MetaGFN: Exploring Distant Modes with Adapted Metadynamics for Continuous GFlowNets

Anonymous Authors[1]

Abstract

Generative Flow Networks (GFlowNets) are a class of generative models that sample objects in proportion to a specified reward function through a learned policy. They can be trained either on-policy or off-policy, needing a balance between exploration and exploitation for fast convergence to a target distribution. While exploration strategies for discrete GFlowNets have been studied, exploration in the continuous case remains to be investigated, despite the potential for novel exploration algorithms due to the local connectedness of continuous domains. Here, we introduce Adapted Metadynamics, a variant of metadynamics that can be applied to arbitrary black-box reward functions on continuous domains. We use Adapted Metadynamics as an exploration strategy for continuous GFlowNets. We show two continuous domains where the resulting algorithm, MetaGFN, accelerates convergence to the target distribution and discovers more distant reward modes than previous off-policy exploration strategies used for GFlowNets.

1. Introduction

Generative Flow Networks (GFlowNets) are a type of generative model that samples from a discrete space χ by sequentially constructing objects via actions taken from a learned policy P_F (2). The policy $P_F(s, s')$ specifies the probability of transitioning from some state s to some other state s'. The policy is parameterised and trained so that, at convergence, the probability of sampling an object $x \in \chi$ is proportional to a specified reward function $R(x)$. GFlowNets offer advantages over more traditional sampling methods, such as Markov chain Monte Carlo (MCMC), by learning an amortised sampler, capable of single-shot generation of samples from the desired distribution. Since GFlowNets learn a parametric policy, they are able to generalise across states, resulting in higher performance across various tasks (2; 24; 40; 15; 9; 16; 13; 39; 34) and applications to conditioned molecule generation (34), maximum likelihood estimation in discrete latent variable models (13), structure learning of Bayesian networks (9), scheduling computational operations (39), and discovering reticular materials for carbon capture (7).

Although originally conceived for discrete state spaces, GFlowNets have been extended to more general state spaces, such as entirely continuous spaces, or spaces that are hybrid discrete-continuous (19). In the continuous setting, given the current state, the policy specifies a continuous probability distribution over subsequent states, and the probability density over states $x \in \chi$ sampled with the policy is proportional to a reward density function $r(x)$. The continuous domain unlocks more applications for GFlowNets, such as molecular conformation sampling (37) and continuous control problems (22).

GFlowNets are trained in a manner similar to reinforcement learning agents. Trajectories of states are generated either on-policy or off-policy, with the terminating state $x \in \chi$ providing a reward signal for informing a gradient step on the policy parameters. GFlowNets therefore suffer from the same training pitfalls as reinforcement learning. One such issue is slow temporal credit assignment, which has thus far been addressed by designing more effective loss functions, such as detailed balance (3), trajectory balance (24) and sub-trajectory balance (23).

Besides loss functions, another aspect of GFlowNet training is the exploration strategy for acquiring training samples. Exclusively on-policy learning is generally inadequate as it leads to inefficient exploration of new modes. More successful strategies therefore rely on off-policy exploration. For the discrete setting, numerous exploration strategies have been proposed including ϵ-noisy with a uniform random policy, tempering, Generative Augmented Flow Networks (GAFN) (28), Thompson sampling (30) and Local Search GFlowNets (18). While these approaches can be generalised to the continuous domain, there is no literature benchmarking their effectiveness in this setting.

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

Sampling in the continuous setting is a common occurrence in various domains such as molecular modelling (11; 38) and Bayesian inference (32). The local connectedness of a continuous domain allows for novel exploration strategies that are not directly applicable in the discrete setting. In this work we create MetaGFN, an exploration algorithm for continuous GFlowNets inspired by metadynamics, an enhanced sampling method widely used for molecular modelling (21). The main contributions of this work are:

- Presenting MetaGFN, an algorithm created by adapting metadynamics to black box rewards and continuous GFlowNets;
- Proving that the proposed Adapted Metadynamics is consistent and reduces to standard metadynamics in a limit;
- Showing empirically that MetaGFN outperforms existing GFlowNets exploration strategies.

The rest of the paper is as follows. In Section 2 we review the theory of discrete and continuous GFlowNets. We present Adapted Metadynamics and MetaGFN in Section 3. In Section 4, we evaluate MetaGFN against other exploration strategies, showing that MetaGFN outperforms existing exploration strategies in two continuous environments. We finish with limitations and conclusions in Sections 5 and 6. Code for MetaGFN is available at [link in camera-ready].

2. Preliminaries

2.1. Discrete GFlowNets

In a GFlowNet, the *network* refers to a directed acyclic graph (DAG), denoted as $G = (\mathcal{S}, \mathcal{A})$. Nodes represent *states* $s \in \mathcal{S}$, and edges represent *actions* $s \to s' \in \mathcal{A}$ denoting one-way transitions between states. The DAG has two distinguishable states: a unique *source state* s_0, that has no incoming edges, and a unique *sink state* $\perp$, that has no outgoing edges. The set of states, $\chi \subset \mathcal{S}$, that are directly connected to the sink state are known as *terminating states*. GFlowNets learn forward transition probabilities, known as a *forward policy* $P_F(s'|s)$, along the edges of the DAG so that the resulting marginal distribution over the terminal states, denoted as $P^{\perp}(x)$, is proportional to a given *reward function* $R : \chi \to \mathbb{R}$. GFlowNets also introduce additional learnable objects, such as a *backward policy* $P_B(s|s')$, which is a distribution over the parents of any state of the DAG, to create losses that train the forward policy. Objective functions for GFlowNets include flow matching (FM), detailed balance (DB), trajectory balance (TB) and subtrajectory balance (STB) (2; 3; 24; 23). During training, the parameters of the flow objects are updated with stochastic gradients of the objective function applied to batches of trajectories. These trajectory batches can be obtained either directly from the current forward policy or from an alternative algorithm that encourages exploration. These approaches are known as *on-policy* and *off-policy* training respectively.

2.2. Continuous GFlowNets

Continuous GFlowNets extend the generative problem to continuous spaces (19), where the analogous quantity to the DAG is a measurable pointed graph (MPG) (27). MPGs can model continuous spaces (e.g., Euclidean space, spheres, tori), as well as hybrid spaces, with a mix of discrete and continuous components, as often encountered in robotics, finance, and biology (26; 36; 5).

Definition 2.1 (Measurable pointed graph (MPG))**.** Let $(\bar{\mathcal{S}}, \mathcal{T})$ be a topological space, where $\bar{\mathcal{S}}$ is the *state space*, $\mathcal{T}$ is the set of open subsets of $\bar{\mathcal{S}}$, and Σ is the Borel σ-algebra associated with the topology of $\bar{\mathcal{S}}$. Within this space, we identify: the *source state* $s_0 \in \bar{\mathcal{S}}$ and *sink state* $\perp \in \bar{\mathcal{S}}$, both distinct and isolated from the rest of the space. On this space we define a *reference transition kernel* $\kappa : \bar{\mathcal{S}} \times \Sigma \to [0, +\infty)$ and a *backward reference transition kernel* $\kappa^b : \bar{\mathcal{S}} \times \Sigma \to [0, +\infty)$. The support of $\kappa(s, \cdot)$ are all open sets accessible from s. The support of $\kappa^b(s, \cdot)$ are all open sets where s is accessible from. Additionally, these objects must be well-behaved in the following sense:

(i) *Continuity*: For all $B \in \Sigma$, the mapping $s \mapsto \kappa(s, B)$ is continuous.

(ii) *No way back from the source*: The backward reference kernel has zero support at the source state, i.e. for all $B \in \Sigma$, $\kappa^b(s_0, B) = 0$.

(iii) *No way forward from the sink*: When at the sink, applying the forward kernel keeps you there, i.e. $\kappa(\perp, \cdot) = \delta_{\perp}(\cdot)$, where $\delta_{\perp}$ is the Dirac measure of the sink state.

(iv) *A fully-explorable space*: The number of steps required to be able to reach any measurable $B \in \Sigma$ from the source state with the forward reference kernel is bounded.

The set of objects $(\bar{\mathcal{S}}, \mathcal{T}, \Sigma, s_0, \perp, \kappa, \kappa^b)$ then defines an MPG.

Note that the support of $\kappa(s, \cdot)$ and $\kappa^b(s, \cdot)$ are analogous to the child and parent sets of a state s in a DAG. Similarly, a discrete GFlowNet's DAG satisfies discrete versions of (ii), (iii), and (iv).

