

# Diffusion Generative Flow Samplers: Improving Learning Signals Through Partial Trajectory Optimization

Dinghuai Zhang\*, Ricky T. Q. Chen, Cheng-Hao Liu, Aaron Courville &  
Yoshua Bengio

Thomas Mousseau

October 22, 2025

# Overview

---

## 1. Introduction

- 1.1 Problem Statement
- 1.2 Stochastic Optimal Control

## 2. Stochastic Optimal Control to GFlowNets

- 2.1 Stochastic Optimal Control
- 2.2 GFlowNet

## 3. Diffusion Generative Flow Sampler Results

- 3.1 Gradients Variance Reduction and  $Z$  estimation
- 3.2 Convergence Guarantees

## 4. Conclusion

- 4.1 Summary
- 4.2 Q&A

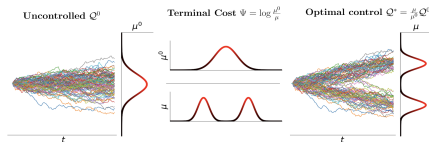
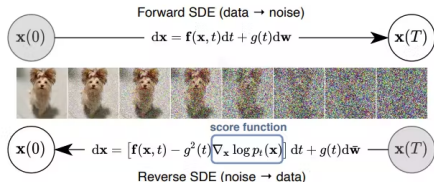
# Generative Modeling

## Task

Sample from a complex (high-dimensional and multimodal) distribution  $D$ .

$D$  can be given under the form of:

- A dataset of samples  $\{x_i\}_{i=1}^N \sim D$  (e.g., images, text, audio)
- An unnormalized density  $\mu(x)$  where  $D$  has density  $\pi(x) \propto \mu(x)$  (e.g., energy-based models, physics/chemistry)



# Sampling from Unnormalized Densities

**Context.** Sample from a  $D$ -dimensional target with unnormalized density  $\mu(x)$  where  $\mathbb{R}^D \rightarrow \mathbb{R}^+$ .

$$\pi(x) = \frac{\mu(x)}{Z}, \quad Z = \int_{\mathbb{R}^D} \mu(x) dx \text{ (unknown)}.$$

We assume we can evaluate  $\mu(x)$ , but we have no samples from  $\pi$  and do not know  $Z$ .

**Goal.** We seek a *sampler* (similar to MCMC/VI) that produces calibrated samples and, ideally, estimates  $Z$  *without* any dataset from  $\pi$ .

## Chemistry (molecule conformers).

Different 3D conformations have a formation energy from force-field terms (bonds, angles, dihedrals, nonbonded). A low energy means a higher probability of being sampled. A well-calibrated sampler is needed to draw conformers in proportion to these probabilities, which is important in binding-pose ranking, free-energy estimation and generating diverse realistic 3D conformers for screening.

# Diffusion Generative Flow Samplers

**Idea.** We will *reframe sampling* from an unnormalized target  $\pi(x) \propto \mu(x)$  as a *stochastic optimal control (SOC)* problem. This means learning a control function that steers a diffusion process so its *terminal marginal* matches  $\pi$ .

**Why this helps.**

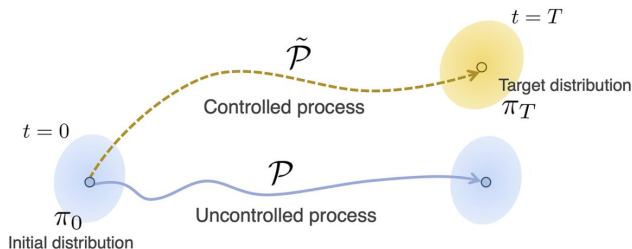
- Gives a *path-space* training objective/metric: a KL on trajectories  $\text{KL}(Q \parallel P)$  where  $P$  is the reference paths reweighted by  $\mu(x_T)$ .
- The *partition function*  $Z$  *cancels* inside this objective, so we can train using only  $\mu$  (and optionally  $\nabla \log \mu$ ).
- Lets us optimize *without samples from*  $\pi$  and still measure closeness to the true normalized endpoint.

