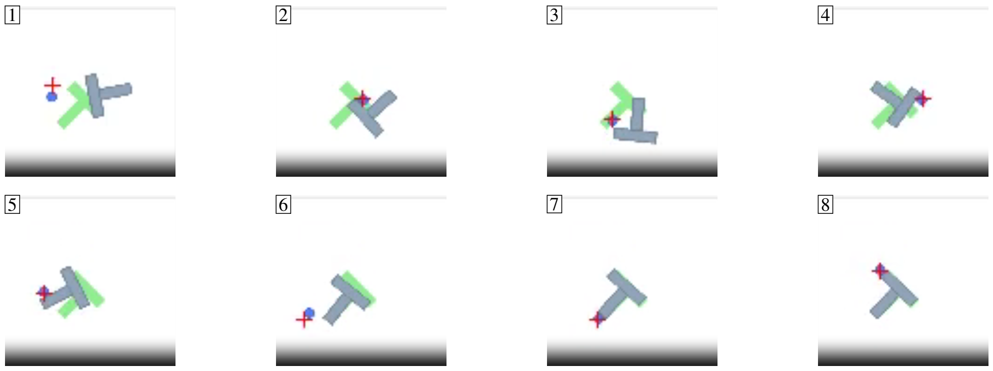

Figure 1: Successful PushT rollout: the end-effector pushes the T-shaped object from the initial pose to the green goal configuration.

discontinuous. The overall goal is to push the "T" from its initial position to the green goal position. Figure 1 visualizes a successful example of our experiments.

PushT is both challenging and informative for robotics: (i) it exhibits hybrid contact dynamics and frictional effects typical of real-world manipulation; (ii) small errors accumulate over a multi-step horizon, making long-horizon consistency critical; and (iii) success requires tight coupling between perception and control. As such, it provides a compact but demanding benchmark for assessing how different diffusion SDE formulations affect policy quality in visuomotor control.

3.1.1 Observation Space

At each time step t, the observation is $o_t = (I_{1,t}, I_{2,t}, s_t)$, with:

- $I_{1,t}, I_{2,t} \in \mathbb{R}^{3\times 84\times 84}$: the last two RGB images from a fixed camera capturing the robot, the t-shaped block and the target configuration.
- $s_t \in \mathbb{R}^2$: the robot position in the 2-dimensional state space.

The image provides global scene context and object-goal relations, while the 2D state offers precise localization of the end-effector. This combination stresses the policy's ability to fuse high-dimensional visual input with concise proprioceptive information.

3.1.2 Action Space

The policy outputs a sequence of future actions at each step $a_t \in \mathbb{R}^{2\times H}$, where each column $a_{t,i} \in \mathbb{R}^2$ (for $i \in \{1, 2, ..., H\}$ denotes the position difference for the i-th step-ahead in the horizon, similar to Zhao et al. [2023]. Similar to Chi et al. [2025], we will use $H = 8$.

This sequence-level representation enables the model to directly encode temporal correlations and multi-step strategies within a single diffusion sample, and it makes the quality of the learned reverse SDE particularly visible through the coherence and smoothness of the predicted action trajectories.

3.1.3 Diffusion Policy Architecture

Our policy adopts a two-stage architecture consisting of a visual backbone and a diffusion-based action decoder. The observation o_t is first encoded by a ResNet-18 vision backbone ($\approx$ 22M parameters), which preprocesses the RGB images. The resulting visual features are concatenated with an embedding of the 2D robot state s_t, yielding a compact contextual representation that conditions the policy.

Conditioned on this context, a Diffusion Transformer ($\approx$ 9M parameters). serves as the action decoder. The noisy action sequence $\tilde{a}_\tau \in \mathbb{R}^{2\times H}$ at diffusion time τ is tokenized along the horizon (8 tokens, one per future step), embedded into a latent space, and augmented with temporal positional encodings. Several layers of self-attention operate over this token sequence to model temporal dependencies, while the observation features and diffusion time embedding condition the transformer.

Only the Diffusion Transformer is trained to denoise the action sequence under the chosen SDE. This hierarchical design, combining ResNet-18 for high-capacity visual encoding with a comparatively lightweight diffusion transformer for sequence-level action generation, mirrors common practice in state-of-the-art autonomous robot control (Black et al. [2024], Kim et al. [2024]). Consequently, it provides an appropriate and controlled setting for isolating and analyzing the impact of different SDE formulations on the quality and consistency of the learned action distribution on PushT.

3.2 Phase 1: Training the Score Network

The first phase of the experiment is to train the neural network s_θ (or its noise-prediction counterpart $\boldsymbol{\epsilon}_\theta$) using the expert demonstration dataset. The network learns to predict the noise that corrupts a clean expert action. This process is entirely offline and does not involve running the robot.

Algorithm 1 Training a Continuous-Time Diffusion Policy

```
1:  Initialize network parameters $\theta$.
2:  **repeat**
3:      Sample an expert trajectory chunk from the dataset. Let $\mathbf{a}_0 \in \mathbb{R}^D$ be the ground-truth action
        sequence. Let $\mathbf{o}$ be the observation at the start of the sequence.
4:      Sample a continuous time $t \sim \mathcal{U}(0, T)$.
5:      Sample a noise vector $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I}_D)$.
6:      Corrupt the clean action $\mathbf{a}_0$ to create a noisy sample $\mathbf{a}_t$ using the analytical formula (forward
        kernel) for the chosen SDE:
          • **VP:** $\mathbf{a}_t \leftarrow \sqrt{\bar{\alpha}_t}\mathbf{a}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}$
          • **VE:** $\mathbf{a}_t \leftarrow \mathbf{a}_0 + \sigma(t)\boldsymbol{\epsilon}$
          • **CLD:** $\mathbf{a}_t \leftarrow \mathbf{a}_0 + \sigma_{\text{CLD}}(t)\boldsymbol{\epsilon}$
7:      Feed $(\mathbf{a}_t, t, \mathbf{o})$ into the network to get its prediction (e.g., $\boldsymbol{\epsilon}_\theta(\mathbf{a}_t, t, \mathbf{o})$).
8:      Compute the loss by comparing the network's prediction to the true target. For noise-
        predicting networks, the loss is $\mathcal{L}(\theta) = \|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta\|^2$. For score-predicting networks, it is the
        corresponding weighted score-matching loss.
9:      Update parameters $\theta$ using a gradient descent step: $\theta \leftarrow \theta - \eta\nabla_\theta\mathcal{L}(\theta)$.
10: **until** convergence
```

3.3 Phase 2: Generating Actions at Inference Time

After training, the network is frozen and deployed to control the robot. When the robot provides a new observation $\mathbf{o}$, the policy generates an action by starting with pure noise and iteratively denoising it using the trained network. This is the reverse process.