The set of *terminating states* χ are the states that can transition to the sink, given by $\chi = \{s \in \mathcal{S} : \kappa(s, \{\perp\}) > 0\}$, where $\mathcal{S} := \bar{\mathcal{S}} \setminus \{s_0\}$. *Trajectories* τ are sequences of states that run from source to sink, $\tau = (s_0, \ldots, s_n, \perp)$. The

forward Markov kernel $P_F : \bar{\mathcal{S}} \times \Sigma \to [0, \infty)$ and *backward Markov kernel* $P_B : \bar{\mathcal{S}} \times \Sigma \to [0, \infty)$ have the same support as $\kappa(s, \cdot)$ and $\kappa^b(s, \cdot)$ respectively, where being a Markov kernel means states are mapped to probability measure, hence $\int_{\bar{\mathcal{S}}} P_F(s, ds') = \int_{\bar{\mathcal{S}}} P_B(s, ds') = 1$. A *flow* F is a tuple $F = (f, P_F)$, where $f : \Sigma \to [0, \infty)$ is a *flow measure*, satisfying $f(\{\perp\}) = f(s_0) = Z$, where Z is the *total flow*.

The *reward measure* is a positive and finite measure R over the terminating states χ, we denote the density of this reward measure as r. A flow F is said to satisfy the *reward-matching conditions* if

$$R(dx) = f(dx)P_F(x, \{\perp\}).$$

If a flow satisfies the reward-matching conditions and trajectories are recursively sampled from the Markov kernel P_F starting at s_0, the resulting *measure over terminating states*, $P^{\perp}(B)$, is proportional to the reward: $P^{\perp}(B) = \frac{R(B)}{R(\chi)}$ for any B in the σ-algebra of terminating states (19).

Objective functions for discrete GFlowNets generalise to continuous GFlowNets. However, in the continuous case, the *forward policy* $\hat{p}_F : \mathcal{S} \times \bar{\mathcal{S}} \to [0, \infty)$, *backward policy* $\hat{p}_B : \mathcal{S} \times \bar{\mathcal{S}} \to [0, \infty)$ and *parameterised flow* $\hat{f} : \mathcal{S} \to [0, \infty)$ parameterise the P_F, P_B transition kernels and flow measure f on an MPG. Discrete GFlowNets parameterise log transition probabilities and flows on a DAG. In this work, we consider DB, TB and STB losses. For a complete trajectory τ, the TB loss can be written as

$$L_{TB}(\tau) = \left(\log \frac{Z_\theta \prod_{t=0}^{n} \hat{p}_F(s_t, s_{t+1}; \theta)}{r(s_n) \prod_{t=0}^{n-1} \hat{p}_B(s_{t+1}, s_t; \theta)} \right)^2,$$

where Z_θ is the parameterised total flow (see Appendix A for the DB and STB loss functions).

2.3. Exploration strategies for GFlowNets

GFlowNets can reliably learn using off-policy trajectories, a key advantage over hierarchical variational models (25). For optimal training, it is common to use a replay buffer and alternate between on-policy and off-policy (exploration) batches (33). Exploration strategies for discrete GFlowNets include ϵ-noisy with a uniform random policy, tempering, Generative Augmented Flow Networks (GAFN) (28), Local Search GFlowNets (18), and Thompson sampling (TS), which outperforms the others in grid and bit sequence domains (30). TS aims to bias exploration in regions where there is high uncertainty. When the forward policy is parameterised as an MLP, this is achieved using an ensemble of $K \in \mathbb{Z}^+$ policy heads with a common network torso. During training, an ensemble member is randomly sampled and used to generate a trajectory τ. In a training batch, each ensemble member is included with probability p and parameters are updated by taking a gradient step on the total loss of τ over all included members.

No comparative literature exists on exploration strategies for continuous GFlowNets, but many methods can be adapted, and we do so here. For example, for TS in the continuous setting, policy heads parameterise forward policy functions instead of log probabilities. Another strategy unique to the continuous setting is what we call *noisy exploration*. This strategy involves introducing an additive noise parameter, denoted as $\bar{\sigma}$, to the variance parameters in the forward policy distribution, where the value of $\bar{\sigma}$ is scheduled to gradually decrease to zero over the course of training.

2.4. Metadynamics

Molecular dynamics (MD) uses *Langevin dynamics* (LD) (29), a stochastic differential equation modelling particle motion with friction and random fluctuations, to simulate atomic trajectories that ergodically sample a molecule's Gibbs measure, $\rho_\beta(x) \propto e^{-\beta V(x)}$. Here, x and p are atomic positions and momenta, $V(x)$ is the molecular potential, and β is thermodynamic beta.[1] If the potential $V(x)$ has multiple local minima, then unbiased LD can get trapped in these minima, which can lead to inefficient sampling. *Metadynamics* is an algorithm that enhances sampling by regularly depositing repulsive Gaussian bias potentials at the center of an evolving LD trajectory (21). The conservative component of the LD force is then given by the negative gradient of the total potential, i.e. $-\nabla V_{\text{total}} = -\nabla(V + V_{\text{bias}})$, where V_{bias} is the cumulative bias. Bias potentials only vary in the direction of user-specified low-dimensional *collective variables* (CVs), $z(x) : \mathcal{X} \to \mathcal{Z}$, mapping from the original space $\mathcal{X}$ to the *CV space* $\mathcal{Z}$. For biomolecules, typical CVs include protein backbone angles or distances between charge centers - quantities that play a central role in rare-event transitions. As the bias potential progressively fills up the potential landscape, energetic barriers are reduced, thus accelerating exploration, eventually ensuring uniform diffusion in CV space. In practice, the bias is defined on a regular grid, which limits CV space dimensionality to 5-10 due to exponential memory costs. Thus, identifying good CVs is crucial for effective metadynamics simulations in applications such as drug discovery, chemistry and materials science (8; 20). In this work, we adapt the original metadynamics algorithm discussed above. The method has also seen numerous extensions. For a detailed review, see Bussi and Laio (2020) and the references therein (6).

[1] We review Langevin dynamics in Appendix B.

3. MetaGFN: Adapted Metadynamics for GFlowNets

Training GFlowNets in high-dimensional continuous spaces requires exploration, especially if valuable reward peaks are separated by large regions of low reward. In some tasks, apriori knowledge of the principal manifold directions of the reward measure can reduce the effective dimension of the search. This is where exploration algorithms that guarantee uniform sampling in that manifold, such as metadynamics, become most effective. With this intuition in mind, we adapt metadynamics to the black-box reward setting of continuous GFlowNets.

Assumptions We assume that χ is a manifold (locally homeomorphic to Euclidean space) and that the reward density r is bounded and L1-integrable function over χ with at most finitely many discontinuities. This implies the target density over terminal states, $\rho = r(x)/\int_\chi r(x)\,dx$, can be expressed as a Gibbs distribution: $\rho = \exp(-\beta' V(x))/\int_\chi r(x)\,dx$, where $V(x) = -\frac{1}{\beta'}\ln r(x)$, with fixed $\beta' > 0$, is a potential with at most finitely many discontinuities. If $r(x)$ is multimodal, then $V(x)$ has multiple minima. **Our aim is to explore $V(x)$ using a variant of metadynamics, thereby generating off-policy, high-reward terminal states that encourage the GFlowNet to eventually sample all the modes of the reward density**.

Kernel density potential Metadynamics force computations require the gradient of the total potential, where $-\nabla V_{\text{total}} = -\nabla(V + V_{\text{bias}})$. Using the above assumptions, we have $\nabla V = -r(x)/(\beta' \nabla r(x))$. However, $r(x)$ is often a computationally expensive black-box function, and its gradient, $\nabla r(x)$, is in general unknown. While finite differences can estimate $\nabla r(x)$ for smooth, low-dimensional reward distributions, this approach is impractical in high-dimensional spaces. We avoid finite difference calculations by computing a kernel density estimate (KDE) of V instead. We use $\hat{V}$ to denote the KDE estimate. We assume collective variables $z(x) = (z_1(x), \ldots, z_k(x))$, and compute both the KDE and bias potentials in the CV space. Here, each z_i is a one-dimensional coordinate, and $z : \chi \to \mathcal{Z}$, where $\mathcal{Z}$ is k-dimensional. As the potentials are stored on a low-dimensional grid, gradient computations are guaranteed to be cheap relative to the cost of evaluating $r(x)$.

