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 21, 2025

Overview

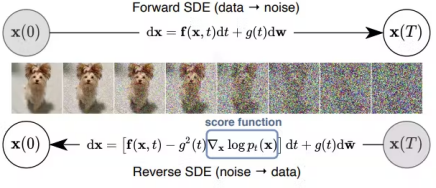
Generative Modeling

Task

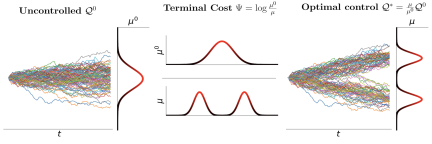
Sample from a complex (high-dimensional and multimodal) distribution D

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

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

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

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

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

Chemistry (small-molecule conformers). Different 3D conformations have a formation energy from force-field terms (bonds, angles, dihedrals, nonbonded); lower energy $\Rightarrow$ higher Boltzmann probability. A well-calibrated sampler is needed to draw conformers in proportion to these probabilities, which is important in binding-pose ranking/free-energy estimation, Boltzmann-weighted property prediction (e.g., NMR shifts), 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: learn a control that steers a simple reference diffusion so its *terminal marginal* matches π.

Why this helps.

- Gives a *path-space* training objective/metric: a KL on trajectories $\mathrm{KL}(Q \,\|\, P)$ 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 μ (and optionally $\nabla \log \mu$).
- Lets us optimize *without samples from* π 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.

Steps 1–2: Forward & Reference in discrete time

Controlled forward transition (learned drift).

$$P_F(x_{n+1} \mid x_n) \;=\; \mathcal{N}\big(x_{n+1};\; x_n + h\, f(x_n, n),\; h\sigma^2 I\big)$$

Controlled path law.

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

Uncontrolled (reference) transition (zero drift).

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

Reference path law and marginals.

$$Q^{\text{ref}}(x_{0:N}) \;=\; p_0^{\text{ref}}(x_0) \prod_{n=0}^{N-1} P_F^{\text{ref}}(x_{n+1} \mid x_n), \qquad p_n^{\text{ref}}(x) \text{ is closed form.}$$

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

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:

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

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^{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.

Step 3: Path target & KL $\Rightarrow$ SOC objective

Target path measure via terminal reweighting.

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

KL decomposition.

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

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

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

Terminal potential from the target Girsanov theorem.

$$\mathbb{E}_Q\left[\log\frac{p_N^{\text{ref}}(x_N)}{\pi(x_N)}\right] = \mathbb{E}_Q\big[\log p_N^{\text{ref}}(x_N) - \log\mu(x_N)\big] \;+\; \log Z.$$

SOC objective

SOC objective (discrete-time).

$$\min_f \; \mathbb{E}_Q\Big[\sum_{n=0}^{N-1} \tfrac{h}{2\sigma^2}\|f(x_n,n)\|^2 \; + \; \log p_N^{\text{ref}}(x_N) - \log \mu(x_N)\Big]$$

we will be using this discrete-time objective as our loss function to optimize the drift f. Thus we can model the drift with a neural network f_θ and optimize the parameters θ backpropagating through time (BPTT) while avoiding to use the stochastic-adjoint method necessary for a neural SDE.

Credit Assignment Problem in Path Space

Current Loss Function

$$\min_f \ \mathbb{E}_Q\Big[\sum_{n=0}^{N-1} \tfrac{h}{2\sigma^2}\|f(x_n,n)\|^2 \ + \ \log p_N^{\text{ref}}(x_N) - \log \mu(x_N)\Big]$$

Explanation. Since we are guiding our trajectory using the terminal marginal distribution p_N^{ref} and $\mu(x)$, the only signal we have is at the end. Thus, when we do the chain rule for backpropagation through time, it is very hard to know which decision at which time step got the right or wrong action (credit assignment problem).

GFlowNet's perspective on DGFS

Recap of GFlowNets. GFlowNets learn to sample from unnormalized distributions by modeling flows on a DAG.

- **DAG Structure:** Define a DAG $G = (S, A)$ with states S and directed edges $s \to s' \in A$.
- **Trajectories:** Complete trajectories $\tau = s_0 \to s_1 \to \cdots \to s_T$ from start to terminal states.
- **Forward Policy:** $P_F(s'|s)$, the probability of transitioning from s to s'.
- **Terminal Marginal:** $P_T(x) = \sum_{\tau \text{ ending at } x} P_F(\tau)$, the marginal over terminating states.
- **Goal:** Learn P_F such that $P_T(x) \propto R(x)$, where $R(x)$ is the unnormalized reward/density, and $Z = \sum_x R(x)$.

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	$P_F(s'\|s)$	$P_F(x_{n+1}\|x_n) = \mathcal{N}(x_{n+1}; x_n + hf(x_n), h\sigma^2 I)$
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.

Decomposition of the Target Process

Target Process Decomposition. The target path measure (Equation 4) 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 (Equation 10). 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^{ref} is known (e.g., a Gaussian bridge for diffusion processes).

Intuition. This factorization shows that the target process is like sampling backward from the terminal distribution $\pi(x_N)$, using the backward kernel to condition earlier states. It's analogous to reverse-time diffusion, where we start from the target and go backward.