Algorithm 2 Action Generation at Inference Time (for VP/VE SDEs)

```
1:  **Input:** Trained network $s_\theta$ (or $\boldsymbol{\epsilon}_\theta$), current robot observation $\mathbf{o}$, number of integration steps $N$.
2:  **Output:** A clean action sequence $\mathbf{a}_0$.
3:  **Step 1: Initialization.** Sample a random action from the prior distribution: $\mathbf{a}_T \sim \mathcal{N}(0, \mathbf{I})$.
4:  **Step 2: Discretization.** Define the time schedule for denoising: $T = t_0, t_1, \ldots, t_N = 0$.
5:  Let $\mathbf{a}_{\text{current}} \leftarrow \mathbf{a}_T$.
6:  **for** $i = 0$ to $N - 1$ **do**                                    ▷ Iterate from high noise to low noise
7:      Let current time be $t_i$.
8:      **Step 3: Prediction.** Query the network with the current noisy action, time, and observation
        to get the predicted score or noise. E.g., $\boldsymbol{\epsilon}_{\text{pred}} = \boldsymbol{\epsilon}_\theta(\mathbf{a}_{\text{current}}, t_i, \mathbf{o})$.
9:      **Step 4: Update.** Take one small step "backwards in time" using a numerical solver to get a
        slightly cleaner action, $\mathbf{a}_{\text{next}}$. This step uses the prediction from the network to guide the update.
10:         (The specific update formula depends on the SDE and solver, as detailed later).
11:     $\mathbf{a}_{\text{current}} \leftarrow \mathbf{a}_{\text{next}}$.
12: **return** $\mathbf{a}_{\text{current}}$ (which is now the fully denoised action $\mathbf{a}_0$).
```

3.4 Numerical Integration Schemes

To generate an action, we numerically solve the generative differential equation by discretizing the time interval $[0, T]$ into N steps: $T = t_0, t_1, \ldots, t_N = 0$. The update rules for moving from a state $\mathbf{a}_{t_i}$ at the current time t_i to a new state $\mathbf{a}_{t_{i+1}}$ at the next time t_{i+1} depend on the SDE formulation and the chosen solver. For VP and VE SDEs, we integrate backward in time, so the time step $\Delta t = t_{i+1} - t_i$ is negative. For CLD, we integrate forward, so Δt is positive.

3.4.1 Euler Method (ODE Solver)

The general form of the forward Euler update is:

$$\mathbf{a}_{t_{i+1}} = \mathbf{a}_{t_i} + F(\mathbf{a}_{t_i}, t_i)(t_{i+1} - t_i)$$

The specific form of the velocity field $F(\mathbf{a}_t, t)$ is (Song et al. [2021a], Karras et al. [2022]):

- **For VP SDE:** The network $\boldsymbol{\epsilon}_\theta(\mathbf{a}_t, t, \mathbf{o})$ predicts the noise.

$$F_{\text{VP}}(\mathbf{a}_t, t) = -\frac{1}{2}\beta(t)\left[\mathbf{a}_t - \frac{\boldsymbol{\epsilon}_\theta(\mathbf{a}_t, t, \mathbf{o})}{\sqrt{1 - \bar{\alpha}_t}}\right]$$

- **For VE SDE:** The network $s_\theta(\mathbf{a}_t, t, \mathbf{o})$ predicts the score. Let $g(t)^2 = \frac{d[\sigma^2(t)]}{dt}$.

$$F_{\text{VE}}(\mathbf{a}_t, t) = -\frac{1}{2}g(t)^2 s_\theta(\mathbf{a}_t, t, \mathbf{o})$$

- **For CLD (Deterministic):** We solve the coupled system of first-order ODEs by removing the noise term from the generative SDE. The state is $(\mathbf{a}_t, \mathbf{p}_t)$.

$$\begin{aligned}
\mathbf{a}_{t_{i+1}} &= \mathbf{a}_{t_i} + \mathbf{p}_{t_i}(t_{i+1} - t_i) \\
\mathbf{p}_{t_{i+1}} &= \mathbf{p}_{t_i} + \left[-\gamma\mathbf{p}_{t_i} + s_\theta(\mathbf{a}_{t_i}, t_i, \mathbf{o})\right](t_{i+1} - t_i)
\end{aligned}$$

3.4.2 Heun's Method (ODE Solver)

The general form is a two-step predictor-corrector update:

$$\begin{aligned}
\tilde{\mathbf{a}} &= \mathbf{a}_{t_i} + F(\mathbf{a}_{t_i}, t_i)(t_{i+1} - t_i) \\
\mathbf{a}_{t_{i+1}} &= \mathbf{a}_{t_i} + \frac{t_{i+1} - t_i}{2}\left[F(\mathbf{a}_{t_i}, t_i) + F(\tilde{\mathbf{a}}, t_{i+1})\right]
\end{aligned}$$

The function F is identical to the one defined for the Euler Method for both the VP and VE SDEs.

3.4.3 Euler-Maruyama Method (SDE Solver)

- **For VP and VE SDEs (Backward Integration):** The update rule is

$$\mathbf{a}_{t_{i+1}} = \mathbf{a}_{t_i} + \mu(\mathbf{a}_{t_i}, t_i)(t_{i+1} - t_i) + \sigma(t_i)\sqrt{t_i - t_{i+1}}\,\mathbf{z}, \quad \mathbf{z} \sim \mathcal{N}(0, \mathbf{I})$$

  The specific drift μ and diffusion σ are:

  - **For VP SDE:**

$$\begin{aligned}
\mu_{\text{VP}}(\mathbf{a}_t, t) &= -\frac{1}{2}\beta(t)\mathbf{a}_t + \frac{\beta(t)}{\sqrt{1 - \bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\mathbf{a}_t, t, \mathbf{o}) \\
\sigma_{\text{VP}}(t) &= \sqrt{\beta(t)}
\end{aligned}$$

  - **For VE SDE:**

$$\begin{aligned}
\mu_{\text{VE}}(\mathbf{a}_t, t) &= -g(t)^2 s_\theta(\mathbf{a}_t, t, \mathbf{o}) \\
\sigma_{\text{VE}}(t) &= g(t) = \sqrt{\frac{d[\sigma^2(t)]}{dt}}
\end{aligned}$$