To compute $\hat{V}$, we maintain two separate KDEs: $\hat{N}$ for the histogram of metadynamics states and $\hat{R}$ for cumulative rewards. We update these KDE estimates on the fly at the same time the bias potential is updated. If $\mathcal{Z} \cong \mathbb{R}^k$, we use Gaussian kernels with update rules:[2]

$$\hat{N} \leftarrow \hat{N} + \exp\left(-\frac{1}{2}\sum_{i=1}^{k}\left|\frac{z_i - z_{t,i}}{\sigma_i'}\right|^2\right); \quad (1)$$

$$\hat{R} \leftarrow \hat{R} + r(x_t)\exp\left(-\frac{1}{2}\sum_{i=1}^{k}\left|\frac{z_i - z_{t,i}}{\sigma_i'}\right|^2\right), \quad (2)$$

where $\sigma' = (\sigma_1', \ldots, \sigma_k') \in \mathbb{R}^k$ is the kernel width, x_t is the latest metadynamics sample and $z_{t,i}$ is the corresponding i^{th} CV coordinate of x_t. The KDE potential $\hat{V}$ is then computed as:

$$\hat{V} = -\frac{1}{\beta'}\log\left(\frac{\hat{R}}{\hat{N} + \epsilon} + \epsilon\right),$$

where $\epsilon > 0$. We found empirically that ϵ ensured numerical stability by preventing division by zero and bounding the potential above by $\log(1/\epsilon)/\beta'$, while the ratio of a reward and frequency KDE means that $\hat{V}$ rapidly and smoothly when new modes are discovered. In particular, we prove that $\hat{V}$ eventually discovers all reward modes in the CV space. More precisely,

Theorem 3.1. *If the collective variable $z(x)$ is analytic with a bounded domain, then*

$$\lim_{\epsilon\to 0}\left(\lim_{\sigma'\to 0}\left(\lim_{t\to\infty}\hat{V}(z,t)\right)\right) = V, \quad (3)$$

where $V = V(z') := \int_{\mathcal{X}}\delta(z' - z(x))V(x)\mathrm{d}x$ is the marginal potential in the CV space if $z(x)$ is not invertible. If $z(x)$ is invertible, V is the original potential $V(x)$ in the original coordinates.

The proof is in Appendix C.

Implementation details We set potential energy beta (β') and Langevin dynamics beta (β) to be equal. This reduces the number of parameters but also aids interpretability since β is now inversely proportional to the (unbiased) transition rates between minima of the potential.[3] We also set the bias and kernel widths equal, $\sigma = \sigma'$. This is reasonable since it is the variability of $V(x)$ that determines sensible values for both these parameters. Finally, we set the Langevin dynamics mass parameter to $M = 1$. This is reasonable since mass is non-physical in the GFlowNet context. The dynamic effect of changing M can be emulated by changing other parameters of Langevin dynamics, namely the friction γ, thermodynamic β and integration timestep Δt. The resulting exploration algorithm we call *Adapted Metadynamics* (AM), and is presented in Algorithm 1. Note that

[2] If $\mathcal{Z} \cong \mathbb{T}^k$ (k-torus), we use von Mises distributions instead of Gaussians.

[3] From the Kramer formula; transition rate $\propto \frac{1}{\beta}\exp(\beta\Delta V)$, where $\Delta V \propto \frac{1}{\beta'} = \frac{1}{\beta}$.

the algorithm can be extended to a batch of trajectories, where each metadynamics trajectory evolves independently, but with a shared $\hat{V}$ and V_{bias} which receive updates from every trajectory in the batch. This is the version we use in our experiments - it accelerates exploration and reduces stochastic gradient noise during training.

Algorithm 1 Adapted Metadynamics

Input: Manifold environment of terminating states χ with reward density $r\colon \chi \to \mathbb{R}$. Initial state $(x_t, p_t) \in \chi \times T_{x_t}(\chi)$. Collective variables $z = (z_1, \ldots, z_k)$.
Parameter: Gaussian width $\sigma = (\sigma_1, \ldots, \sigma_k) \in \mathbb{R}^k$. Gaussian height $w > 0$. Stride $n \in \mathbb{Z}^+$. LD parameters: γ, β. Timestep Δt.
$\hat{N} \leftarrow 0$
$\hat{R} \leftarrow 0$
$\hat{V}(z) \leftarrow 0$
$V_{\text{bias}}(z) \leftarrow 0$
every timestep Δt:
  $z_t \leftarrow z(x_t)$
  every n timesteps $n\Delta t$:
    $\hat{N} \leftarrow \hat{N} + \exp\left(-\frac{1}{2}\sum_{i=1}^{k}\left|\frac{z_i - z_{t,i}}{\sigma_i}\right|^2\right)$
    $\hat{R} \leftarrow \hat{R} + r(x_t) \cdot \exp\left(-\frac{1}{2}\sum_{i=1}^{k}\left|\frac{z_i - z_{t,i}}{\sigma_i}\right|^2\right)$
    $\hat{V} \leftarrow -\frac{1}{\beta}\log\left(\frac{\hat{R}}{\hat{N}+\epsilon} + \epsilon\right)$
    $V_{\text{bias}}(z) \leftarrow V_{\text{bias}}(z) + n \cdot \Delta t \cdot w \cdot \exp\left(-\frac{1}{2}\sum_{i=1}^{k}\left|\frac{z_i - z_{t,i}}{\sigma_i}\right|^2\right)$
  compute forces:
    $F \leftarrow -\left(\nabla_z \hat{V}(z)\big|_{z=z_t} + \nabla_z V_{\text{bias}}(z)\big|_{z=z_t}\right) \cdot \nabla_x z\big|_{x=x_t}$
  propagate x_t, p_t by Δt using Langevin dynamics with computed force F (Alg. 2, Appendix D).

Training GFlowNets with Adaptive Metadynamics (MetaGFN) Each Adaptive Metadynamics sample $x_i \in \chi$ is an off-policy terminal state sample. To train a GFlowNet, complete trajectories are required. We generate these by backward sampling from the terminal state, giving a trajectory $\tau = (s_0, s_1, \ldots, s_n = x_i)$, where each state s_{i-1} is sampled from the current backward policy distribution $\hat{p}_B(s_{i-1}|s_i;\theta)$, for i from n to 1. This approach means that the generated trajectory τ has reasonable credit according to the loss function, thereby providing a useful learning signal. However, since this requires a backward policy, this is compatible with DB, TB, and STB losses, but not FM loss. Given the superior credit assignment of the former losses, this is not a limitation (23).

Additionally, we use a replay buffer. Due to the theoretical guarantee that Adaptive Metadynamics will eventually sample all collective variable space (Theorem 3.1), AM samples are ideal candidates for storing in a replay buffer. When storing these trajectories in the replay buffer, there are two obvious choices:

1. Store the entire trajectory the first time it is generated;
2. Store only the Adaptive Metadynamics sample and regenerate trajectories using the current backward policy when retrieving from the replay buffer.

We investigate both options in our experiments. We call the overall training algorithm *MetaGFN*, with pseudocode presented in Algorithm 3, Appendix D.

4. Experiments

4.1. Line environment

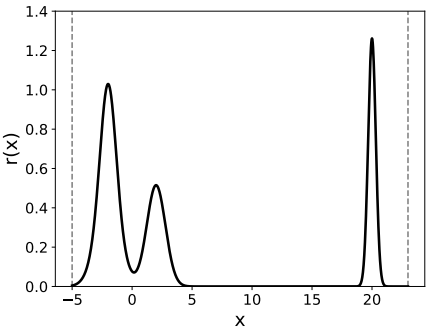

Figure 1: Line Environment reward function, equation (4).

We consider a one-dimensional line environment, with state space $\mathcal{S} = \mathbb{R} \times \{t \in \mathbb{N}, 1 \le t \le 3\}$, where t indexes the position of a state in a trajectory. The source state is $s_0 = (0, 0)$ and trajectories terminate after exactly 3 steps. The terminal states are therefore $\chi = \mathbb{R} \times \{3\} \cong \mathbb{R}$. The reward density, plotted in Figure 1, consists of an asymmetric bimodal peak near the origin and an additional distant lone peak. It is given by the Gaussian mixture distribution:

$$r(x) = \begin{cases} \mathcal{N}(-2.0, 1.0) + \mathcal{N}(-2.0, 0.4) + \\ \quad \mathcal{N}(2.0, 0.6) + \mathcal{N}(20.0, 0.1); & -5 \le x \le 23 \\ 0; & \text{otherwise,} \end{cases} \tag{4}$$

where $\mathcal{N}(\mu, \sigma^2)$ is a Gaussian density with mean μ and variance σ^2. The forward and backward probability transition kernels are a mixture of 3 Gaussian distributions which, along with the flow $\hat{f}$, are parameterised by an MLP.

We compare the following exploration strategies: entirely on-policy (no exploration), noisy exploration, Thompson Sampling, and MetaGFN. For each strategy, we use a replay buffer and alternate between an exploration batch and

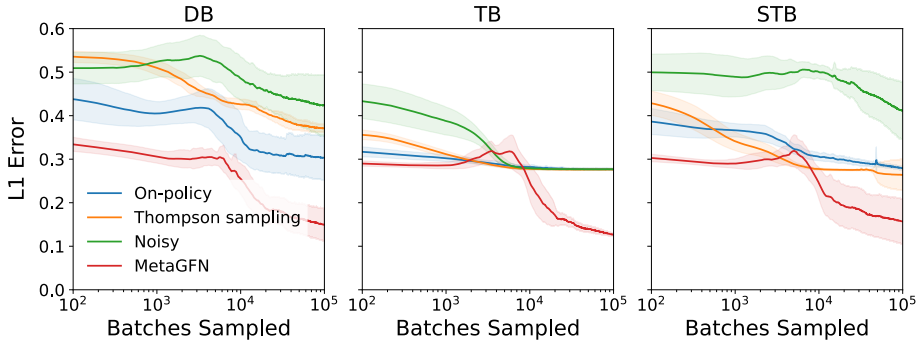

Figure 2: The L1 difference between on-policy and reward distribution during training in the line environment for different loss functions and exploration strategies. Mean is plotted with standard error over 10 repeats. DB - Detailed Balance loss, TB - Trajectory Balance loss, STB - Subtrajectory Balance loss.

a replay buffer batch. For MetaGFN, we use $\texttt{freqRB} = 2$, $\texttt{freqMD} = 10$ and always regenerate MetaGFN training trajectories using the current backward policy. As the environment is one-dimensional, the collective variable is simply $z(x) = x$. We evaluate performance by computing the L1 error between the known reward distribution and the empirical on-policy distribution during training (see Appendix E.1 for the full experimental details).