**Caveat (sets up DGFS).** This path-KL places supervision *only at the terminal time*  $\Rightarrow$  poor *credit assignment* and high-variance gradients.

**DGFS fix (preview).** Inject *intermediate* learning signals via a GFlowNet-inspired *learned flow* and *subtrajectory balance*, enabling partial-trajectory training and more stable learning.

# Diffusion Generative Flow Samplers v2

Minimizing the running and terminal costs between the optimal controlled process and the uncontrolled reference process.



$$\begin{aligned} & \underset{\tilde{\mathcal{P}}}{\text{minimize}} && D_{\text{KL}}[\tilde{\mathcal{P}} \parallel \mathcal{P}] = \int_{\Omega} \log \frac{d\tilde{\mathcal{P}}}{d\mathcal{P}} d\tilde{\mathcal{P}} \\ & \text{s.t.} && x(0) \sim \pi_0, \quad x(T) \sim \pi_T \end{aligned}$$

2025-10-22

# Diffusion Generative Flow Samplers: Improving Learning Signals Through Partial Trajectory Optimization

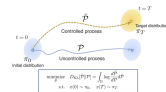
- └ Introduction

- └ Problem Statement

- └ Diffusion Generative Flow Samplers v2

## Diffusion Generative Flow Samplers v2

Minimizing the running and terminal costs between the optimal controlled process and the uncontrolled reference process.



refaire cette illustration avec le controlled, uncontrolled et target (optimal controlled) processus

# Controlled and Reference Processes

---

Controlled forward transition (learned drift).

$$P_F(x_{n+1} | x_n) = \mathcal{N}(x_{n+1}; x_n + h f_\theta(x_n, n), h\sigma^2 I)$$

Controlled process.

$$Q(x_{0:N}) = p_0^{\text{ref}}(x_0) \prod_{n=0}^{N-1} P_F(x_{n+1} | x_n)$$

Uncontrolled/Reference forward transition (zero drift).

$$P_F^{\text{ref}}(x_{n+1} | x_n) = \mathcal{N}(x_{n+1}; x_n, h\sigma^2 I)$$

Uncontrolled/Reference process and marginals.

$$Q^{\text{ref}}(x_{0:N}) = p_0^{\text{ref}}(x_0) \prod_{n=0}^{N-1} P_F^{\text{ref}}(x_{n+1} | x_n),$$

$$p_n^{\text{ref}}(x) = \mathcal{N}(x; \mu_0, \Sigma_0 + nh\sigma^2 I), \text{ is closed form.}$$

**Goal.** Learn  $f$  so that the terminal marginal  $Q(x_N)$  matches  $\pi(x) = \mu(x)/Z$  (no data,  $Z$  unknown.).



# Diffusion Generative Flow Samplers: Improving Learning Signals Through Partial Trajectory Optimization

- Introduction
- Stochastic Optimal Control
- Controlled and Reference Processes

## Controlled and Reference Processes

Controlled forward transition (learned drift):

$$P_T(x_{n+1} | x_n) = N(x_{n+1}; x_n + h f(x_n, n), h\sigma^2 I)$$

Controlled process:

$$Q(x_0, x_N) = \mu_0^{\text{ref}}(x_0) \prod_{n=0}^{N-1} P_T(x_{n+1} | x_n)$$

Uncontrolled/Reference forward transition (zero drift):

$$P_T^{\text{ref}}(x_{n+1} | x_n) = N(x_{n+1}; x_n, h\sigma^2 I)$$

Uncontrolled/Reference process and marginals:

$$Q^{\text{ref}}(x_0, x_N) = \mu_0^{\text{ref}}(x_0) \prod_{n=0}^{N-1} P_T^{\text{ref}}(x_{n+1} | x_n).$$

$$\mu_n^{\text{ref}}(x) = N(x; \mu_0, \Sigma_0 + nh\sigma^2 I), \text{ is closed form.}$$

Goal. Learn  $f$  so that the terminal marginal  $Q(x_N)$  matches  $\pi(x) = \mu(x)/Z$  (no data,  $Z$  unknown).