- **For CLD (Forward Integration):** We integrate the generative SDE forward in time. The update rule for the coupled system is:

$$\begin{aligned}
\mathbf{a}_{t_{i+1}} &= \mathbf{a}_{t_i} + \mathbf{p}_{t_i}(t_{i+1} - t_i) \\
\mathbf{p}_{t_{i+1}} &= \mathbf{p}_{t_i} + \left[-\gamma\mathbf{p}_{t_i} + s_\theta(\mathbf{a}_{t_i}, t_i, \mathbf{o})\right](t_{i+1} - t_i) + \sqrt{2\gamma}\sqrt{t_{i+1} - t_i}\,\mathbf{z}
\end{aligned}$$

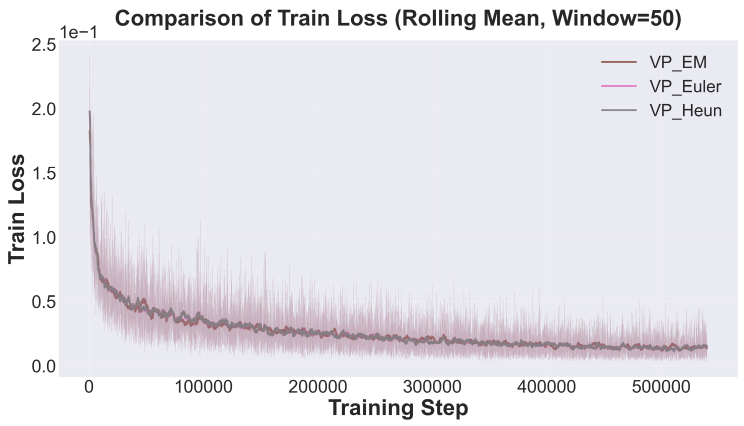

Figure 3: Smoothed Training Loss for VP-SDE

4 Results

Figures 3 and 5 highlight that the choice of SDE strongly affects both the scale and dynamics of optimization.

Because the loss magnitude depends on the noise parametrization, the absolute values must be interpreted with care: the training losses obtained with VE-SDE and CLD are orders of magnitude larger than those of VP-SDE. This scale difference is consistent with the substantially larger noise standard deviations typically induced by the VE formulation, which increases the expected denoising error and thus inflates the loss.

Beyond scale, the learning-curve shapes reveal qualitatively different training behavior. For VP-SDE, the training loss decreases rapidly early in training and then gradually plateaus, while the validation loss initially improves but subsequently increases, indicative of overfitting. In contrast, VE-SDE and CLD do not exhibit a comparable late-stage deterioration in validation loss over the same training horizon, suggesting that these settings did not reach the same degree of saturation on the validation set.

Overall, our experiments indicate that the VP-SDE configuration is easier to optimize under the current training setup, while VE-SDE and CLD may require additional hyperparameter tuning (e.g., learning rate, weighting, or noise schedule) to reach their best performance. An alternative interpretation is that the VE/CLD formulations present a more challenging learning problem for the model in this diffusion-policy setting, motivating further investigation into when and why VP-style parametrizations yield more favorable optimization behavior.

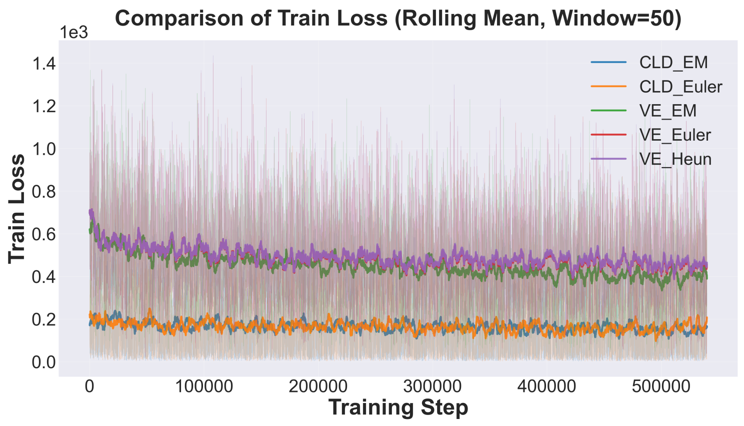

Figure 2: Smoothed Training Loss for VE-SDE and CLD

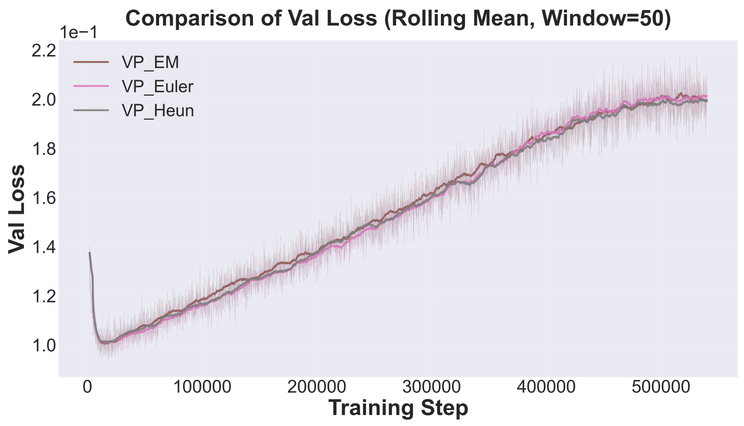

Figure 5: Smoothed Training Loss for VP-SDE

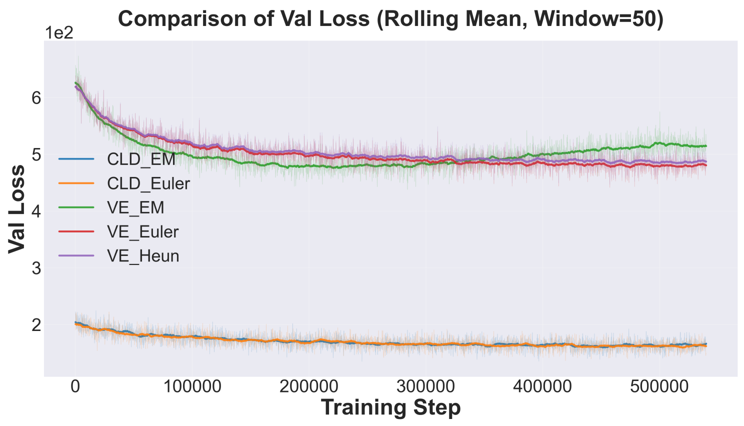

Figure 4: Smoothed Training Loss for VE-SDE and CLD

4.1 Quantitative Performance

Table 1 summarizes the performance across all configurations. The Variance Preserving (VP) SDE significantly outperforms the other formulations, achieving high success rates regardless of the solver used.

Table 1: Average Goal Coverage (Success Rate) on PushT ($N = 50$ episodes). VP-SDE demonstrates superior robustness, while VE-SDE shows a strong dependence on stochastic sampling.