The results (mean and standard deviation over 10 random seeds) for each loss function and exploration strategy are shown in Figure 2. Among the three loss functions, TB loss has the lowest variance loss profiles, and MetaGFN consistently converges to a lower minimum error than all other exploration strategies. Indeed, MetaGFN was the only method that consistently sampled the distant reward peak at $x = 20$, while other methods plateau in error after locking onto the central modes (Appendix E.2). Despite this, we observed that in the initial stages of training, all exploration strategies found occasional samples from the distant peak. The reason MetaGFN is the only method that converges is that Adaptive Metadynamics manages to *consistently* sample the distant peak, even if the on-policy starts to focus on the central modes. This keeps the replay buffer populated with samples from every reward peak during training, which eventually encourages the on-policy to sample from every mode. The small increase in the loss of MetaGFN around batch number 5×10^3 happens because the on-policy distribution widens when Adaptive Metadynamics first discovers the distant peak. In Appendix E.2, we show further details of Adaptive Metadynamics in this environment and we compare different MetaGFN variants, with and without noise, and with and without trajectory regeneration. We confirm that the version of MetaGFN presented in Figure 2 (no added noise and always regenerate trajectories) is the most robust variant.

4.2. Alanine dipeptide environment

One application of continuous GFlowNets is molecular conformation sampling (37). Here, we train a GFlowNet to sample conformational states of alanine dipeptide (AD), a small biomolecule of 23 atoms that plays a key role in modelling backbone dynamics of proteins (12). The metastable states of AD can be distinguished in a two-dimensional CV space of ϕ and ψ, the two backbone dihedral angles. The resulting free energy surface for AD in explicit water, $V(\phi, \psi)$, obtained after extensive sampling long molecular dynamics simulation is shown in Figure 3. The metastable states (energy minima), in increasing energy, are $P_{||}$, α_R, C_5, α', α_L and α_D.

The state space is $\mathcal{S} = \mathbb{T}^2 \times \{t \in \mathbb{N}, 1 \leq t \leq 3\}$. The source state is $s_0 = P_{||} = (-1.2, 2.68)$ and trajectories terminate after exactly 3 steps. Terminal states are $\chi = \mathbb{T}^2 \times \{3\} \cong \mathbb{T}^2$. The reward density is given by the Boltzmann weight, $r(\phi, \psi) = \frac{1}{Z} \exp(-\beta V(\phi, \psi))$, where Z is the normalisation constant. The forward and backward probability transition kernels are defined as a mixture of 3 bivariate von Mises distributions, parameterised through an MLP. We consider the same exploration strategies and evaluation measure as for the Line Environment (see Appendix F for full experimental details).

The results (mean and standard deviation over 10 random seeds) for each loss function and exploration strategy are shown in Figure 4. For each loss function, models trained with MetaGFN generally converge to a lower minimum error than all other exploration strategies. For TB loss, however, the average L1 error is marginally higher than on-policy training, but this conceals the fact that the best-case error is smaller. Thus, to better understand this result, we examine the best and worst training runs (as measured by L1 error) for TB on-policy and TB MetaGFN, shown in Figure 5. We see that the best run trained with metadynamics can sample from the rare α_L minima, unlike the on-policy run. In the

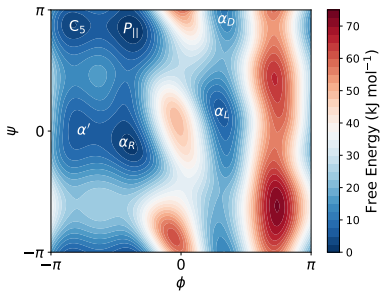

Figure 3: Free energy surface of alanine dipeptide in explicit solvent with all metastable states annnotated.

worst run, MetaGFN fails to converge (although this is rare; only one of 10 runs failed).

In Table 1, we quantify how often the different AD modes were correctly sampled over the different repeats (a mode is correctly sampled if the on-policy distribution also has a mode within the correct basin of attraction). The only mode not correctly sampled by any method is α_D, which has a natural abundance approximately 10 times less frequent than α_L. We see that TB loss with MetaGFN is the only combination that can consistently sample the majority of modes, whilst noisy exploration and Thompson Sampling both perform worse than on-policy in this environment.

Table 1: Number of correct samples of AD modes in trained GFlowNets over 10 independent repeats for DB, STB, and TB loss functions. OP - On-policy and MD - MetaGFN. The α_D mode wasn't sampled in any model due to its low natural frequency.

	DB		STB		TB	
	OP	MD	OP	MD	OP	MD
$P_{\|\|}$	1	7	6	5	10	8
α_R	6	9	7	10	10	9
C_5	2	7	5	6	10	8
α'	6	9	5	10	5	9
α_L	0	1	1	0	0	8

5. Limitations

For metadynamics to be an effective sampler, the CVs must be low-dimensional and bounded, properties that were satisfied in both our experimental environments. Therefore, it is necessary to either know such CVs in advance, assuming they exist or learn them automatically from data (35). An alternative approach would be to learn CVs adaptively by parameterising the CV function by a neural network and updating its parameters by back-propagating through the GFlowNet loss when training on MetaGFN trajectories. A final improvement could be to replace the metadynamics algorithm itself with a variant with smoother convergence properties, such as well-tempered metadynamics (1) or on-the-fly probability enhanced sampling (OPES) (14). We leave these extensions for future work.

6. Conclusions

While exploration strategies for discrete Generative Flow Networks (GFlowNets) have received extensive attention, the methodologies for continuous GFlowNets remain relatively underexplored. To address this gap, we illustrated how metadynamics, a widely used enhanced sampling technique in molecular dynamics, can be adapted as an effective exploration strategy for continuous GFlowNets.

In molecular dynamics, atomic forces can be computed as the gradient of the potential, whereas continuous GFlowNets tackle problems where the reward function is a black box and gradients are inaccessible. We demonstrated how the method could be adapted by updating a kernel density estimate of the reward function on-the-fly, and proved that this is guaranteed to explore the space in an appropriate limit. Our empirical investigations show that MetaGFN offers a computationally efficient means to explore new modes in environments where prior knowledge of collective variables exists. Importantly, this work advocates an approach wherein techniques derived from molecular modelling can be adapted for machine learning tasks. Looking ahead, we anticipate that this could be a fruitful area of cross-disciplinary research, where existing ideas from the enhanced sampling literature can find applications in a broad range of generative modelling and reinforcement learning tasks.

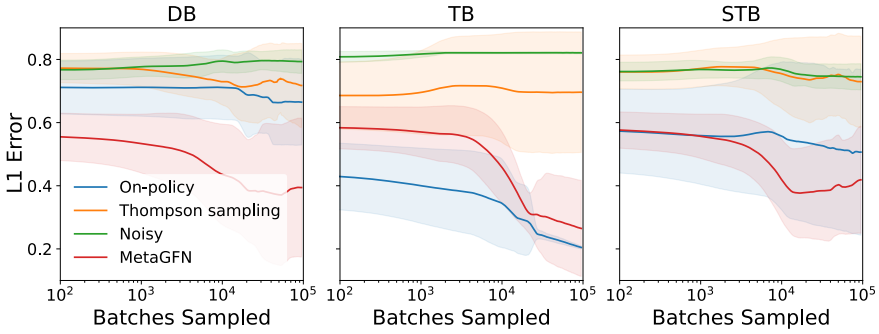

Figure 4: The L1 difference between on-policy and reward distribution during training in the alanine dipeptide environment for different loss functions and exploration strategies. Mean is plotted with standard error over 10 repeats. DB - Detailed Balance loss, TB - Trajectory Balance loss, STB - Subtrajectory Balance loss.

