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 27, 2025

Overview

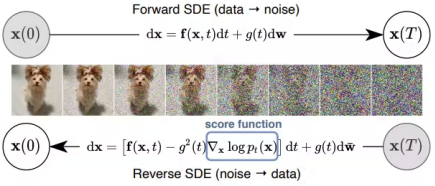
Generative Modeling

Task

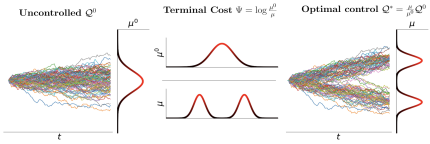
Sample from a complex (high-dimensional and multimodal) distribution D.

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

Context. 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.

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

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 π.

Why this helps.

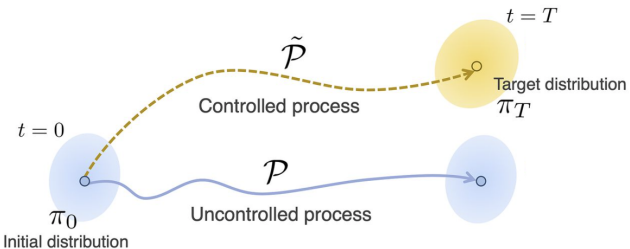
- Gives a *path-space* training objective/metric: a KL on trajectories $\mathrm{KL}(Q \,\|\, P)$ where P is the reference paths reweighted by $\mu(x_T)$.
- The *partition function* Z *cancels* inside this objective, so we can train using only μ (and optionally $\nabla \log \mu$).
- Lets us optimize *without 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.

Diffusion Generative Flow Samplers v2

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



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

Controlled and Reference Processes

Controlled forward transition (learned drift).

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

Controlled process.

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

Uncontrolled/Reference forward transition (zero drift).

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

Uncontrolled/Reference process and marginals.

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

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

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

Closed Form of Reference Marginal

Reference Process.

$$x_N = x_0 + \sqrt{h}\sigma \sum_{k=0}^{N-1} \varepsilon_k.$$

Distribution of Terminal State. Since the sum of independent Gaussians is Gaussian:

$$x_N \sim \mathcal{N}(m_0, \Sigma_0 + Nh\sigma^2 I).$$

Terminal Marginal Density.

$$p_N^{\text{ref}}(x) = \frac{1}{(2\pi)^{D/2}|\Sigma_0 + Nh\sigma^2 I|^{1/2}} \exp\left(-\frac{1}{2}(x - m_0)^\top (\Sigma_0 + Nh\sigma^2 I)^{-1}(x - m_0)\right).$$

Path space KL objective

Target path measure via terminal reweighting.

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

KL decomposition.

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

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.

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

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.

Credit Assignment Problem in SOC objective

SOC Discrete-time objective

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

This objective is used on the seminar paper *Path Integral Sampler: Diffusion-based Sampling for Unnormalized Densities* Zhang et al. 2022a 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.

Adressing Credit Assignment via Target Process

Target process necessitates terminal reweighting...

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

Could we write the target process differently to allow intermediate supervision?

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

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

where $Q_B^{\text{ref}}(\cdot|\cdot)$ is the backward transition probability (derived from the target joint). This is tractable because Q_B^{ref} is known.

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:

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

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

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

$$p_n(x_n) \propto \int_{\mathbb{R}^{(N-n)D}} \mu(x_N) \prod_{k=n}^{N-1} Q^{\text{ref}} B(x_k|x_{k+1}), dx_{n+1:N}.$$

Why is $p_n(x_n)$ hard to compute? The integral represents summing over all possible future paths from x_n to the terminal x_N, weighting each by the terminal density $\pi(x_N)$. This high-dimensional integration (over $(N-n) \times D$ dimensions) has no analytical solution for general $\mu(x_N)$. Numerically approximating it via quadratures or Monte Carlo is computationally expensive, especially since training requires computing it repeatedly for subtrajectories as intermediate signals (replacing the single terminal signal), making it impractical.

Trajectory Balance with Learned Flow

Proposed Amortized Approach. Train $F_n(\cdot;\phi)$, a neural network, 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 θ and ϕ 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.
- Very similar to a Value Function in RL, estimating future expected rewards (here, future density mass).