	Deterministic (ODE)		**Stochastic (SDE)**
SDE Formulation	**Euler**	**Heun**	**Euler-Maruyama**
Variance Preserving (VP)	0.780	0.786	**0.802**
Variance Exploding (VE)	0.108	0.105	0.524
Critically Damped (CLD)	0.405	–	0.311

Robustness of VP-SDE: The VP formulation is remarkably robust to the choice of solver, with results clustering tightly around 0.79. The mean-reverting drift of the Ornstein-Uhlenbeck process ensures that the state remains bounded, creating a well-conditioned vector field. Consequently, higher-order corrections (Heun) or stochastic noise injection (EM) provide negligible benefits over a simple Euler step.

Stochastic Correction in VE-SDE: A striking result is the massive performance gap in VE-SDE between deterministic solvers (≈ 0.10) and the stochastic Euler-Maruyama solver (0.52). The deterministic ODE integrators fail almost completely. We attribute this to the unbounded nature of the VE variance. As $t \to 0$, the score norm grows large; small approximation errors in the neural network can cause a deterministic trajectory to diverge permanently from the data manifold.

However, the Euler-Maruyama solver injects Gaussian noise at every step. This noise acts as a **stochastic corrector**, kicking the state out of spurious local minima and back toward high-probability regions. For variance-exploding processes, stochasticity appears essential for compensating for score estimation errors.

The deterministic (ODE) samplers often generated very slow motions, with predicted actions remaining close to the agent's current state. One plausible explanation is the distributional mismatch in the initialization of the reverse process: during training, the initial action samples are centered around the next action, whereas at test time they are initialized around the current state. While this shift should be small, it may still bias early denoising steps toward conservative action updates.

Instability in CLD: For the Critically Damped system, the deterministic Euler solver (0.405) outperforms the stochastic solver (0.311). Since the CLD formulation controls momentum, the resulting trajectories are naturally prone to oscillation ("jitter"). In this regime, the injection of additional noise via Euler-Maruyama exacerbates these oscillations, destabilizing the control. The deterministic solver, by ignoring the diffusion term, effectively smooths the trajectory, resulting in marginally better performance.

5 Conclusion

In this project, we re-examined Diffusion Policies for visuomotor control through the lens of controlled stochastic differential equations. We systematically derived the forward and reverse dynamics for three distinct processes: Variance Preserving (VP), Variance Exploding (VE), and Critically Damped Langevin Dynamics (CLD). By training these models on the PushT benchmark and evaluating them with various numerical solvers, we established a clear hierarchy of performance for this contact-rich task.

Our results highlight a critical insight from stochastic control: while all three formulations theoretically target the same data distribution in the limit of perfect score estimation, the properties of the underlying SDEs dictate the difficulty of the learning problem and the conditioning of the sampling process. The VP-SDE, with its mean-reverting Ornstein-Uhlenbeck structure, ensures bounded state variance and stable gradients, leading to superior and solver-robust performance. In contrast, the unbounded variance of VE-SDE requires stochastic correction to function, while the second-order dynamics of CLD introduce oscillations that degrade precision. For practical robotic control, the stability provided by the VP formulation appears to be the dominant factor for success.

References

Brian DO Anderson. Reverse-time diffusion equation models. *Stochastic Processes and their Applications*, 12(3):313–326, 1982.

Julius Berner, Philipp Grohs, Gitta Kutyniok, and Philipp Petersen. An optimal control perspective on diffusion-based generative modeling. *arXiv preprint arXiv:2211.01364*, 2022.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A vision-language-action flow model for general robot control, 2024. URL https://arxiv.org/abs/2410.24164.

Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. In *Advances in Neural Information Processing Systems*, 2018.

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 44(10-11):1684–1704, 2025.

Tim Dockhorn, Arash Vahdat, and Karsten Kreis. Score-based generative modeling with critically-damped langevin diffusion, 2022. URL https://arxiv.org/abs/2112.07068.

Aapo Hyvärinen. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6:695–709, 2005.

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. In *Advances in Neural Information Processing Systems*, 2022.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. Openvla: An open-source vision-language-action model, 2024. URL https://arxiv.org/abs/2406.09246.

Alex Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models, 2021. URL https://arxiv.org/abs/2102.09672.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2021a.

Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations, 2021b. URL https://arxiv.org/abs/2011.13456.

Belinda Tzen and Maxim Raginsky. Theoretical guarantees for sampling and inference in generative models with latent diffusions. In *Conference on Learning Theory*, 2019.

Pascal Vincent. A connection between score matching and denoising autoencoders. *Neural computation*, 23(7):1661–1674, 2011.

Tony Z Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost arms. In *Robotics: Science and Systems*, 2023.

Appendix

A Mathematical Framework

A.1 The Forward and Reverse Processes

Definition 4 (Forward Process SDE)**.** *A forward process is an Itô diffusion $\{\mathbf{a}_t\}_{t\in[0,T]}$ that transforms a data sample $\mathbf{a}_0 \sim \pi(\cdot \mid \mathbf{o})$ into a sample from a simple prior distribution p_T. It is defined by the SDE:*

$$d\mathbf{a}_t = f(\mathbf{a}_t, t)\, dt + g(t)\, d\mathbf{W}_t, \quad \mathbf{a}_0 \sim \pi(\cdot \mid \mathbf{o}), \tag{12}$$

where $\mathbf{W}_t$ is a standard D-dimensional Wiener process adapted to $\{\mathcal{F}_t\}$, $f : \mathbb{R}^D \times [0,T] \to \mathbb{R}^D$ is the drift vector, and $g : [0,T] \to \mathbb{R}$ is the scalar diffusion coefficient. Let $p_t(\mathbf{a})$ denote the marginal probability density of $\mathbf{a}_t$.

The key to reversing this process lies in the score function, $\nabla_{\mathbf{a}} \log p_t(\mathbf{a})$, which points in the direction of maximal increase of the log-probability density at time t. A remarkable result from stochastic calculus Anderson [1982] provides a way to express the dynamics of the time-reversed process using this score.

Theorem 1 (Reverse-Time SDE)**.** *Let $\{\mathbf{a}_t\}_{t\in[0,T]}$ satisfy the forward SDE (1). Under mild regularity conditions, the reverse-time process is also a diffusion process. It can be described in two equivalent ways:*

1. (Time-flipped variable) Let $\tau = T - t$ and define a new process $\tilde{\mathbf{a}}_\tau = \mathbf{a}_{T-\tau}$. This process satisfies the SDE:

$$d\tilde{\mathbf{a}}_\tau = \left[-f(\tilde{\mathbf{a}}_\tau, T-\tau) + g(T-\tau)^2 \nabla_{\tilde{\mathbf{a}}_\tau} \log p_{T-\tau}(\tilde{\mathbf{a}}_\tau)\right] d\tau + g(T-\tau)\, d\tilde{\mathbf{W}}_\tau, \tag{13}$$

where $\tilde{\mathbf{W}}_\tau$ is a standard Wiener process running forward in $\tau \in [0,T]$.