Why closed form for uncontrolled but not controlled? The uncontrolled reference process has no drift, so each transition is a simple Gaussian convolution, leading to Gaussian marginals with closed-form means and variances (e.g.,  $p_n^{\text{ref}}(x) = N(x; \mu_0, \text{cov}_0 + nh\sigma^2 I)$ ). The controlled process includes a learned drift  $f(x_n, n)$ , which makes transitions non-Gaussian and dependent on  $f$ , preventing a closed-form marginal expression without solving the integral numerically or via simulation.

je dois refaire cette slide, on ne voit meme comment  $x_n$  est sample (par rapport a la SDE) et aussi d'ou sort sigma mu et la closed form expliquer et c'est une convolution de gaussienne

# Path space KL objective

---

Target path measure via terminal reweighting.

$$P(x_{0:N}) \propto Q^{\text{ref}}(x_{0:N-1}|x_N)\mu(x_N) \propto Q^{\text{ref}}(x_{0:N}) \frac{\mu(x_N)}{p_N^{\text{ref}}(x_N)} \implies P(x_N) \propto \mu(x_N).$$

KL decomposition.

$$\text{KL}(Q\|P) = \mathbb{E}_Q \left[ \log \frac{Q}{P} \right] = \mathbb{E}_Q \left[ \log \frac{Q}{Q^{\text{ref}}} \right] + \mathbb{E}_Q \left[ \log \frac{p_N^{\text{ref}}(x_N)}{\mu(x_N)} \right] + \log Z.$$

Running control cost (Gaussian mean-shift) Girsanov theorem.

$$\mathbb{E}_Q \left[ \log \frac{Q}{Q^{\text{ref}}} \right] = \mathbb{E}_Q \sum_{n=0}^{N-1} \frac{h}{2\sigma^2} \|f_\theta(x_n, n)\|^2.$$

Terminal cost.

$$\mathbb{E}_Q \left[ \log \frac{p_N^{\text{ref}}(x_N)}{\mu(x_N)} \right] = \mathbb{E}_Q [\log p_N^{\text{ref}}(x_N) - \log \mu(x_N)]$$

2025-10-22

# Diffusion Generative Flow Samplers: Improving Learning Signals Through Partial Trajectory Optimization

- └ Introduction
- └ Stochastic Optimal Control
- └ Path space KL objective

Je devrais avoir une slide complete a expliquer d'ou vient le foward target process  $P$  parce que c'est le coeur de mon algorithme

## Path space KL objective

Target path measure via terminal reweighting:

$$P\{x_{0:N}\} \propto Q^{ref}(x_{0:N-1}|x_0) \pi(x_N) \propto Q^{ref}(x_{0:N}) \frac{\pi(x_N)}{P(x_N)} \implies P\{x_N\} \propto \pi\{x_N\}.$$

KL decomposition:

$$\text{KL}(Q\|P) = \mathbb{E}_Q \left[ \log \frac{Q}{P} \right] = \mathbb{E}_Q \left[ \log \frac{Q}{Q^{ref}} \right] + \mathbb{E}_Q \left[ \log \frac{Q^{ref}(x_N)}{P(x_N)} \right] + \log Z.$$

Running control cost (Gaussian mean-shift) Jensen's theorem:

$$\mathbb{E}_Q \left[ \log \frac{Q}{Q^{ref}} \right] = \mathbb{E}_Q \sum_{t=0}^{N-1} \frac{b_t}{2\pi\sigma_t^2} \|\tilde{x}_t(x_{0:t})\|^2.$$

Terminal cost:

$$\mathbb{E}_Q \left[ \log \frac{Q^{ref}(x_N)}{P(x_N)} \right] = \mathbb{E}_Q \left[ \log \pi(x_N) - \log \pi(x_N) \right]$$

# Target Path Measure via Importance Sampling

**Importance Sampling Basics.** Importance sampling is a technique to estimate expectations under a hard-to-sample target distribution  $P$  using samples from an easy-to-sample proposal distribution  $Q$ . The key formula is:

$$\mathbb{E}_{x \sim Q} [f(x) \cdot w(x)] = \mathbb{E}_{x \sim P} [f(x)],$$

where the *importance weight*  $w(x) = \frac{P(x)}{Q(x)}$  corrects the samples from  $Q$  to behave as if they were from  $P$ .