Detailed and SubTB Balance with Learned Flow

Detailed Balance from Trajectory Balance condition.

Comparing the constraint for n and $n+1$ gives a formula independent of μ which now only involves local transitions (signal subtrajectories instead of full trajectories)

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

Subtrajectory Balance Loss for Partial Trajectories

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

Using a mix of different subtrajectory lengths (m,n) allows better credit assignment thus more stable training of both the forward policy and the learned flow.

Overall Training Objective

Recall SubTB Loss

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

Diffusion Generative Flow Sampler (DGFS) Loss

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

This combines signals from all subtrajectory lengths, reducing variance and improving credit assignment

- τ: Full trajectory $(x_0,\ldots,x_N)$.
- λ: 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 λ^{n-m} (length-based weighting).
- The denominator normalizes to average the losses..

Algorithm 1 DGFS Training

Require: $\mu(\cdot)$, $\bar{\sigma}$, N, λ, B, η

Init $\theta = (\theta_f, \phi)$
repeat
  Sample trajectories:
  for $b = 1$ to B **do**
    $\tau^{(b)} = (x_0^{(b)}, \ldots, x_N^{(b)})$ under $x_{n+1} = x_n + h f_{\theta_f}(x_n, n) + \sqrt{h}\bar{\sigma}\varepsilon_n$, $\varepsilon_n \sim \mathcal{N}(0, I)$
  end for
  Build subtrajectories: $\mathcal{S}(\tau^{(b)})$ of (m, n) with $0 \le m < n \le N$
  Compute SubTB loss:

$$\mathcal{L}(\tau^{(b)}; \theta) = \sum_{(m,n)\in\mathcal{S}(\tau^{(b)})} \lambda^{n-m} \left[\log \frac{F_\phi(x_m) \prod_{l=m}^{n-1} P_F(x_{l+1} \mid x_l; \theta_f)}{F_\phi(x_n) \prod_{l=m}^{n-1} P_B^{\text{ref}}(x_l \mid x_{l+1})} \right]^2$$

  $g \leftarrow \nabla_\theta \frac{1}{B} \sum_{b=1}^{B} \mathcal{L}(\tau^{(b)}; \theta)$
  $\theta \leftarrow \text{Adam}(\theta, g, \eta)$
until convergence

Reduction of gradient variance

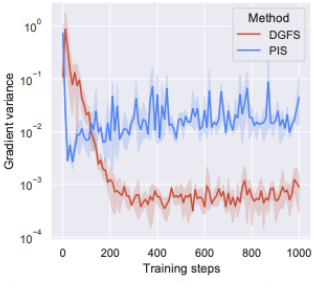


Figure: Gradient variance comparison between DGFS and PIS. DGFS shows significantly lower variance, leading to more stable training.

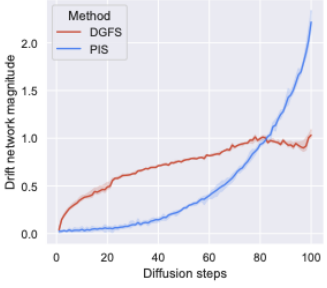


Figure: Drift network magnitude of DGFS and PIS.

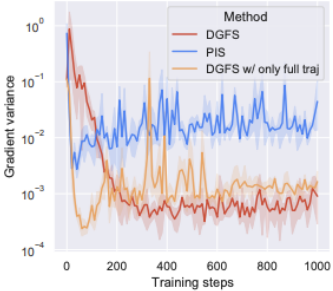


Figure: Gradient variance of DGFS, PIS, and DGFS trained with only full paths

Accurate partition function Z estimation

How to Estimate Z. The partition function $Z = \int \mu(x)\, dx$ normalizes the density. Since we sample trajectories from the learned process Q, we estimate Z using importance sampling on the terminal states.
Log-partition function estimator:

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

Intuition (Simple).

- $S(\tau; \theta)$: Weight showing how well the trajectory matches the target (higher if it ends in high-μ regions).
- Sample many trajectories from Q, compute their weights, and average in log space.
- This gives a lower bound on $\log Z$ because Q approximates the target, providing an estimate without direct π samples.