2. (Backward-in-time) The original process, viewed backward in time $t \in [T,0]$, satisfies:

$$d\mathbf{a}_t = \left[f(\mathbf{a}_t, t) - g(t)^2 \nabla_{\mathbf{a}_t} \log p_t(\mathbf{a}_t)\right] dt + g(t)\, d\bar{\mathbf{W}}_t, \tag{14}$$

where $d\bar{\mathbf{W}}_t$ is a reverse-time standard Wiener process increment.

Justification and Neural Network Approximation. The intuition behind this theorem comes from the Fokker-Planck equation, which describes the evolution of the density p_t. The term $-g(t)^2 \nabla_{\mathbf{a}_t} \log p_t(\mathbf{a}_t)$ is precisely the drift correction required to reverse the entropy production of the forward process. However, the true score, $\nabla_{\mathbf{a}_t} \log p_t(\mathbf{a}_t \mid \mathbf{o})$, is intractable to compute because it requires knowing the marginal density $p_t(\mathbf{a}_t \mid \mathbf{o}) = \int p_t(\mathbf{a}_t|\mathbf{a}_0) p_{\text{data}}(\mathbf{a}_0|\mathbf{o}) d\mathbf{a}_0$, which involves an integral over the entire unknown data distribution.

A.2 The Probability Flow ODE

For every SDE, there exists a corresponding deterministic process described by an ordinary differential equation (ODE) whose trajectories evolve such that their marginal densities $\{p_t(\mathbf{a})\}$ match those of the SDE. (Chen et al. [2018])

Definition 5 (Probability Flow ODE)**.** *The deterministic process $\{\mathbf{a}_t\}_{t\in[0,T]}$ sharing the same marginals as the SDE in Eq. (1) is given by:*

$$\frac{d\mathbf{a}_t}{dt} = f(\mathbf{a}_t, t) - \frac{1}{2} g(t)^2 \nabla_{\mathbf{a}_t} \log p_t(\mathbf{a}_t). \tag{15}$$

For generation, we solve the reverse-time version of this ODE, substituting the learned score s_θ:

$$\frac{d\mathbf{a}_t}{dt} = f(\mathbf{a}_t, t) - \frac{1}{2} g(t)^2 s_\theta(\mathbf{a}_t, t, \mathbf{o}). \tag{16}$$

This ODE is integrated backward from $t = T$ to $t = 0$, starting from the same prior sample $\mathbf{a}_T \sim p_T$. This provides a deterministic method for sampling. (Song et al. [2021a])

Justification. The evolution of p_t under the SDE (1) is given by the Fokker-Planck equation: $\frac{\partial p_t}{\partial t} = -\nabla \cdot (f p_t) + \frac{1}{2} g(t)^2 \nabla^2 p_t$. The velocity field of the ODE (5) is chosen such that the continuity equation $\frac{\partial p_t}{\partial t} = -\nabla \cdot (\frac{d\mathbf{a}_t}{dt} p_t)$ yields the same evolution for p_t. The factor of $1/2$ arises from this correspondence, effectively averaging out the stochastic fluctuations. (Karras et al. [2022])

B Derivations for the Variance Preserving SDE

B.0.1 Discrete noising

In diffusion-based learning, this SDE is the continuous-time analogue of the discrete forward noising process used in DDPM. In discrete time, with a step index $k \in \{0, \ldots, K-1\}$ and per-step noise variances $\beta_k \in (0,1)$, the forward process is

$$\mathbf{a}_{k+1} = \sqrt{1-\beta_k}\, \mathbf{a}_k + \sqrt{\beta_k}\, \boldsymbol{\epsilon}_k, \qquad \boldsymbol{\epsilon}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{I}).$$

It is clear that if the schedule satisfies $\bar{\alpha}_K := \prod_{k=0}^{K-1}(1-\beta_k) \approx 0$ (e.g., $\beta_K \approx 1$), the final state $\mathbf{a}_K$ is approximately Gaussian, independent of the initial data.

We can rewrite the update as

$$\Delta \mathbf{a}_k := \mathbf{a}_{k+1} - \mathbf{a}_k = \big(\sqrt{1-\beta_k} - 1\big)\mathbf{a}_k + \sqrt{\beta_k}\, \boldsymbol{\epsilon}_k.$$

To obtain a continuous-time limit, we introduce a time step Δt and a rate function $\beta(t)$ such that $\beta_k \approx \beta(t_k)\, \Delta t$ with $t_k = k\, \Delta t$. For small β_k we use the Taylor expansion

$$\sqrt{1-\beta_k} = 1 - \frac{1}{2}\beta_k + o(\beta_k),$$

so that

$$\Delta \mathbf{a}_k \approx -\frac{1}{2}\beta_k \mathbf{a}_k + \sqrt{\beta_k}\, \boldsymbol{\epsilon}_k.$$

Writing $\boldsymbol{\epsilon}_k = \Delta \mathbf{W}_k / \sqrt{\Delta t}$ with Wiener increments $\Delta \mathbf{W}_k \sim \mathcal{N}(\mathbf{0}, \Delta t\, \mathbf{I})$, we obtain

$$\Delta \mathbf{a}_k \approx -\frac{1}{2}\beta(t_k)\, \mathbf{a}_k\, \Delta t + \sqrt{\beta(t_k)}\, \Delta \mathbf{W}_k.$$

Dividing by Δt and taking the limit $\Delta t \to 0$ yields the VP SDE and justifies the choice of $f(\mathbf{a}_t, t)$ and $g(t)$.

$$d\mathbf{a}_t = -\frac{1}{2}\beta(t)\, \mathbf{a}_t\, dt + \sqrt{\beta(t)}\, d\mathbf{W}_t.$$