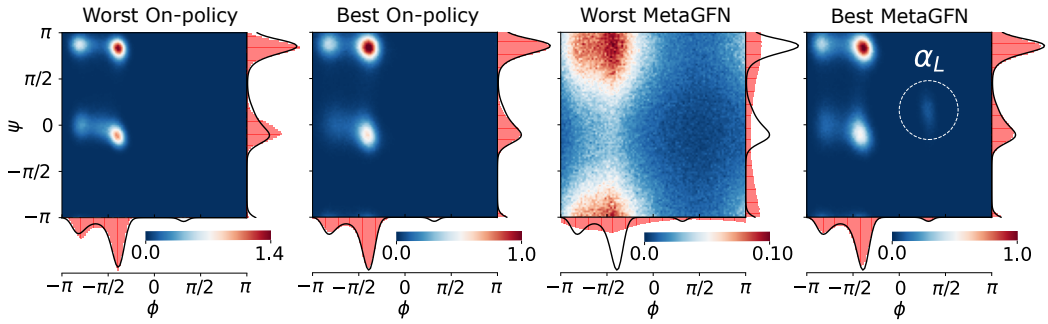

Figure 5: Learned on-policy distribution for TB on-policy and TB MetaGFN training runs. Colour bar shows the probability density. Red histograms show the marginal distribution along the angular coordinates. Black curves show the marginal distributions of the ground truth. In the best case, MetaGFN is able to learn the α_L mode. In the worst case, MetaGFN fails to converge. On-policy training, although more consistent, fails to learn to sample from the α_L mode.

References

[1] Alessandro Barducci, Giovanni Bussi, and Michele Parrinello. Well-Tempered Metadynamics: A Smoothly Converging and Tunable Free-Energy Method. *Physical Review Letters*, 100(2):020603, January 2008.

[2] Emmanuel Bengio, Moksh Jain, Maksym Korablyov, Doina Precup, and Y. Bengio. Flow Network based Generative Models for Non-Iterative Diverse Candidate Generation. *ArXiv*, June 2021.

[3] Yoshua Bengio, T. Deleu, J. E. Hu, Salem Lahlou, Mo Tiwari, and Emmanuel Bengio. GFlowNet Foundations. *ArXiv*, November 2021.

[4] Massimiliano Bonomi, Giovanni Bussi, Carlo Camilloni, Gareth A. Tribello, Pavel Banáš, Alessandro Barducci, Mattia Bernetti, Peter G. Bolhuis, Sandro Bottaro, Davide Branduardi, Riccardo Capelli, Paolo Carloni, Michele Ceriotti, Andrea Cesari, Haochuan Chen, Wei Chen, Francesco Colizzi, Sandip De, Marco De La Pierre, Davide Donadio, Viktor Drobot, Bernd Ensing, Andrew L. Ferguson, Marta Filizola, James S. Fraser, Haohao Fu, Piero Gasparotto, Francesco Luigi Gervasio, Federico Giberti, Alejandro Gil-Ley, Toni Giorgino, Gabriella T. Heller, Glen M. Hocky, Marcella Iannuzzi, Michele Invernizzi, Kim E. Jelfs, Alexander Jussupow, Evgeny Kirilin, Alessandro Laio, Vittorio Limongelli, Kresten Lindorff-Larsen, Thomas Löhr, Fabrizio Marinelli, Layla Martin-Samos, Matteo Masetti, Ralf Meyer, Angelos Michaelides, Carla Molteni, Tetsuya Morishita, Marco Nava, and The PLUMED consortium. Promoting transparency and reproducibility in enhanced molecular simulations. *Nature Methods*, 16(8):670–673, August 2019.

[5] Luca Bortolussi and Alberto Policriti. Hybrid Systems and Biology. In Marco Bernardo, Pierpaolo Degano, and Gianluigi Zavattaro, editors, *Formal Methods for Computational Systems Biology*, pages 424–448, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[6] Giovanni Bussi and Alessandro Laio. Using metadynamics to explore complex free-energy landscapes. *Nature Reviews Physics*, 2:200–212, March 2020. ADS Bibcode: 2020NatRP...2..200B.

[7] Flaviu Cipcigan, Jonathan Booth, Rodrigo Neumann Barros Ferreira, Carine Ribeiro dos Santos, and Mathias Steiner. Discovery of Novel Reticular Materials for Carbon Dioxide Capture using GFlowNets, October 2023. arXiv:2310.07671 [cond-mat].

[8] Marco De Vivo, Matteo Masetti, Giovanni Bottegoni, and Andrea Cavalli. Role of Molecular Dynamics and Related Methods in Drug Discovery. *Journal of Medicinal Chemistry*, 59(9):4035–4061, May 2016. Publisher: American Chemical Society.

[9] Tristan Deleu, António Góis, Chris Emezue, Mansi Rankawat, Simon Lacoste-Julien, Stefan Bauer, and Yoshua Bengio. Bayesian Structure Learning with Generative Flow Networks, June 2022. arXiv:2202.13903 [cs, stat].

[10] Peter Eastman, Jason Swails, John D. Chodera, Robert T. McGibbon, Yutong Zhao, Kyle A. Beauchamp, Lee-Ping Wang, Andrew C. Simmonett, Matthew P. Harrigan, Chaya D. Stern, Rafal P. Wiewiora, Bernard R. Brooks, and Vijay S. Pande. OpenMM 7: Rapid development of high performance algorithms for molecular dynamics. *PLoS computational biology*, 13(7):e1005659, July 2017.

[11] Paul C. D. Hawkins. Conformation Generation: The State of the Art. *Journal of Chemical Information and Modeling*, 57(8):1747–1756, August 2017. Publisher: American Chemical Society.

[12] Jan Hermans. The amino acid dipeptide: Small but still influential after 50 years. *Proceedings of the National Academy of Sciences*, 108(8):3095–3096, February 2011. Publisher: Proceedings of the National Academy of Sciences.

[13] Edward J. Hu, Nikolay Malkin, Moksh Jain, Katie Everett, Alexandros Graikos, and Yoshua Bengio. GFlowNet-EM for learning compositional latent variable models, June 2023. arXiv:2302.06576 [cs, stat].

[14] Michele Invernizzi. OPES: On-the-fly Probability Enhanced Sampling Method. *Il Nuovo Cimento C*, 44(405):1–4, September 2021. arXiv:2101.06991 [physics].

[15] Moksh Jain, Emmanuel Bengio, Alex Hernandez-Garcia, Jarrid Rector-Brooks, Bonaventure F. P. Dossou, Chanakya Ajit Ekbote, Jie Fu, Tianyu Zhang, Michael Kilgour, Dinghuai Zhang, Lena Simine, Payel Das, and Yoshua Bengio. Biological Sequence Design with GFlowNets. In *Proceedings of the 39th International Conference on Machine Learning*, pages 9786–9801. PMLR, June 2022. ISSN: 2640-3498.

[16] Moksh Jain, Sharath Chandra Raparthy, Alex Hernandez-Garcia, Jarrid Rector-Brooks, Yoshua Bengio, Santiago Miret, and Emmanuel Bengio. Multi-Objective GFlowNets, July 2023. arXiv:2210.12765 [cs, stat].

[17] William L. Jorgensen, Jayaraman Chandrasekhar, Jeffry D. Madura, Roger W. Impey, and Michael L. Klein. Comparison of simple potential functions for simulating liquid water. *The Journal of Chemical Physics*, 79(2):926–935, July 1983.

[18] Minsu Kim, Taeyoung Yun, Emmanuel Bengio, Dinghuai Zhang, Yoshua Bengio, Sungsoo Ahn, and Jinkyoo Park. Local Search GFlowNets, March 2024. arXiv:2310.02710 [cs, stat].

[19] Salem Lahlou, Tristan Deleu, Pablo Lemos, Dinghuai Zhang, Alexandra Volokhova, Alex Hernández-García, Léna Néhale Ezzine, Yoshua Bengio, and Nikolay Malkin. A theory of continuous generative flow networks. 2023. Publisher: arXiv Version Number: 2.

[20] Alessandro Laio and Francesco L. Gervasio. Metadynamics: a method to simulate rare events and reconstruct the free energy in biophysics, chemistry and material science. *Reports on Progress in Physics*, 71(12):126601, November 2008.

[21] Alessandro Laio and Michele Parrinello. Escaping free-energy minima. *Proceedings of the National Academy of Sciences*, 99(20):12562–12566, October 2002. Publisher: Proceedings of the National Academy of Sciences.

[22] Shuang Luo, Yinchuan Li, Shunyu Liu, Xu Zhang, Yunfeng Shao, and Chao Wu. Multi-agent Continuous Control with Generative Flow Networks. *Neural Networks*, 174:106243, June 2024.

[23] Kanika Madan, Jarrid Rector-Brooks, Maksym Korablyov, Emmanuel Bengio, Moksh Jain, Andrei Nica, Tom Bosc, Yoshua Bengio, and Nikolay Malkin. Learning GFlowNets from partial episodes for improved convergence and stability. arXiv, 2022. Version Number: 3.

[24] Nikolay Malkin, Moksh Jain, Emmanuel Bengio, Chen Sun, and Y. Bengio. Trajectory Balance: Improved Credit Assignment in GFlowNets. *ArXiv*, January 2022.

[25] Nikolay Malkin, Salem Lahlou, Tristan Deleu, Xu Ji, Edward Hu, Katie Everett, Dinghuai Zhang, and Yoshua Bengio. GFlowNets and variational inference, March 2023. arXiv:2210.00580 [cs, stat].

