Diffusion Generative Flow Samplers: Improving Learning Signals Through Partial Trajectory Optimization (ICLR 2024)

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

Thomas Mousseau

October 28, 2025

Overview

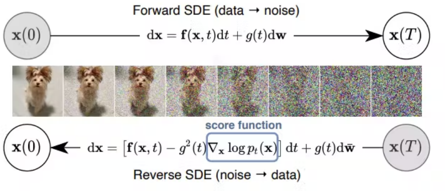
Generative Modeling

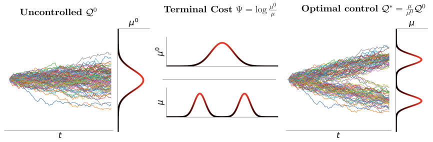
Task

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)


Forward SDE (data → noise)
$\mathbf{x}(0)$
$\mathrm{d}\mathbf{x} = \mathbf{f}(\mathbf{x}, t)\mathrm{d}t + g(t)\mathrm{d}\mathbf{w}$
$\mathbf{x}(T)$
score function
$\mathbf{x}(0)$
$\mathrm{d}\mathbf{x} = [\mathbf{f}(\mathbf{x}, t) - g^2(t)\boxed{\nabla_{\mathbf{x}} \log p_t(\mathbf{x})}]\,\mathrm{d}t + g(t)\mathrm{d}\bar{\mathbf{w}}$
$\mathbf{x}(T)$
Reverse SDE (noise → data)



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.

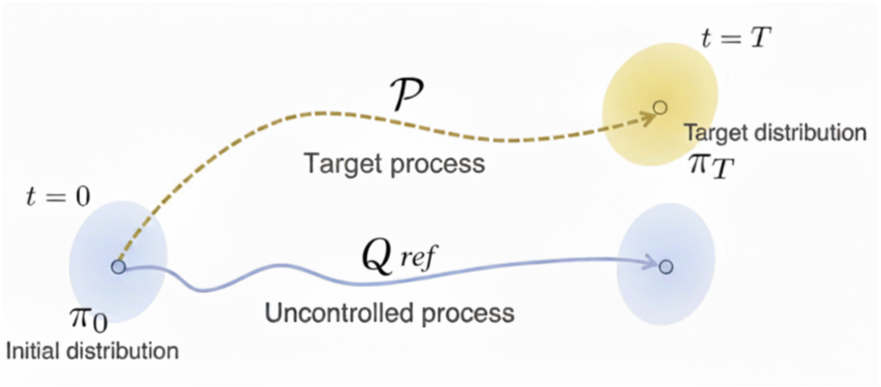
Goal. We seek a *sampler* (similar to MCMC/VI) that produces calibrated samples and, ideally, estimates Z *without* any dataset from π.

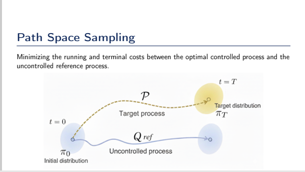
Chemistry (molecule conformers).

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.

Path Space Sampling

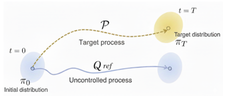
Minimizing the running and terminal costs between the optimal controlled process and the uncontrolled reference process.





Path Space Sampling

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).

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

Controlled process and marginals.

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

$$p_n(x) = \int Q(x_{0:N})\, dx_{0:n-1} dx_{n+1:N}, \text{ no closed form.}$$

Uncontrolled/Reference forward transition (zero drift).

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

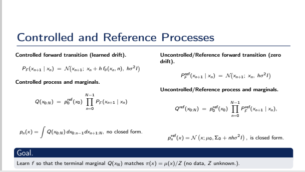
Uncontrolled/Reference process and marginals.

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

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

Goal.

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



Controlled and Reference Processes

Controlled forward transition (learned drift).

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

Controlled process and marginals.

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

$$p_n(x) = \int Q(x_{0:N})\, dx_{0:n-1}\, dx_{n+1:N}, \text{ no closed form.}$$

Uncontrolled/Reference forward transition (zero drift).