B.0.2 Derivation of the Forward Transition Kernel

This is a linear SDE. To find its solution and thus the transition kernel $p_t(\mathbf{a}_t \mid \mathbf{a}_0)$, we use an integrating factor. Let $\alpha(t) = \frac{1}{2}\int_0^t \beta(s) ds$. Define a new process $Y_t = e^{\alpha(t)}\mathbf{a}_t$. Applying Itô's lemma for a product:

$$\begin{aligned} dY_t &= \left(\frac{d}{dt} e^{\alpha(t)}\right) \mathbf{a}_t\, dt + e^{\alpha(t)}\, d\mathbf{a}_t \\ &= \frac{1}{2}\beta(t) e^{\alpha(t)} \mathbf{a}_t\, dt + e^{\alpha(t)} \left(-\frac{1}{2}\beta(t)\mathbf{a}_t\, dt + \sqrt{\beta(t)}\, d\mathbf{W}_t\right) \\ &= \left(\frac{1}{2}\beta(t) e^{\alpha(t)}\mathbf{a}_t - \frac{1}{2}\beta(t) e^{\alpha(t)}\mathbf{a}_t\right) dt + e^{\alpha(t)}\sqrt{\beta(t)}\, d\mathbf{W}_t = e^{\alpha(t)}\sqrt{\beta(t)}\, d\mathbf{W}_t. \end{aligned}$$

Integrating from 0 to t gives $Y_t - Y_0 = \int_0^t e^{\alpha(s)}\sqrt{\beta(s)}\, d\mathbf{W}_s$. Since $Y_0 = e^{\alpha(0)}\mathbf{a}_0 = \mathbf{a}_0$, we solve for $\mathbf{a}_t = e^{-\alpha(t)} Y_t$:

$$\mathbf{a}_t = e^{-\alpha(t)}\mathbf{a}_0 + e^{-\alpha(t)} \int_0^t e^{\alpha(s)} \sqrt{\beta(s)}\, d\mathbf{W}_s = e^{-\alpha(t)}\mathbf{a}_0 + \int_0^t e^{-(\alpha(t)-\alpha(s))}\sqrt{\beta(s)}\, d\mathbf{W}_s.$$

This shows $\mathbf{a}_t$ conditioned on $\mathbf{a}_0$ is a Gaussian random variable.

- **Mean:** $\mathbb{E}[\mathbf{a}_t \mid \mathbf{a}_0] = e^{-\alpha(t)}\mathbf{a}_0 = e^{-\frac{1}{2}\int_0^t \beta(s)ds}\mathbf{a}_0$.
- **Variance:** Using Itô isometry, the variance is $\left(\int_0^t e^{-2(\alpha(t)-\alpha(s))}\beta(s)\,ds\right)\mathbf{I}$. Let $v(s) = -2(\alpha(t)-\alpha(s)) = -\int_s^t \beta(u)du$. Then $dv = \beta(s)ds$. The integral becomes $\int_{s=0}^{s=t} e^v dv = [e^v]_{s=0}^{s=t} = [e^{-\int_s^t \beta(u)du}]_{s=0}^{s=t} = e^0 - e^{-\int_0^t \beta(u)du} = 1 - e^{-\int_0^t \beta(s)ds}$.

Thus, letting $\bar{\alpha}_t = \exp\left(-\int_0^t \beta(s)ds\right)$, the transition kernel is exactly: $p_t(\mathbf{a}_t \mid \mathbf{a}_0) = \mathcal{N}(\mathbf{a}_t; \sqrt{\bar{\alpha}_t}\mathbf{a}_0, (1-\bar{\alpha}_t)\mathbf{I})$.

- **Generative Models:**
  - **Reverse SDE:** Integrated from $t = T \to 0$:
$$d\mathbf{a}_t = \left[-\frac{1}{2}\beta(t)\mathbf{a}_t - \beta(t)s_\theta(\mathbf{a}_t, t, \mathbf{o})\right]dt + \sqrt{\beta(t)}\,d\bar{\mathbf{W}}_t.$$
  - **Probability Flow ODE:** Integrated from $t = T \to 0$:
$$\frac{d\mathbf{a}_t}{dt} = -\frac{1}{2}\beta(t)\left[\mathbf{a}_t + s_\theta(\mathbf{a}_t, t, \mathbf{o})\right].$$

B.0.3 Derivation of the VP Training Objective

The goal is to train a neural network s_θ to match the true score, $\nabla_{\mathbf{a}_t}\log p_t(\mathbf{a}_t \mid \mathbf{o})$. Minimizing the score-matching loss $\mathcal{L}(\theta) = \mathbb{E}_{t,\mathbf{a}_t}[\|s_\theta(\mathbf{a}_t, t) - \nabla_{\mathbf{a}_t}\log p_t(\mathbf{a}_t)\|^2]$ is equivalent to minimizing the KL divergence between the generative and data distributions. Since the true score is intractable, we instead match the score of the tractable conditional distribution $p_t(\mathbf{a}_t|\mathbf{a}_0)$. The conditional score is $\nabla_{\mathbf{a}_t}\log p_t(\mathbf{a}_t \mid \mathbf{a}_0) = -(\mathbf{a}_t - \sqrt{\bar{\alpha}_t}\mathbf{a}_0)/(1-\bar{\alpha}_t)$.

A sample $\mathbf{a}_t$ can be written via the reparameterization trick: $\mathbf{a}_t = \sqrt{\bar{\alpha}_t}\mathbf{a}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}$ for $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I})$. Rearranging gives $\boldsymbol{\epsilon} = (\mathbf{a}_t - \sqrt{\bar{\alpha}_t}\mathbf{a}_0)/\sqrt{1-\bar{\alpha}_t}$. Substituting this into the score expression reveals a key relationship:
$$\nabla_{\mathbf{a}_t}\log p_t(\mathbf{a}_t \mid \mathbf{a}_0) = -\frac{\sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}}{1-\bar{\alpha}_t} = -\frac{\boldsymbol{\epsilon}}{\sqrt{1-\bar{\alpha}_t}}.$$
This shows that knowing the conditional score is equivalent to knowing the noise $\boldsymbol{\epsilon}$ that was added to $\mathbf{a}_0$ to create $\mathbf{a}_t$. Instead of training s_θ to predict the score, it is more stable and convenient to reparameterize the network to predict the noise itself. We define a network $\boldsymbol{\epsilon}_\theta(\mathbf{a}_t, t, \mathbf{o})$ such that our score estimate is $s_\theta(\mathbf{a}_t, t, \mathbf{o}) = -\frac{\boldsymbol{\epsilon}_\theta(\mathbf{a}_t, t, \mathbf{o})}{\sqrt{1-\bar{\alpha}_t}}$. Substituting this into a weighted score matching loss yields:
$$\begin{aligned}\mathcal{L}(\theta) &= \mathbb{E}_{t,\mathbf{a}_0,\boldsymbol{\epsilon}}\left[\left\|-\frac{\boldsymbol{\epsilon}_\theta(\mathbf{a}_t, t, \mathbf{o})}{\sqrt{1-\bar{\alpha}_t}} - \left(-\frac{\boldsymbol{\epsilon}}{\sqrt{1-\bar{\alpha}_t}}\right)\right\|^2\right] \\ &= \mathbb{E}_{t,\mathbf{a}_0,\boldsymbol{\epsilon}}\left[\frac{1}{1-\bar{\alpha}_t}\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{a}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t, \mathbf{o})\|^2\right].\end{aligned}$$
The term $\frac{1}{1-\bar{\alpha}_t}$ acts as a time-dependent weight. For simplicity and training stability, many implementations (including the original DDPM) drop this weight, leading to the widely-used denoising objective:
$$\mathcal{L}_{VP}(\theta) = \mathbb{E}_{t,\mathbf{a}_0,\boldsymbol{\epsilon}}\left[\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{a}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t, \mathbf{o})\|^2\right]. \tag{17}$$