Rewriting the Target Process with Marginal

Rewriting the Joint Using Marginal Densities. The backward factorization of the target process is:

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

To express it in terms of the marginal densities $p_n(x_n)$, note that the marginal at step n is obtained by integrating out the future states $x_{n+1:N}$ from the joint:

$$p_n(x_n) = \int P(x_{0:N})\, dx_{0:n-1}\, dx_{n+1:N} = \int \pi(x_N) \prod_{l=n}^{N-1} P_B^{\text{ref}}(x_l|x_{l+1})\, dx_{n+1:N} \propto \int \mu(x_N) \prod_{l=n}^{N-1} P_B^{\text{ref}}(x_l|x_{l+1})\, dx_{n+1:N}.$$

This shows that $p_n(x_n)$ is the "unnormalized" density at step n, propagated backward from the terminal $\pi(x_N)$ via the backward transitions.

What $p_n(x_n)$ Represents.

- $p_N(x_N) = \pi(x_N)$: The terminal marginal, which is the normalized target density we want to match (proportional to $\mu(x_N)$).
- For $n < N$, $p_n(x_n)$ is the marginal density at step n under the target process P. It represents how likely the state x_n is at time n, given that the trajectory will eventually reach a terminal state distributed as $\pi(x_N)$. In other words, it's the distribution of x_n marginalized over all possible future paths that lead to $\pi(x_N)$.
- Intuitively, $p_n(x_n)$ encodes the "value" or importance of being at x_n at step n, as it accounts for the probability of reaching high-μ terminals from there. This is why approximating $p_n(x_n)$ allows training with partial trajectories starting from step n.

Why This Helps in Writing the Target Process. The joint $P(x_{0:N})$ can be thought of as a chain where each $p_n(x_n)$ summarizes the "progress" toward the terminal, but the backward form directly ties it to $\pi(x_N)$. By learning $F_n \approx p_n$, we can reconstruct or approximate the joint without computing the full integral, enabling efficient partial-trajectory optimization.

Rewriting the Target Process with Marginal

If We Had Access to $p_n(x_n)$. We could write the partial joint as:

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

where P_B is the backward kernel. This would allow us to define partial trajectory targets, enabling training with shorter trajectories and better credit assignment.
But There's No Closed Form for $p_n(x_n)$. To calculate $p_n(x_n)$, we would need to compute the integral:

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

This is a high-dimensional integral with no analytical solution for general π.
Why Parameterize Both $F_n(\theta)$ and P_F? The constraint requires both: F_n approximates p_n, and P_F is the forward policy we learn. We can't simply calculate the forward process because it's unknown—we're learning it to steer trajectories toward the target. Parameterizing both allows the constraint to hold, enabling amortized learning without per-step integrals.
Naive Alternative: Monte Carlo Quadrature. Quadrature is a numerical integration method that approximates integrals by evaluating the integrand at a set of points (nodes) and weighting them. Monte Carlo quadrature uses random sampling for high dimensions.
Why Expensive. Quadrature requires many evaluations of the integrand (e.g., sampling trajectories), which is computationally intensive and scales poorly with dimensionality, making it impractical for per-training-step use as a replacement for p_n.

Amortized Training: The Subtrajectory Constraint

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

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

Details.

- P_F (forward policy) and F_n are parameterized by deep neural networks.
- Once this constraint holds, $F_n(x_n;\theta)$ equals the integral in Equation 11, amortizing the integration into the learning of θ.
- We use only $\mu(\cdot)$ (no Z), so the unknown normalization is absorbed into F_n.

Decision. This avoids per-step quadrature by enforcing a global constraint on partial trajectories, making training efficient and scalable.

Training Objective and Implementation

Training Objective. Regress the left-hand side (LHS) of the constraint to the right-hand side (RHS). For stability, perform this in log space:

$$\min_{\theta} |\log(\text{LHS}) - \log(\text{RHS})| .$$

Details.

- This is a regression loss that minimizes the difference between the learned flow products and the target products involving μ.
- Log space prevents numerical issues from large or small values.

Shared Parameters. The flow function at different steps shares the same parameters; achieved by adding a step embedding input to the network $F(\cdot, n; \theta)$.
Decision. Sharing parameters reduces model complexity and ensures consistency across time steps, while embeddings allow step-specific behavior.

Derived Subtrajectory Balance (SubTB)

Deriving SubTB. Comparing the constraint for n and $n+1$ gives a formula independent of μ:

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

Details.

- This is the Subtrajectory Balance (SubTB) constraint, a generalization of detailed balance for partial trajectories.
- It provides intermediate learning signals without needing μ directly, improving credit assignment.

Overall Decision. The method learns two neural networks: the flow $F(\cdot, n;\theta)$ and the forward policy P_F. This setup enables efficient, stable training with intermediate supervision.

DGFS algorithm

Algorithm 1 DGFS training algorithm

Require: DGFS model with parameters θ, reward function $\mu(\cdot)$, variance coefficient $\bar{\sigma}$.
  repeat
    Sample τ with control $\mathbf{f}(\cdot, \cdot; \theta)$ and $\bar{\sigma}$;
    $\triangle\theta \leftarrow \nabla_\theta \mathcal{L}(\tau; \theta)$ (as per Equation 15);
    Update θ with Adam optimzier;
  until some convergence condition

Conclusion