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

Uncontrolled/Reference process and marginals.

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

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

Goal.

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

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 ($e.g., p_n^{ref}(x) = N(x; mu_0, cov_0 + nh\sigma^2 I)$). 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 xn 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

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.

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

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.

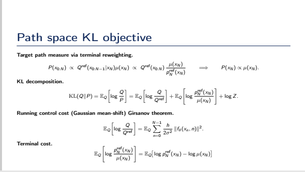
$$\mathrm{KL}(Q\|P) = \mathbb{E}_Q\left[\log \frac{Q}{P}\right] = \mathbb{E}_Q\left[\log \frac{Q}{Q^{\text{ref}}}\right] + \mathbb{E}_Q\left[\log \frac{p_N^{\text{ref}}(x_N)}{\mu(x_N)}\right] + \log Z.$$

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.

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



Path space KL objective

Target path measure via terminal reweighting.

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

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.

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

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

Credit Assignment Problem in SOC objective

SOC Discrete-time objective

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

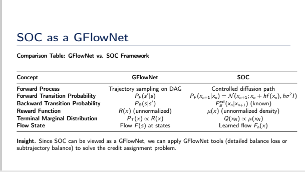
This objective is used on the seminar paper *Path Integral Sampler: Diffusion-based Sampling for Unnormalized Densities* Zhang et al. 2022 which presented the sampling from unnormalized densities as a stochastic optimal control problem.

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 (detailed balance loss or subtrajectory balance) to solve the credit assignment problem.



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 (detailed balance loss or subtrajectory balance) to solve the credit assignment problem.

make a mental note to comeback to professor hernandez comment on Tristan's Deleu presentation when someone asked: "The if GFlowNets are just Reinfrocement Learning, then why keep doing research on this framework?" Professor Hernandez reponded 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.

Adressing Credit Assignment via Target Process

Target process necessitates terminal reweighting...

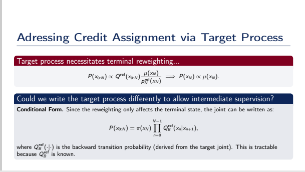
$$P(x_{0:N}) \propto Q^{\text{ref}}(x_{0:N}) \frac{\mu(x_N)}{p_N^{\text{ref}}(x_N)} \implies P(x_N) \propto \mu(x_N).$$

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.



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.

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

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:

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

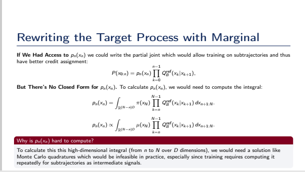
But There's No Closed Form for $p_n(x_n)$. To calculate $p_n(x_n)$, we would need to compute the integral:

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

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

Why is $p_n(x_n)$ hard to compute?

To calculate this this high-dimensional integral (from n to N over D dimensions), we would need a solution like Monte Carlo quadratures which would be infeasible in practice, especially since training requires computing it repeatedly for subtrajectories as intermediate signals.



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:

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

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

Why is $p_n(x_n)$ hard to compute?

To calculate this this high-dimensional integral (from n to N over D dimensions), we would need a solution like Monte Carlo quadratures which would be infeasible in practice, especially since training requires computing it repeatedly for subtrajectories as intermediate signals.

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

Detailed and SubTB Balance with Learned Flow

Detailed Balance from Trajectory Balance condition.

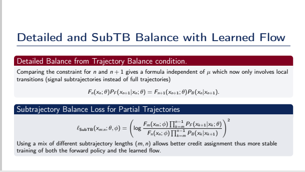
Comparing the constraint for n and $n+1$ gives a formula independent of μ which now only involves local transitions (signal subtrajectories instead of full trajectories)

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

Subtrajectory Balance Loss for Partial Trajectories

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

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.



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

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

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.