**Intuition.**  $Q$  provides "biased" samples;  $w(x)$  upweights samples that are likely under  $P$  and downweights those that aren't, effectively resampling from  $P$  without directly sampling it.

**Application to Path Measures.** In our case, the "samples" are entire trajectories  $x_{0:N}$ , and we want the path measure  $P$  to have the correct terminal marginal  $\pi(x_N) \propto \mu(x_N)$ . We use the reference path measure  $Q^{\text{ref}}$  (easy to sample, e.g., Gaussian paths) as the proposal. The reweighted path measure is:

$$P(x_{0:N}) \propto Q^{\text{ref}}(x_{0:N}) \cdot w(x_{0:N}),$$

where the weight  $w(x_{0:N}) = \frac{\pi(x_N)}{p_N^{\text{ref}}(x_N)}$ . This ensures  $P(x_N) \propto \mu(x_N)$ , as the weight depends only on the endpoint.

# Credit Assignment Problem in SOC objective

## SOC Discrete-time objective

$$\min_{f_\theta} \mathbb{E}_Q \left[ \underbrace{\sum_{n=0}^{N-1} \frac{h}{2\sigma^2} \|f_\theta(x_n, n)\|^2}_{\text{Running cost}} + \underbrace{\log p_N^{\text{ref}}(x_N) - \log \mu(x_N)}_{\text{Terminal cost}} \right]$$

This objective is used on the seminar paper *Path Integral Sampler: Diffusion-based Sampling for Unnormalized Densities* by Dinghuai Zhang et al which presented the sampling from unnormalized densities as a stochastic optimal control problem.

**Explanation.** The SOC objective provides feedback signal only at the terminal step, making credit assignment difficult via backpropagation through time. This causes high-variance gradients, weak feedback for early actions, poor mode discovery, and inefficient optimization of the drift  $f$  without intermediate signals.

# SOC as a GFlowNet

## Comparison Table: GFlowNet vs. SOC Framework

Concept	GFlowNet	SOC
Forward Process	Trajectory sampling on DAG	Controlled diffusion path
Forward Transition Probability	$P_F(s' s)$	$P_F(x_{n+1} x_n) = \mathcal{N}(x_{n+1}; x_n + hf(x_n), h\sigma^2 I)$
Backward Transition Probability	$P_B(s s')$	$P_B^{\text{ref}}(x_n x_{n+1})$ (known)
Reward Function	$R(x)$ (unnormalized)	$\mu(x)$ (unnormalized density)
Terminal Marginal Distribution	$P_T(x) \propto R(x)$	$Q(x_N) \propto \mu(x_N)$
Flow State	Flow $F(s)$ at states	Learned flow $F_n(x)$

**Insight.** Since SOC can be viewed as a GFlowNet, we can apply GFlowNet tools (e.g., detailed balance loss, subtrajectory balance) to solve the credit assignment problem in diffusion sampling.

# Diffusion Generative Flow Samplers: Improving Learning Signals Through Partial Trajectory Optimization

- └ Stochastic Optimal Control to GFlowNets
  - └ Stochastic Optimal Control
  - └ SOC as a GFlowNet

make a mental note to comeback to professor hernandez comment on Tristan's Deleu presentation when someone asked: "The if GFlowNets are just Reinforcement Learning, then why keep doing research on this framework?" Professor Hernandez repoded by proving the felxibility of GFN, he said that it was en entery point between RL, Diffusion models, Energy-based models and in this case we prove that it can also be used in Stochastic Optimal Control problems.