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[5]	$0.003_{\pm 0.0005}$	$0.0022_{\pm 0.0005}$	$0.032_{\pm 0.004}$	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[6]
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: The lower the better, DGFS achieves the lowest bias in estimating the partition function across various benchmarks compared to PIS and DDS.

Convergence Guarantees

Convergence Guarantees for DGFS.

- Previous studies De Bortoli et al. 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. 2022b prove that a perfectly learned score corresponds to zero GFlowNet training loss E. Bengio, Jain, et al. 2021; E. Bengio, S. Bengio, et al. 2023, ensuring a well-trained DGFS accurately samples from the target distribution.

Mode coverage results

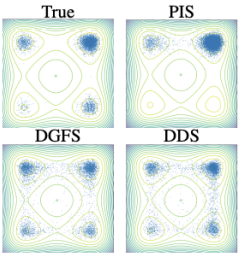


Figure: Manywell plots. DGFS and DDS but not PIS recover all modes

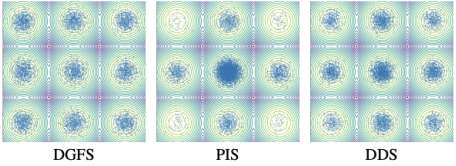


Figure: MoG visualization of DGFS and other diffusion-based samplers shows that DGFS could capture the diverse modes well. The contours display the landscape of the target density

Off-policy training capability (a voir si on garde cette slide)

Off-Policy Training in DGFS. Unlike previous KL-based methods restricted to on-policy training (high-variance off-policy via importance sampling), DGFS supports off-policy exploration without it, as its objectives (Eq. 9/14) don't require samples from a specific distribution **malkin2023**; **lahlou2023**.
Exploration Strategy. Use a larger variance coefficient $\tilde{\sigma} > \sigma$ in the forward process during rollouts to capture diverse modes, improving multimodal sampling.
Example: MoG+ Task. With distant modes, setting $\tilde{\sigma} = 2$ (vs. $\sigma = 1$) achieves better mode coverage and lower Z estimation bias.
DGFS+ Experiment. Linearly anneal $\tilde{\sigma}$ from 1.1 to 1 over first 1000 steps, enhancing performance in some scenarios as a tunable hyperparameter.

Conclusion

Summary. Diffusion Generative Flow Sampler (DGFS), a novel algorithm that trains diffusion models to sample from given unnormalized target densities. Different from prior works (Path Integral Sampler and Denoising Diffusion Sampler) that could only learn from complete diffusion chains, DGFS can update its parameters with only partial specification of the stochastic process trajectory. These features help DGFS benefit from efficient credit assignment and thus achieve better partition function estimation bias in the sampling benchmarks while keeping the theoretical guarantees of convergence from diffusion-based samplers.

Questions and hopefully answers :)

Questions?

References I

- Bengio, Emmanuel, Salem Bengio, et al. (2023). "GFlowNet Foundations". In: *Journal of Machine Learning Research* 24, pp. 1–55.
- Bengio, Emmanuel, Moksh Jain, et al. (2021). "Flow Network based Generative Models for Non-Iterative Diverse Candidate Generation". In: *Advances in Neural Information Processing Systems* 34, pp. 27381–27394.
- Chen, Nan et al. (2022). "On the convergence of score-based generative modeling". In: *arXiv preprint arXiv:2206.04114*.
- De Bortoli, Valentin et al. (2022). "Convergence of score-based generative modeling for general data distributions". In: *Advances in Neural Information Processing Systems* 35, pp. 21892–21904.

References II

- Lee, Hyeontae et al. (2022). "Convergence of denoising diffusion models under the manifold hypothesis". In: *arXiv preprint arXiv:2205.12551*.
- Zhang, Dinghuai et al. (2022a). "Path Integral Sampler: Diffusion-based Sampling for Unnormalized Densities". In: *Advances in Neural Information Processing Systems* 35, pp. 11738–11751.
- — (2022b). "Path Integral Sampler: Diffusion-based Sampling for Unnormalized Densities". In: *Advances in Neural Information Processing Systems* 35, pp. 11738–11751.