[26] Michael Neunert, Abbas Abdolmaleki, Markus Wulfmeier, Thomas Lampe, Tobias Springenberg, Roland Hafner, Francesco Romano, Jonas Buchli, Nicolas Heess, and Martin Riedmiller. Continuous-Discrete Reinforcement Learning for Hybrid Control in Robotics. In *Proceedings of the Conference on Robot Learning*, pages 735–751. PMLR, May 2020. ISSN: 2640-3498.

[27] Esa Nummelin. *General Irreducible Markov Chains and Non-Negative Operators*. Cambridge Tracts in Mathematics. Cambridge University Press, Cambridge, 1984.

[28] Ling Pan, Dinghuai Zhang, Aaron Courville, Longbo Huang, and Yoshua Bengio. Generative Augmented Flow Networks, October 2022. arXiv:2210.03308 [cs].

[29] Grigorios A. Pavliotis. *Stochastic Processes and Applications: Diffusion Processes, the Fokker-Planck and Langevin Equations*, volume 60 of *Texts in Applied Mathematics*. Springer, New York, NY, 2014.

[30] Jarrid Rector-Brooks, Kanika Madan, Moksh Jain, Maksym Korablyov, Cheng-Hao Liu, Sarath Chandar, Nikolay Malkin, and Yoshua Bengio. Thompson sampling for improved exploration in GFlowNets, June 2023. arXiv:2306.17693 [cs].

[31] Romelia Salomon-Ferrer, David A. Case, and Ross C. Walker. An overview of the Amber biomolecular simulation package. *WIREs Computational Molecular Science*, 3(2):198–210, 2013. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/wcms.1121.

[32] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P. Adams, and Nando De Freitas. Taking the Human Out of the Loop: A Review of Bayesian Optimization. *Proceedings of the IEEE*, 104(1):148–175, January 2016.

[33] Max W. Shen, Emmanuel Bengio, Ehsan Hajiramezanali, Andreas Loukas, Kyunghyun Cho, and Tommaso Biancalani. Towards Understanding and Improving GFlowNet Training. 2023. Publisher: arXiv Version Number: 1.

[34] Tony Shen, Mohit Pandey, Jason Smith, Artem Cherkasov, and Martin Ester. TacoGFN: Target Conditioned GFlowNet for Structure-Based Drug Design, December 2023. arXiv:2310.03223 [cs].

[35] Hythem Sidky, Wei Chen, and Andrew L. Ferguson. Machine learning for collective variable discovery and enhanced sampling in biomolecular simulation. *Molecular Physics*, 118(5), March 2020. Institution: Argonne National Laboratory (ANL), Argonne, IL (United States) Publisher: Taylor & Francis.

[36] Laura Painton Swiler, Patricia Diane Hough, Peter Qian, Xu Xu, Curtis B. Storlie, and Herbert K. H. Lee. Surrogate models for mixed discrete-continuous variables. Technical Report SAND2012-0491, Sandia National Laboratories (SNL), Albuquerque, NM, and Livermore, CA (United States), August 2012.

[37] Alexandra Volokhova, Michał Koziarski, Alex Hernández-García, Cheng-Hao Liu, Santiago Miret, Pablo Lemos, Luca Thiede, Zichao Yan, Alán Aspuru-Guzik, and Yoshua Bengio. Towards equilibrium molecular conformation generation with GFlowNets, October 2023. arXiv:2310.14782 [cs].

[38] Yi Isaac Yang, Qiang Shao, Jun Zhang, Lijiang Yang, and Yi Qin Gao. Enhanced sampling in molecular dynamics. *The Journal of Chemical Physics*, 151(7):070902, August 2019.

[39] David W. Zhang, Corrado Rainone, Markus Peschl, and Roberto Bondesan. Robust Scheduling with GFlowNets, February 2023. arXiv:2302.05446 [cs].

[40] Dinghuai Zhang, Nikolay Malkin, Zhen Liu, Alexandra Volokhova, Aaron Courville, and Yoshua Bengio. Generative Flow Networks for Discrete Probabilistic Modeling. In *Proceedings of the 39th International Conference on Machine Learning*, pages 26412–26428. PMLR, June 2022. ISSN: 2640-3498.

A. Loss functions

For a complete trajectory τ, the *detailed balanced loss* (DB) is

$$L_{DB}(\tau) = \sum_{t=0}^{n-1} \left(\log \frac{\hat{f}(s_t;\theta)\hat{p}_F(s_t, s_{t+1};\theta)}{\hat{f}(s_{t+1};\theta)\hat{p}_B(s_{t+1}, s_t;\theta)} \right)^2,$$

where $\hat{f}(s_{t+1};\theta)$ is replaced with $r(s_n)$ if s_n is terminal.

The *subtrajectory balance loss* (STB) is

$$L_{STB}(\tau) = \frac{\sum_{0\le i<j\le n} \lambda^{j-1}\mathcal{L}_{TB}(\tau_{i:j})}{\sum_{0\le i<j\le n} \lambda^{j-i}},$$

$$\mathcal{L}_{STB}(\tau_{i:j}) := \left(\log \frac{\hat{f}(s_i;\theta)\prod_{t=i}^{j-1}\hat{p}_F(s_{t+1}|s_t;\theta)}{\hat{f}(s_j;\theta)\prod_{t=i+1}^{j}\hat{p}_B(s_{t-1}|s_t;\theta)} \right)^2,$$

where $\hat{f}(s_j;\theta)$ is replaced with $r(s_j)$ if s_j is terminal. In the above, $\lambda < 0$ is a hyperparameter. The limit $\lambda \to 0^+$ leads to average detailed balance. The $\lambda \to \infty$ limit gives the trajectory balance objective. We use $\lambda = 0.9$ in our experiments.

B. Langevin dynamics

Langevin dynamics (LD), is defined through the Stochastic Differential Equation (SDE):

$$\mathrm{d}x = M^{-1}p\mathrm{d}t \tag{5}$$

$$\mathrm{d}p = F(x)\mathrm{d}t - \gamma p\mathrm{d}t + \sqrt{2\gamma\beta^{-1}}M^{1/2}\mathrm{d}W. \tag{6}$$

In the above, $x, p \in \mathbb{R}^D$ are vectors of instantaneous position and momenta respectively, $F : \mathbb{R}^D \to \mathbb{R}^D$ is a force function, $W(t)$ is a vector of D independent Wiener processes, M is a constant diagonal mass matrix, and $\gamma, \beta > 0$ are constant scalars which can be interpreted as a friction coefficient and inverse temperature respectively. In conventional Langevin dynamics, the force function is given by the gradient of the potential energy function, $F = -\nabla V$, where $V : \mathbb{R}^D \to \mathbb{R}$ and the dynamics are ergodic with respect to the Gibbs-Boltzmann density

$$\rho_\beta(x, p) \propto e^{-\beta H(x,p)},$$

where $H(x, p) = p^T M^{-1} p/2 + V(x)$ is the Hamiltonian. Since the Hamiltonian is separable in position and momenta terms, the marginal Gibbs-Boltzmann density is position space is simply $\rho_\beta(x) \propto e^{-\beta V(x)}$.

C. Proofs

Lemma C.1. *Let $(f_n(x))$ and $(g_n(x))$ be sequences of real functions where $\lim_{n\to\infty} f_n(x) = \infty$, $\lim_{n\to\infty} g_n(x) = \infty$ and $\lim_{n\to\infty} \frac{f_n(x)}{g_n(x)} = h(x)$. Then, for all $\epsilon > 0$, we have $\lim_{n\to\infty} \frac{f_n(x)}{g_n(x)+\epsilon} = h(x)$.*

Proof.

$$\lim_{n\to\infty} \frac{f_n(x)}{g_n(x) + \epsilon} = \lim_{n\to\infty} \frac{f_n(x)}{g_n(x)} \frac{1}{1 + \epsilon/g_n(x)}$$

and the right hand side is the product of two functions whose limit exists so, by the product rule of limits

$$\lim_{n\to\infty} \frac{f_n(x)}{g_n(x)} \lim_{n\to\infty} \frac{1}{1 + \epsilon/g_n(x)} = h(x) \cdot 1 = h(x),$$

so done. □

Lemma C.2. *Let (X_i) be a sequence of continuous random variables that take values on a bounded domain $D \subset \mathbb{R}^d$ that asymptotically approach the uniform random variable U on D, i.e. $X_i \to U$ uniformly. Further, suppose*

$$h(x) := \frac{\sum_{i=1}^\infty f(x_i)g(x, x_i)}{\sum_{i=1}^\infty g(x, x_i)}$$

exists, where $x_i \in D$ is a sample from X_i and $f(x)$ and $g(x, x')$ are analytic functions on D and $D \times D$ respectively. Then,

$$h(x) = \frac{\sum_{i=1}^\infty f(u_i)g(x, u_i)}{\sum_{i=1}^\infty g(x, u_i)},$$

where the u_i are samples from U. We make no assumption of independence of samples.