## SOC as a GFlowNet

Comparison Table: GFlowNet vs. SOC Framework

Concept	GFlowNet	SOC
Forward Process	Trajectory sampling on DAG	Controlled diffusion path
Forward Transition Probability	$P_{\theta}(x' x)$	$P_{\theta}(x_{w+1} x_w) \propto \mathcal{N}(x_{w+1}   M(x_w), \sigma^2 I)$
Backward Transition Probability	$P_{\theta}(x x')$	$P_{\theta}^{\text{rev}}(x_w x_{w+1})$ (known)
Reward Function	$R(x)$ (unnormalized)	$\mu(x)$ (unnormalized density)
Terminal Marginal Distribution	$P_{\theta}(x) \propto R(x)$	$Q(x_0) \propto \mu(x_0)$
Flow State	Flow $F(x)$ at states	Learned flow $F_{\theta}(x)$

Insight: Since SOC can be viewed as a GFlowNet, we can apply GFlowNet tools (e.g., detailed balance loss, subtrajectory balance) to solve the credit assignment problem in diffusion sampling.

# Decomposition of the Target Process

**Target Process Decomposition.** The target path measure can be decomposed into a product of conditional distributions:

$$P(x_{0:N}) \propto Q^{\text{ref}}(x_{0:N}) \frac{\mu(x_N)}{p_N^{\text{ref}}(x_N)} \implies P(x_N) \propto \mu(x_N).$$

**Conditional Form.** Since the reweighting only affects the terminal state, the joint can be written as:

$$P(x_{0:N}) = \pi(x_N) \prod_{n=0}^{N-1} P_B^{\text{ref}}(x_n | x_{n+1}),$$

where  $P_B^{\text{ref}}(\cdot | \cdot)$  is the backward transition probability (derived from the target joint). This is tractable because  $P_B^{\text{ref}}$  is known because they are Gaussian convolutions with a known mean and variance.



2025-10-22

# Diffusion Generative Flow Samplers: Improving Learning Signals Through Partial Trajectory Optimization

└ Stochastic Optimal Control to GFlowNets

└ GFlowNet

└ Decomposition of the Target Process

conditional form is exactly like the RHS of the trajectory balance equation in gfn. Lire le paper de Nikolay sur Trajectory Balance pour bien comprendre cette equation et comment elle se relie a DGFS

## Decomposition of the Target Process

**Target Process Decomposition.** The target path measure can be decomposed into a product of conditional distributions:

$$P(x_{0:n}) \propto Q^n(x_{0:n}) \frac{P(x_n)}{P^n(x_n)} \implies P(x_0) \propto \mu(x_0),$$

**Conditional Form.** Since the resampling only affects the terminal state, the joint can be written as:

$$P(x_{0:n}) = \pi(x_n) \prod_{s=0}^{n-1} P^n(x_{s+1}|x_s),$$

where  $P^n(x_{s+1}|\cdot)$  is the backward transition probability (derived from the target joint). This is tractable because  $P^n$  is known because they are Gaussian convolutions with a known mean and variance.

# Rewriting the Target Process with Marginal

If We Had Access to  $p_n(x_n)$  we could write the partial joint which would allow training on subtrajectories and thus have better credit assignment:

$$P(x_{0:n}) = p_n(x_n) \prod_{k=0}^{n-1} P_B(x_k | x_{k+1}),$$

But There's No Closed Form for  $p_n(x_n)$ . To calculate  $p_n(x_n)$ , we would need to compute the integral:

$$p_n(x_n) = \int \pi(x_N) \prod_n^{N-1} P_B(x_n | x_{n+1}) dx_{n+1:N}.$$

**Why Is  $p_n(x_n)$  Hard to Compute?** The reference marginal at step  $N$  is a simple Gaussian formula. But for the target marginal at earlier steps  $n < N$ , there's no easy formula because the target process depends on the terminal state  $\pi(x_N)$ , making it complex. Computing it needs integrating over many future states, which can't be done analytically for general targets. At  $n = N$ , it's just  $\pi(x_N)$ , but earlier steps are harder.

**Solution: Learn an Approximation  $F_n \approx p_n$ .** We introduce a learned flow function  $F_n(x_n; \theta)$  parameterized by a neural network to approximate  $p_n(x_n)$ . This avoids computing the intractable integral at every training step.