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

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

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

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

- τ: Full trajectory $(x_0,\dots,x_N)$.
- 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: $\mu(\cdot)$, $\bar{\sigma}$, $N$, $\lambda$, $B$, $\eta$
 1: Init $\theta = (\theta_f, \phi)$
 2: repeat
 3:     Sample trajectories:
 4:     for $b = 1$ to $B$ do
 5:         $\tau^{(b)} = (x_0^{(b)}, \dots, x_N^{(b)})$ under $x_{n+1} = x_n + hf_{\theta_f}(x_n, n) + \sqrt{h}\bar{\sigma}\varepsilon_n$, $\varepsilon_n \sim \mathcal{N}(0, I)$
 6:     end for
 7:     Build subtrajectories: $\mathcal{S}(\tau^{(b)})$ of $(m, n)$ with $0 \le m < n \le N$
 8:     Compute SubTB loss:
            $$\mathcal{L}(\tau^{(b)}; \theta) = \sum_{(m,n)\in\mathcal{S}(\tau^{(b)})} \lambda^{n-m} \left[\log \frac{F_\phi(x_m) \prod_{l=m}^{n-1} P_F(x_{l+1} \mid x_l; \theta_f)}{F_\phi(x_n) \prod_{l=m}^{n-1} P_B^{\mathrm{ref}}(x_l \mid x_{l+1})}\right]^2$$
 9:     $g \leftarrow \nabla_\theta \frac{1}{B} \sum_{b=1}^{B} \mathcal{L}(\tau^{(b)}; \theta)$
10:     $\theta \leftarrow \mathrm{Adam}(\theta, g, \eta)$
11: until convergence
```

Reduction of gradient variance

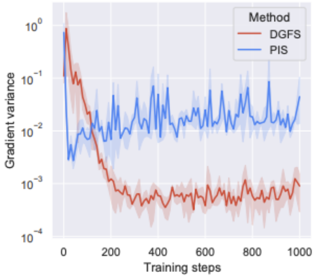


Figure: Gradient variance comparison between DGFS and PIS. DGFS shows significantly lower variance, leading to more stable training.

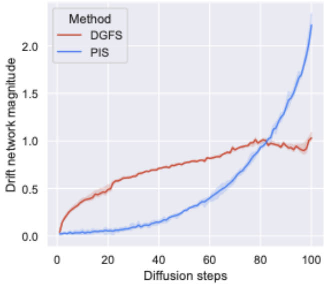


Figure: Drift network magnitude of DGFS and PIS.

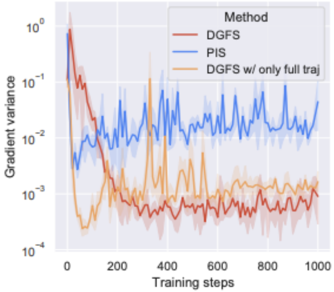


Figure: Gradient variance of DGFS, PIS, and DGFS trained with only full paths

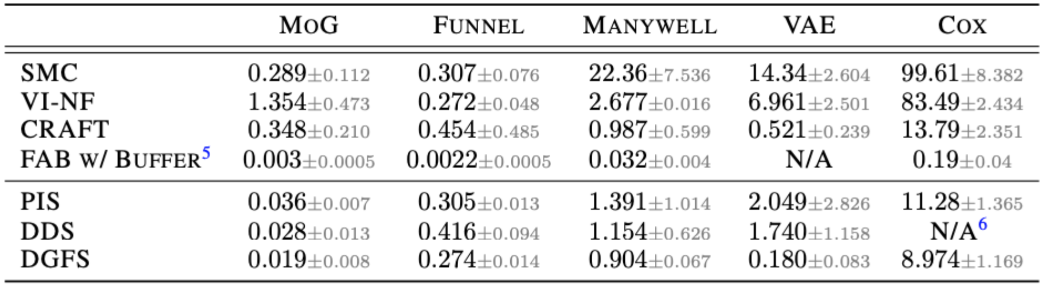
Accurate partition function Z estimation

Partition Function Estimation

$$\log \sum_{\tau}^{\mathrm{T}} \exp(\log(P(\tau)) - \log(Q(\tau))) - \log \mathrm{T} \leq \log Z, \quad \tau \sim Q(\cdot;\theta).$$

The log-sum-exp trick is used as a smooth approximation to the maximum, providing a lower bound on $\log Z$.

Partition function estimation results



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

Mode coverage results

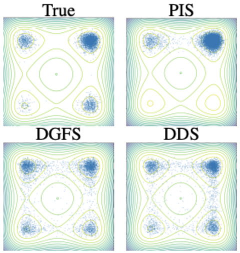


Figure: Manywell plots. DGFS and DDS but not PIS recover all modes

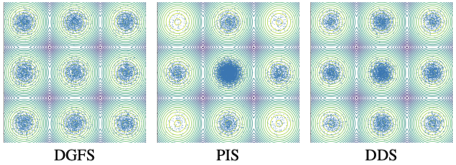


Figure: MoG visualization of DGFS and other diffusion-based samplers shows that DGFS could capture the diverse modes well. The contours display the landscape of the target density

Strengths of DGFS

Strengths

- Works from unnormalized targets. Trains directly with $\mu(x)$ (and optionally $\nabla \log \mu$), no dataset or partition function Z needed.
- Well-calibrated endpoints. At convergence the terminal marginal $p_N(x)$ matches $\pi(x) \propto \mu(x)$; samples come with the right probabilities.