Proof. Fix a probability space $(\Omega, \mathcal{F}, P)$ on which (X_i) and U are defined. Recall that a continuous random variable X that takes values on $D \subset \mathbb{R}^d$ is a measurable function $X : \Omega \to D$ where $(D, \mathcal{B})$ is a measure space and $\mathcal{B}$ is the Borel σ-algebra on D. Let $\omega \subset \Omega$ denote an arbitrary element of the sample space. The requirement that $X_i \to U$ uniformly can be written formally as:

$$\forall \epsilon > 0, \exists N(\epsilon) \text{ s.t. } \forall i > N(\epsilon), \forall \omega \subset \Omega, |X_i(\omega) - U(\omega)| < \epsilon.$$

We prove the Lemma by showing that equality holds for all possible sequences of outcomes $\omega_1, \omega_2, \ldots$. That is, we prove:

$$\begin{aligned} &\frac{\sum_{i=1}^\infty f(X_i(\omega_i))g(x, X_i(\omega_i))}{\sum_{i=1}^\infty g(x, X_i(\omega_i))} \\ &= \frac{\sum_{i=1}^\infty f(U(\omega_i))g(x, U(\omega_i)))}{\sum_{i=1}^\infty g(x, U(\omega_i)))}. \end{aligned} \tag{7}$$

Since these are ratios of infinite series, to prove their equality it is sufficient to show that the numerator of the LHS is asymptotically equivalent to the numerator of the RHS, and that the denominator of the LHS is asymptotically equivalent to the denomiantor of the RHS. Recall that two sequences of real functions (a_n) and (b_n) are asymptotically equivalent if $\lim_{n\to\infty} \frac{a_n(x)}{b_n(x)} = c$ where c is a constant. First, we prove that this holds with

$$a_n := \sum_{i=1}^n f(X_i(\omega_i))g(x, X_i(\omega_i)) \tag{8}$$

and

$$b_n := \sum_{i=1}^n f(U(\omega_i))g(x, U(\omega_i))). \tag{9}$$

We write $\frac{a_n}{b_n}$ as

$$\frac{\sum_{i=1}^{N(\epsilon)} f(X_i(\omega_i))g(x, X_i(\omega_i)) + \sum_{i=N(\epsilon)+1}^n f(X_i(\omega_i))g(x, X_i(\omega_i))}{\sum_{i=1}^{N(\epsilon)} f(U(\omega_i))g(x, U(\omega_i))) + \sum_{i=N(\epsilon)+1}^n f(U(\omega_i))g(x, U(\omega_i)))}.$$

Dividing by $\sum_{i=N(\epsilon)+1}^n f(U(\omega_i))g(x, U(\omega_i)))$ and taking the limit $n \to \infty$ we have

$$\lim_{n\to\infty} \frac{a_n}{b_n} = \lim_{n\to\infty} \frac{\sum_{i=N(\epsilon)+1}^n f(X_i(\omega_i))g(x, X_i(\omega_i))}{\sum_{i=N(\epsilon)+1}^n f(U(\omega_i))g(x, U(\omega_i)))}.$$

Since f and g are analytic and $i > N(\epsilon)$ for all terms in the sums we have, by Taylor expansion, $f(X_i(\omega_i)) = f(U(\omega_i)) + O(\epsilon)$ and $g(x, X_i(\omega_i)) = g(x, U(\omega_i)) + O(\epsilon)$, hence

$$\begin{aligned} \lim_{n\to\infty} \frac{a_n}{b_n} &= \lim_{n\to\infty} \left(1 + \frac{nO(\epsilon)}{\sum_{i=N(\epsilon)+1}^n f(U(\omega_i))g(x, U(\omega_i)))}\right) \\ &= 1 + \lim_{n\to\infty} \frac{nO(\epsilon)}{O(n)} = 1 + O(\epsilon). \end{aligned}$$

Finally, since ϵ can be made arbitrarily small by partitioning the sum at a $N(\epsilon)$ that is sufficiently large, we conclude that $\lim_{n\to\infty} \frac{a_n}{b_n} = 1$, hence (a_n) and (b_n) as defined in (8) and (9) are asymptotically equivalent. By a similar argument, it can be shown that

$$c_n := \sum_{i=1}^{n} g(x, X_i(\omega_i))$$

and

$$d_n := \sum_{i=1}^{n} g(x, U(\omega_i)))$$

are also asymptotically equivalent. This proves (7).

□

Below, we present the proof of Theorem 3.1 that appears in the main text.

Proof. For concreteness, throughout this proof we assume that the kernel function is a Gaussian. We explain at the appropriate stage in the proof, indicated by (*), how this assumption can be relaxed.

First we take the $t \to \infty$ limit. Since the log function is continuous, the limit and log can be interchanged and we have

$$\lim_{t\to\infty} \hat{V}(x,t) = -\frac{1}{\beta'} \log\left(\lim_{t\to\infty}\left(\frac{\hat{R}(x,t)}{\hat{N}(x,t)+\epsilon}\right)+\epsilon\right).$$

Recall the uniform time discretisation $t_n = n\Delta t$ of metadynamics (Algorithm 1). Thus, we can write

$$\hat{R}(x,t_n) = \sum_{i=1}^{n} R(x_i) \exp\left(-\sum_{i=1}^{d} \frac{(z_i(x)-z(x_i))^2}{2\sigma_i'^2}\right), \tag{10}$$

$$\hat{N}(x,t_n) = \sum_{i=1}^{n} \exp\left(-\sum_{i=1}^{d} \frac{(z_i(x)-z(x_i))^2}{2\sigma_i'^2}\right). \tag{11}$$

Since the domain is bounded, we know that for fixed x, both (10) and (11) have limit at infinity, i.e. $\lim_{t\to\infty} \hat{R}(x,t) = \infty$ and $\lim_{t\to\infty} \hat{N}(x,t) = \infty$. Hence, by Lemma C.1, we have

$$\lim_{t\to\infty} \frac{\hat{R}(x,t)}{\hat{N}(x,t)+\epsilon} = \lim_{t\to\infty} \frac{\hat{R}(x,t)}{\hat{N}(x,t)},$$

provided the limit on the RHS exists. Next, we show that this limit exists by computing it explicitly. The limit can be written

$$\lim_{t\to\infty} \frac{\hat{R}(x,t)}{\hat{N}(x,t)} = \lim_{n\to\infty} \frac{\hat{R}(x,t_n)}{\hat{N}(x,t_n)}$$
$$= \lim_{n\to\infty} \frac{\sum_{i=1}^{n} R(x_i) \exp\left(-\sum_{i=1}^{d} \frac{(z_i-z(x_i))^2}{2\sigma_i'^2}\right)}{\sum_{i=1}^{n} \exp\left(-\sum_{i=1}^{d} \frac{(z_i-z(x_i))^2}{2\sigma_i'^2}\right)}.$$

Recall that metadynamics eventually leads to uniform sampling over the domain, independent of the potential. Hence, since R and $z(x)$ are analytic, by Lemma C.2 we may replace the metadynamics samples x_i with samples from a uniform distribution, denoted u_i:

$$\lim_{n\to\infty} \frac{\hat{R}(x,t_n)}{\hat{N}(x,t_n)} = \lim_{n\to\infty} \frac{\sum_{i=1}^{n} R(u_i) \exp\left(-\sum_{i=1}^{d} \frac{(z_i-z(u_i))^2}{2\sigma_i'^2}\right)}{\sum_{i=1}^{n} \exp\left(-\sum_{i=1}^{d} \frac{(z_i-z(u_i))^2}{2\sigma_i'^2}\right)}.$$

In the limit, the ratio of sums with uniform sampling becomes a ratio of integrals:

$$\lim_{n\to\infty} \frac{\hat{R}(x,t_n)}{\hat{N}(x,t_n)} = \frac{\int_D R(x') \exp\left(-\sum_{i=1}^{d} \frac{(z_i-z(x_i'))^2}{2\sigma_i'^2}\right) \mathrm{d}x'}{\int_D \exp\left(-\sum_{i=1}^{d} \frac{(z_i-z(x_i'))^2}{2\sigma_i'^2}\right) \mathrm{d}x'},$$

where $D \subset \mathbb{R}^d$ is the domain. The limit is therefore a (scaled) convolution of the reward function with a Gaussian in the collective variable space with width vector σ'. Taking the limit $\sigma_i' \to 0$ for all $i \in \{1, 2, \ldots, d\}$, the Gaussian convergences to a delta distribution in the collective variable space and we have

$$\lim_{\sigma'\to 0} \lim_{n\to\infty} \frac{\hat{R}(x,t_n)}{\hat{N}(x,t_n)} = \int_D R(x')\delta(z-z(x'))\mathrm{d}x'.$$

(*) This step also holds for any kernel that becomes distributionally equivalent to a Dirac delta function in the limit that its variance parameter goes to zero. In particular, it also holds for the von Mises distribution that we use in our alanine dipeptide experiment in $\mathbb{T}^2$.