# Diffusion Generative Flow Samplers: Improving Learning Signals Through Partial Trajectory Optimization

└ Stochastic Optimal Control to GFlowNets

└ GFlowNet

└ Rewriting the Target Process with Marginal

ne pas oublier de parler des MC quadrature method pour approximer l'integral mais c'est couteux en temps de calcul et pas scalable du tout

## Rewriting the Target Process with Marginal

If We Had Access to  $p_\theta(z)$  we could write the partial joint which would allow training on subtrajectories and thus have better credit assignment:

$$P(z_{1:n}) = p_\theta(z_1) \prod_{k=2}^n p_\theta(z_k | z_{1:k-1}).$$

But There's No Closed Form for  $p_\theta(z_n)$ . To calculate  $p_\theta(z_n)$ , we would need to compute the integral:

$$p_\theta(z_n) = \int p(z_n) \prod_{k=1}^{n-1} P_\theta(z_k | z_{k+1:n}) dz_{1:n-1}.$$

**Why Is  $p_\theta(z_n)$  Hard to Compute?** The inference marginal at step  $N$  is a simple Gaussian formula. But for the target marginal at earlier steps  $n < N$ , there's no easy formula because the target process depends on the terminal state  $x(N)$ , making it complex. Computing it needs integrating over many future states, which can't be done analytically for general targets. As  $n \rightarrow 0$ , it's just  $p(x_0)$ , but earlier steps are harder.

**Solution: Learn an Approximation  $F_n \approx p_\theta$ .** We introduce a learned flow function  $F_n(z_n, \theta)$  parameterized by a neural network to approximate  $p_\theta(z_n)$ . This avoids computing the intractable integral at every training step.

# Trajectory Balance with Learned Flow

**Proposed Amortized Approach.** Train  $F_n(\cdot; \phi)$  to satisfy the following constraint for all partial trajectories  $x_{n:N}$ :

$$F_n(x_n; \phi) \prod_{k=n}^{N-1} P_F(x_{k+1}|x_k; \theta) = \mu(x_N) \prod_{k=n}^{N-1} P_B(x_k|x_{k+1}).$$

## Details.

- $P_F$  (forward policy) and  $F_n$  are parameterized by deep neural networks, with parameters  $\theta$  and  $\phi$  respectively.
- We can view  $F_n$  as an approximation to the unknown marginal  $p_n(x_n)$  thus an amortized way to estimate the intractable integral.
- We use only  $\mu(\cdot)$  (no  $Z$ ), so the unknown normalization is absorbed into  $F_n$ .

# Detailed Balance with Learned Flow

---

**Deriving SubTB.** Comparing the constraint for  $n$  and  $n + 1$  gives a formula independent of  $\mu$  which now only involves local transitions (signal subtrajectories instead of full trajectories)

$$F_n(x_n; \theta) P_F(x_{n+1} | x_n; \theta) = F_{n+1}(x_{n+1}; \theta) P_B(x_n | x_{n+1}).$$

# Detailed Balance Loss for Subtrajectories

---

Detailed Balance Loss for Partial Trajectories.

$$\ell_{\text{DB}}(x_{m:n}; \theta, \phi) = \left( \log \frac{F_m(x_m; \phi) \prod_{k=m}^{n-1} P_F(x_{k+1}|x_k; \theta)}{F_n(x_n; \phi) \prod_{k=m}^{n-1} P_B(x_k|x_{k+1})} \right)^2$$

**Forward Process (Controlled).** The learned forward transition depends on  $\theta$ :

$$P_F(x_{n+1}|x_n; \theta) = \mathcal{N}(x_{n+1}; x_n + hf_{\theta}(x_n, n), h\sigma^2 I),$$

where  $f(x_n, n)$  is the learned drift parameterized by  $\theta$ .