C Derivations for the Variance Exploding SDE

C.0.1 Derivation of the Forward Transition Kernel

Integrating the SDE from 0 to t gives $\mathbf{a}_t - \mathbf{a}_0 = \int_0^t \sqrt{\frac{d[\sigma^2(s)]}{ds}}\,d\mathbf{W}_s$. The conditional distribution $p_t(\mathbf{a}_t \mid \mathbf{a}_0)$ is Gaussian.

- **Mean:** $\mathbb{E}[\mathbf{a}_t \mid \mathbf{a}_0] = \mathbf{a}_0$, since the stochastic integral has zero mean.

- **Variance:** By Itô isometry, assuming $\sigma(0) = 0$:

$$\mathrm{Var}(\mathbf{a}_t \mid \mathbf{a}_0) = \left(\int_0^t \left(\sqrt{\frac{d[\sigma^2(s)]}{ds}} \right)^2 ds \right) \mathbf{I} = \left(\int_0^t \frac{d[\sigma^2(s)]}{ds} ds \right) \mathbf{I} = \sigma^2(t)\mathbf{I}.$$

Thus, the transition kernel is $p_t(\mathbf{a}_t \mid \mathbf{a}_0) = \mathcal{N}(\mathbf{a}_t; \mathbf{a}_0, \sigma^2(t)\mathbf{I})$.

- **Generative Models:**
  - **Reverse SDE:** Integrated from $t = T \to 0$:

$$d\mathbf{a}_t = -g(t)^2 s_\theta(\mathbf{a}_t, t, \mathbf{o})\, dt + g(t)\, d\bar{\mathbf{W}}_t.$$

  - **Probability Flow ODE:** Integrated from $t = T \to 0$:

$$\frac{d\mathbf{a}_t}{dt} = -\frac{1}{2} g(t)^2 s_\theta(\mathbf{a}_t, t, \mathbf{o}).$$

C.0.2 Derivation of the VE Training Objective

As with the VP SDE, we train a neural network s_θ to approximate the score of the conditional distribution, $\nabla_{\mathbf{a}_t} \log p_t(\mathbf{a}_t|\mathbf{a}_0)$. The true conditional score is:

$$\nabla_{\mathbf{a}_t} \log p_t(\mathbf{a}_t \mid \mathbf{a}_0) = \nabla_{\mathbf{a}_t} \left(-\frac{\|\mathbf{a}_t - \mathbf{a}_0\|^2}{2\sigma^2(t)} \right) = -\frac{\mathbf{a}_t - \mathbf{a}_0}{\sigma^2(t)}.$$

In this case, it is common to train the network s_θ to directly predict this score. The training objective is a weighted score-matching loss, $\mathcal{L}(\theta) = \mathbb{E}[\lambda(t) \| s_\theta - \nabla_{\mathbf{a}_t} \log p_t(\mathbf{a}_t \mid \mathbf{a}_0)\|^2]$. The weighting function $\lambda(t)$ is crucial for balancing the learning objective across different noise levels t. A theoretically motivated choice is $\lambda(t) = g(t)^2$, but a simpler and often more stable choice is $\lambda(t) = \sigma^2(t)$. This leads to the objective:

$$\mathcal{L}_{VE}(\theta) = \mathbb{E}_{t, \mathbf{a}_0, \mathbf{a}_t} \left[\sigma^2(t) \, \| s_\theta(\mathbf{a}_t, t, \mathbf{o}) - \nabla_{\mathbf{a}_t} \log p_t(\mathbf{a}_t \mid \mathbf{a}_0) \|^2 \right]. \tag{18}$$

During training, we sample a data point $\mathbf{a}_0$, a time t, and noise $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I})$. We form $\mathbf{a}_t = \mathbf{a}_0 + \sigma(t)\boldsymbol{\epsilon}$ and compute the target score $-(\mathbf{a}_t - \mathbf{a}_0)/\sigma^2(t) = -\boldsymbol{\epsilon}/\sigma(t)$. The loss is then the weighted squared error between the network's output $s_\theta(\mathbf{a}_t, t, \mathbf{o})$ and this target.

D Derivations for the Critically Damped Langevin Dynamics

D.0.1 Derivation of the Forward Process Solution

Let the state be $\mathbf{X}_t = [\mathbf{a}_t^\top, \mathbf{p}_t^\top]^\top$. The forward system (Eqs. 8-9) can be written in matrix form as $d\mathbf{X}_t = \mathbf{A}\mathbf{X}_t\, dt + \mathbf{B}\, d\mathbf{W}_t$, where (for each dimension):

$$\mathbf{A} = \begin{pmatrix} 0 & 1 \\ 0 & -\gamma \end{pmatrix}, \quad \mathbf{B} = \begin{pmatrix} 0 \\ \sqrt{2\gamma} \end{pmatrix}.$$

The solution to this linear SDE is $\mathbf{X}_t = e^{\mathbf{A}t}\mathbf{X}_0 + \int_0^t e^{\mathbf{A}(t-s)}\mathbf{B}\, d\mathbf{W}_s$. The matrix exponential is $e^{\mathbf{A}t} = \begin{pmatrix} 1 & \frac{1}{\gamma}(1 - e^{-\gamma t}) \\ 0 & e^{-\gamma t} \end{pmatrix}$. The state $\mathbf{X}_t$ conditioned on $\mathbf{X}_0 = [\mathbf{a}_0^\top, \mathbf{p}_0^\top]^\top$ follows a multivariate Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t)$.

- **Mean:** $\boldsymbol{\mu}_t = e^{\mathbf{A}t}\mathbf{X}_0 = \begin{pmatrix} \mathbf{a}_0 + \frac{1}{\gamma}(1 - e^{-\gamma t})\mathbf{p}_0 \\ e^{-\gamma t}\mathbf{p}_0 \end{pmatrix}$.
- **Covariance:** $\boldsymbol{\Sigma}_t = \int_0^t e^{\mathbf{A}(t-s)}\mathbf{B}\mathbf{B}^\top e^{\mathbf{A}^\top(t-s)}\, ds$.