- Intermediate learning signals. Subtrajectory Balance (SubTB) provides supervision on partial paths $x_{m:n}$, solving the "terminal-only signal" issue of path-KL samplers (better temporal credit assignment, lower variance).
- Fixed, closed-form backward. By terminal tilting, the target's backward kernel equals the reference bridge (Gaussian, analytic), giving stable, non-circular constraints.

Weaknesses of DGFS

Weaknesses / Limitations

- Reliance on Gaussian reference. Closed-form backward requires (near-)Gaussian reference with constant covariance; non-Gaussian or state-dependent noise breaks the neat math.
- Two nets to train. Must learn both the forward drift and the flow F_n; more parameters, more tuning (time embeddings, loss weights).
- μ-query cost. Frequent evaluations of $\mu(x)$ (and possibly $\nabla \log \mu$) can be expensive; SubTB helps, but large-scale tasks need caching/replay/prioritization.
- Coverage still hard in high-D. Even with SubTB, exploring multi-modal, high-barrier landscapes can stall without tempering, momentum, or preconditioning.
- State-space must match target. DGFS assumes a fixed continuous state space $\mathbb{R}^D$; discrete/structured spaces need bespoke references/backwards or relaxations.
- Hyperparameter sensitivity. Noise scale $\bar{\sigma}$, step size h, SubTB length-mixing λ, and trajectory-sampling strategy materially affect stability/convergence.

Personal Opinion on this work

Summary of Contributions

- Addresses a key issue in path-space samplers: PIS and DDS needed full trajectory signals for learning.
- Introduces subtrajectory signals from GFlowNet to reduce the credit assignment problem.

Future Challenges

- For real-world applications (chemistry and physics) with unnormalized densities, better mode coverage is essential due to high dimensionality.
- Current benchmarks show progress, but more work is needed on mode discovery and sample efficiency.

Future Work

Rare Event Sampling. Extend DGFS to efficiently sample from rare event distributions, where the target density is concentrated in low-probability regions of the reference process.

Movement from Path Space to Latent Space. Many diffusion models operate in latent spaces for high-dimensional data (e.g., images). Adapting DGFS to learn flows in latent spaces could improve scalability and efficiency. It means they don't have to start with $\mathbb{R}^d$ as state space but could work in any latent space.

Introduce trick from RL. Incorporate techniques from reinforcement learning, such as experience replay with buffer, off-policy training to improve exploration and sample efficiency in DGFS.
could add something with the partition function estimation Z

Questions and hopefully answers :)

Questions?

References I

Zhang, Dinghuai et al. (2022). "Path Integral Sampler: Diffusion-based Sampling for Unnormalized Densities". In: *Advances in Neural Information Processing Systems* 35, pp. 11738–11751.