# Diffusion Generative Flow Samplers: Improving Learning Signals Through Partial Trajectory Optimization

- └ Stochastic Optimal Control to GFlowNets
  - └ GFlowNet
  - └ Detailed Balance Loss for Subtrajectories

## Detailed Balance Loss for Subtrajectories

Detailed Balance Loss for Partial Trajectories.

$$\ell_{\text{DB}}(x_{0:n}; \theta, \phi) := \left( \log \frac{P_{\phi}(x_{0:n}; \phi) \prod_{t=1}^{n-1} P_{\theta}(x_{t+1}|x_t; \theta)}{P_{\phi}(x_0; \phi) \prod_{t=1}^{n-1} P_{\theta}(x_t|x_{t-1})} \right)^2$$

Forward Process (Controlled). The learned forward transition depends on  $\theta$ :

$$Pr\{x_{t+1}|x_t; \theta\} := \mathcal{N}(x_{t+1}; x_t + \text{bb}(x_t, \theta), \text{bc}^2 I),$$

where  $f(x_t, \theta)$  is the learned drift parameterized by  $\theta$ .

a voir si je voudrais pas mettre une schema sur l'une des 3 derniers slides pour faire un recap de scrap la SOC objective, now we build DB loss with learned flow to solve credit assignment problem

# Overall Training Objective

---

Overall Loss for a Full Trajectory.

$$L(\tau; \theta) = \frac{\sum_{0 \leq m < n \leq N} \lambda^{n-m} \ell_{\text{SubTB}}(x_{m:n})}{\sum_{0 \leq m < n \leq N} \lambda^{n-m}}, \quad \tau = (x_0, \dots, x_N)$$

Explanation of Parameters and Variables.

- $\tau$ : Full trajectory  $(x_0, \dots, x_N)$ .
- $\lambda$ : Positive scalar weighting different subtrajectory lengths (e.g., shorter subtrajectories get higher weight if  $\lambda < 1$ ).
- The numerator sums SubTB losses over all subtrajectories  $x_{m:n}$ , weighted by  $\lambda^{n-m}$  (length-based weighting).
- The denominator normalizes to average the losses.
- This combines signals from all subtrajectory lengths, reducing variance and improving credit assignment.

**Where is  $f$  (Learned Drift)?**  $f$  is embedded in  $P_F$  within each  $\ell_{\text{SubTB}}$ . The loss optimizes  $\theta$ , which includes parameters for  $f$  (via NN1 and NN2:  $f(\cdot, n)/\sigma = \text{NN1}(\cdot, n) + \text{NN2}(n) \cdot \nabla \log \mu(\cdot)$ ), steering the forward process toward the target.



## DGFS algorithm

---



---

### Algorithm 1 DGFS training algorithm

---

**Require:** DGFS model with parameters  $\theta$ , reward function  $\mu(\cdot)$ , variance coefficient  $\bar{\sigma}$ .

**repeat**

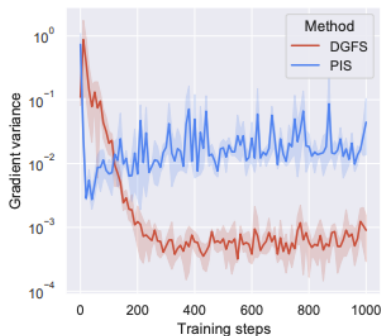
    Sample  $\tau$  with control  $\mathbf{f}(\cdot, \cdot; \theta)$  and  $\bar{\sigma}$ ;

$\Delta\theta \leftarrow \nabla_{\theta} \mathcal{L}(\tau; \theta)$  (as per Equation 15);

    Update  $\theta$  with Adam optimizer;

**until** some convergence condition

# Reduction of gradient variance



**Figure:** Gradient variance comparison between DGFS and PIS. DGFS shows significantly lower variance, leading to more stable training.

# Accurate partition function Z estimation

**Estimation of the partition function.** For an arbitrary trajectory  $\tau = \{x_n\}_{n=0}^N$ , one could define its (log) importance weight to be  $S(\tau; \theta) = \log P(x_{0:N}) - \log Q(x_{0:N}; \theta)$ , where  $Q(x_{0:N}; \theta) = p_0^{\text{ref}}(x_0) \prod_{n=0}^{N-1} P_F(x_{n+1}|x_n; \theta)$ .