D.0.2 Explicit Derivation of the CLD Covariance Matrix

Let $u = t - s$, so $ds = -du$. The integral becomes $\mathbf{\Sigma}_t = \int_0^t e^{\mathbf{A}u}\mathbf{B}\mathbf{B}^\top e^{\mathbf{A}^\top u}\, du$. First, compute the inner matrix products:

$$\begin{aligned}
\mathbf{B}\mathbf{B}^\top &= \begin{pmatrix} 0 \\ \sqrt{2\gamma} \end{pmatrix} \begin{pmatrix} 0 & \sqrt{2\gamma} \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 0 & 2\gamma \end{pmatrix}. \\
e^{\mathbf{A}u}\mathbf{B}\mathbf{B}^\top &= \begin{pmatrix} 1 & \frac{1}{\gamma}(1-e^{-\gamma u}) \\ 0 & e^{-\gamma u} \end{pmatrix} \begin{pmatrix} 0 & 0 \\ 0 & 2\gamma \end{pmatrix} = \begin{pmatrix} 0 & 2(1-e^{-\gamma u}) \\ 0 & 2\gamma e^{-\gamma u} \end{pmatrix}. \\
\text{Integrand} = (e^{\mathbf{A}u}\mathbf{B}\mathbf{B}^\top)e^{\mathbf{A}^\top u} &= \begin{pmatrix} 0 & 2(1-e^{-\gamma u}) \\ 0 & 2\gamma e^{-\gamma u} \end{pmatrix} \begin{pmatrix} 1 & 0 \\ \frac{1}{\gamma}(1-e^{-\gamma u}) & e^{-\gamma u} \end{pmatrix} \\
&= \begin{pmatrix} \frac{2}{\gamma}(1-e^{-\gamma u})^2 & 2(1-e^{-\gamma u})e^{-\gamma u} \\ 2e^{-\gamma u}(1-e^{-\gamma u}) & 2\gamma e^{-2\gamma u} \end{pmatrix} \\
&= \begin{pmatrix} \frac{2}{\gamma}(1-2e^{-\gamma u}+e^{-2\gamma u}) & 2(e^{-\gamma u}-e^{-2\gamma u}) \\ 2(e^{-\gamma u}-e^{-2\gamma u}) & 2\gamma e^{-2\gamma u} \end{pmatrix}.
\end{aligned}$$

Now, we integrate each element of this matrix from $u = 0$ to $u = t$:

- $(\Sigma_t)_{11} = \int_0^t \frac{2}{\gamma}(1 - 2e^{-\gamma u} + e^{-2\gamma u})du = \frac{2}{\gamma}\left[u + \frac{2}{\gamma}e^{-\gamma u} - \frac{1}{2\gamma}e^{-2\gamma u}\right]_0^t = \frac{2t}{\gamma} - \frac{3-4e^{-\gamma t}+e^{-2\gamma t}}{\gamma^2}$.
- $(\Sigma_t)_{12} = (\Sigma_t)_{21} = \int_0^t 2(e^{-\gamma u} - e^{-2\gamma u})du = 2\left[-\frac{1}{\gamma}e^{-\gamma u} + \frac{1}{2\gamma}e^{-2\gamma u}\right]_0^t = \frac{(1-e^{-\gamma t})^2}{\gamma}$.
- $(\Sigma_t)_{22} = \int_0^t 2\gamma e^{-2\gamma u}du = 2\gamma\left[-\frac{1}{2\gamma}e^{-2\gamma u}\right]_0^t = -[e^{-2\gamma t} - 1] = 1 - e^{-2\gamma t}$.

This completes the derivation of the covariance matrix for $(\mathbf{a}_t, \mathbf{p}_t)$ given $(\mathbf{a}_0, \mathbf{p}_0)$.

D.0.3 Deriving the Marginal Transition Kernel and Training Objective

The score network s_θ is a function of position $\mathbf{a}_t$. Therefore, we need the marginal distribution $p_t(\mathbf{a}_t \mid \mathbf{a}_0)$. This requires marginalizing out the initial momentum $\mathbf{p}_0$, which is not part of the expert data. We assume a standard prior $\mathbf{p}_0 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. The distribution $p_t(\mathbf{a}_t \mid \mathbf{a}_0)$ is the result of a linear transformation of Gaussian variables, and is therefore Gaussian itself.

- **Mean:** $\mathbb{E}[\mathbf{a}_t \mid \mathbf{a}_0] = \mathbb{E}_{\mathbf{p}_0}[\mathbf{a}_0 + \frac{1-e^{-\gamma t}}{\gamma}\mathbf{p}_0] = \mathbf{a}_0$.
- **Variance:** Using the law of total variance:

$$\begin{aligned}
\mathrm{Var}(\mathbf{a}_t \mid \mathbf{a}_0) &= \mathbb{E}_{\mathbf{p}_0}[\mathrm{Var}(\mathbf{a}_t \mid \mathbf{a}_0, \mathbf{p}_0)] + \mathrm{Var}_{\mathbf{p}_0}[\mathbb{E}[\mathbf{a}_t \mid \mathbf{a}_0, \mathbf{p}_0]] \\
&= (\Sigma_t)_{11}\mathbf{I} + \mathrm{Var}_{\mathbf{p}_0}\left[\mathbf{a}_0 + \frac{1-e^{-\gamma t}}{\gamma}\mathbf{p}_0\right] \\
&= (\Sigma_t)_{11}\mathbf{I} + \left(\frac{1-e^{-\gamma t}}{\gamma}\right)^2 \mathrm{Var}(\mathbf{p}_0) \\
&= \frac{2}{\gamma^2}\left(t\gamma + e^{-\gamma t} - 1\right)\mathbf{I}
\end{aligned}$$

Let us define $\sigma^2_{\text{CLD}}(t)$ as this total scalar variance term. The final marginal transition kernel is $p_t(\mathbf{a}_t \mid \mathbf{a}_0) = \mathcal{N}(\mathbf{a}_t; \mathbf{a}_0, \sigma^2_{\text{CLD}}(t)\mathbf{I})$. This has the same form as the VE SDE kernel. Therefore, the training objective derivation follows an identical logic.

$$\mathcal{L}_{CLD}(\theta) = \mathbb{E}_{t,\mathbf{a}_0,\mathbf{a}_t}\left[\sigma^2(t)\left\|s_\theta(\mathbf{a}_t, t, \mathbf{o}) - \nabla_{\mathbf{a}_t}\log p_t(\mathbf{a}_t \mid \mathbf{a}_0)\right\|^2\right]. \tag{19}$$