**Log-partition function (log Z) estimator:**

$$\log \sum_{b=1}^B \exp(S(\tau^{(b)}; \theta)) - \log B \leq \log Z, \quad \tau^{(b)} \sim Q(\cdot; \theta).$$

**Explanation.**

- $S(\tau; \theta)$ : The log importance weight, measuring how much more likely the trajectory is under the target  $P$  (which has terminal marginal  $\pi \propto \mu$ ) than under the learned  $Q$ .
- $Q(x_{0:N}; \theta)$ : The learned path measure, with terminal marginal approximating  $\pi$ .
- The estimator uses importance sampling: Sample trajectories from  $Q$ , compute their weights, and average in log space to estimate  $\log Z$ .
- The inequality  $\leq \log Z$  holds because  $Q$  approximates  $P$ , providing a lower bound on  $\log Z$ .
- This gives an accurate estimate of  $Z$  without needing samples from  $\pi$  leveraging the learned model

## Partition function estimation results

	MoG	FUNNEL	MANYWELL	VAE	Cox
SMC	$0.289 \pm 0.112$	$0.307 \pm 0.076$	$22.36 \pm 7.536$	$14.34 \pm 2.604$	$99.61 \pm 8.382$
VI-NF	$1.354 \pm 0.473$	$0.272 \pm 0.048$	$2.677 \pm 0.016$	$6.961 \pm 2.501$	$83.49 \pm 2.434$
CRAFT	$0.348 \pm 0.210$	$0.454 \pm 0.485$	$0.987 \pm 0.599$	$0.521 \pm 0.239$	$13.79 \pm 2.351$
FAB w/ BUFFER <sup>5</sup>	$0.003 \pm 0.0005$	$0.0022 \pm 0.0005$	$0.032 \pm 0.004$	N/A	$0.19 \pm 0.04$
PIS	$0.036 \pm 0.007$	$0.305 \pm 0.013$	$1.391 \pm 1.014$	$2.049 \pm 2.826$	$11.28 \pm 1.365$
DDS	$0.028 \pm 0.013$	$0.416 \pm 0.094$	$1.154 \pm 0.626$	$1.740 \pm 1.158$	N/A <sup>6</sup>
DGFS	$0.019 \pm 0.008$	$0.274 \pm 0.014$	$0.904 \pm 0.067$	$0.180 \pm 0.083$	$8.974 \pm 1.169$

Figure: To write a caption

## Mode coverage results

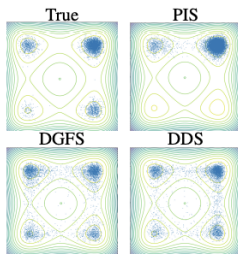


Figure: Manywell plots. DGFS and DDS but not PIS recover all modes

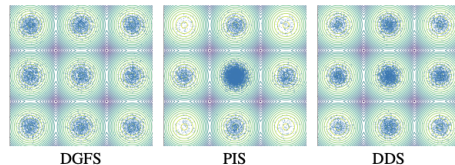


Figure: MoG visualization of DGFS and other diffusion-based samplers shows that DGFS could capture the diverse modes well. The contours display the landscape of the target density

# Convergence Theorem

---

## Convergence Guarantees for DGFS.

- Previous studies (Bortoli, 2022; Chen et al., 2022; Lee et al., 2022) show that the terminal sample distribution of a diffusion model converges to the target under mild assumptions if the control term is well learned. These apply to DGFS since proofs are independent of training method.
- Zhang et al. (2022a) prove that a perfectly learned score corresponds to zero GFlowNet training loss (Bengio et al., 2021; 2023), ensuring a well-trained DGFS accurately samples from the target distribution.

# Off-policy training capability

---

# Conclusion

---

**Summary of DGFS.** We propose the Diffusion Generative Flow Sampler (DGFS), a novel algorithm that trains diffusion models to sample from given unnormalized target densities. Different from prior works that could only learn from complete diffusion chains, DGFS can update its parameters with only partial specification of the stochastic process trajectory; moreover, DGFS can receive intermediate signals before completing the entire path. These features help DGFS benefit from efficient credit assignment and thus achieve better partition function estimation bias in the sampling benchmarks.



# Questions and hopefully answers :)

---

Questions?