Finally, we take the limit $\epsilon \to 0$ to obtain

$$\begin{aligned} &\lim_{\epsilon\to 0} \lim_{\sigma'\to 0} \lim_{t\to\infty} \hat{V}(x,t) \\ &= -\frac{1}{\beta'} \lim_{\epsilon\to 0} \log\left(\int_D R(x')\delta(z-z(x'))\mathrm{d}x' + \epsilon\right) \\ &= -\frac{1}{\beta'} \int_D \log\left(R(x')\right)\delta(z-z(x'))\mathrm{d}x' \\ &= \int_D V(x')\delta(z-z(x'))\mathrm{d}x' := V(z), \end{aligned} \tag{12}$$

where we have used the definition $V(x') = -\frac{1}{\beta'}\log(R(x'))$ and in the last step we used the definition of the marginal potential energy in the collective variable space. If $z(x)$ is invertible, then the delta function simplifies to a delta function in the original space and we obtain the original potential instead of the marginal potential. □

D. Algorithms

In the algorithm below, we present the variant of MetaGFN where we store Adaptive Metadynamics samples in the replay buffer and regenerate trajectories using the current

Algorithm 2 Euler-Maruyama Langevin Dynamics Step

Input: Current state (x_t, p_t). Force F.
Parameter: Friction coefficient γ. Thermodynamic beta β. Timestep Δt.
Output: State $(x_{t+\Delta t}, p_{t+\Delta t})$ at the next timestep.
Sample a random vector R, with the same dimension as x_t, where each element is an independent sample from a standard normal.
$x_{t+\Delta t} = x_t + p_t \Delta t$
$p_{t+\Delta t} = p_t + F\Delta t - \gamma p_t \Delta t + \sqrt{2\gamma \Delta t/\beta} \cdot R$
return $(x_{t+\Delta t}, p_{t+\Delta t})$

backward policy when retrieving from the replay buffer. This is the variant we used in our experiments in the main text.

Algorithm 3 MetaGFN

Input: Forward policy P_F. Backwards policy P_B. Loss function L.
Parameter: How often to run Adaptive Metadynamics batches, `freqMD`. How often to run replay buffer batches, `freqRB`. Batch size, b. Stride, $n \in \mathbb{Z}^+$. Time step, $\Delta t > 0$.
for each episode **do**
  if episode number is divisible by `freqMD` **then**
    Run Adaptive Metadynamics (batch size b) for time $n\Delta t$, obtain samples $\{x_1, \ldots, x_b\}$
    Push $\{x_1, \ldots, x_b\}$ to the replay buffer
    Backward sample from $\{x_1, \ldots, x_b\}$ using current P_B to obtain trajectories $\{\tau_1, \ldots, \tau_b\}$
  else if episode number is divisible by `freqRB` **then**
    Random sample $\{x_1, \ldots, x_b\}$ from the replay buffer
    Backward sample from $\{x_1, \ldots, x_b\}$ using current P_B to obtain trajectories $\{\tau_1, \ldots, \tau_b\}$
  else
    Generate trajectories $\{\tau_1, \ldots, \tau_b\}$ on-policy
  end if
  Compute loss $l = \sum_{i=1}^{b} L(\tau_i, P_F, P_B)$
  Take gradient step on loss l
end for

E. Experiment details: line environment

E.1. Experimental setup

Parameterisation We parameterise $\hat{p}_F$, $\hat{p}_B$ and the flow $\hat{f}$ through an MLP with 3 hidden layers, 256 hidden units per layer. We use the GELU activation function and dropout probability 0.2 after each layer. This defines the torso of the MLP. Connecting from this common torso, the MLP has three single-layer, fully-connected heads. The first two heads have output dimension 9 and parameterise the 3 means (μ), standard deviations (σ) and weights (w) of the mixture of Gaussians for the forward and backward policies respectively. The third head has output dimension 1 and parameterises the flow function $\hat{f}$. The mean and standard deviation outputs are passed through a sigmoid function and transformed so that they map to the ranges $\mu \in (-14, 14)$ and $\sigma \in (0.1, 1)$. The mixture weights are normalised with the softmax function. The exception to this parameterisation is the backward transition to the source state, which in accordance with theory, is fixed to be the Dirac delta distribution centred on the source, i.e. $\hat{p}_B(s_0|s_1;\theta) = \delta_{s_0}$. For the TB loss, we treat $\log Z_\theta$ as a separate learnable parameter.

Replay buffer The replay buffer has capacity for 10^4 trajectories. Trajectories are stored in the replay buffer only if terminal state's reward exceeds 10^{-3}. When drawing a replay buffer batch, trajectories are bias-sampled: 50% randomly drawn from the upper 30% of trajectories with the highest rewards, the remaining 50% randomly drawn from the lower 70%.

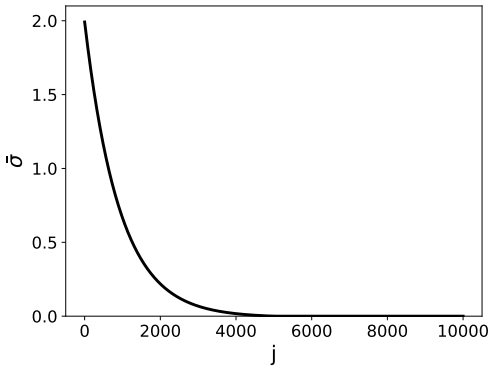

Figure 6: Exponential noise schedule.

Noisy exploration Noisy exploration is defined by adding an additional constant, $\bar{\sigma}$, to the standard deviations of the Gaussian distributions of the forward and backward policies. Specifically, the forward policy becomes $\hat{p}_F(s_t, s_{t-1};\theta) = \sum_{i=1}^{3} w_i \mathcal{N}(\mu_i, (\sigma_i + \bar{\sigma})^2)$, and similarly for the backward policy. We schedule the value of $\bar{\sigma}$ so that it decreases during training according to an exponential-flat schedule:

$$\bar{\sigma} = \begin{cases} \bar{\sigma}_0 \left(e^{-2je/(B/2)} - e^{-2e}\right) & j < B/2 \\ 0 & j \geq B/2, \end{cases} \tag{13}$$

where $j \in (1, \ldots, B)$ is the batch number and $\bar{\sigma}_0 = 2$ is the initial noise, plotted in Figure 6.

Thompson Sampling We use 10 heads with the bootstrapping probability parameter set to 0.3.

MetaGFN We use $\Delta t = 0.05$, $n = 2$, $\beta = 1$, $\gamma = 2$, $w = 0.15$, $\sigma = 0.1$, $\epsilon = 10^{-3}$. The domain of Adaptive

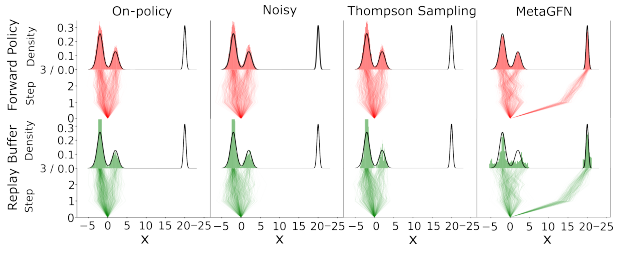

Figure 7: Forward policy and replay buffer distributions after training for 10^5 iterations with TB loss. MetaGFN is the only method that is able to consistently learn all three peaks.

Metadynamics is restricted to $[-5, 23]$ and reflection conditions are imposed at the boundary. The bias and KDE potentials are stored on a uniform grid with grid spacing 0.01. Initial metadynamics samples are drawn from a Gaussian distribution, mean 0 and variance 1, and initial momenta from a Gaussian distribution, mean 0 and variance 0.5.

Training parameters In all experiments, we use batch size $b = 64$ and for $B = 10^5$ batches. We use a learning rate with a linear schedule, starting at 10^{-3} and finishing at 0. For the TB loss we train the $\log Z_\theta$ with a higher initial learning rate of 10^{-1} (also linearly scheduled). For the STB loss we use $\lambda = 0.9$, a value that has worked well in the discrete setting (23). We use the Adam optimiser with gradient clipping. For all loss functions, we clip the minimum log-reward signal at -10. This enables the model to learn despite regions of near-zero reward between the modes of $r(x)$.

Evaluation The L1 error between the known reward distribution, $\rho(x) = r(x)/Z$, and the empirical on-policy distribution, denoted $\hat{\rho}(x)$, estimated by sampling 10^4 on-policy trajectories and computing the empirical distribution over terminal states. Specifically, we compute

$$\text{error} = \frac{1}{2}\int_{-5}^{23} \left| \hat{\rho}(x) - \frac{r(x)}{Z} \right| \mathrm{d}x, \tag{14}$$

where the integral is estimated by a discrete sum with grid spacing 0.01. Note that this error is normalised such that for all valid probability distributions $\hat{\rho}(x)$, we have $0 \leq \text{error} \leq 1$.

Compute resources Experiments are performed in PyTorch a desktop with 32Gb of RAM and a 12-core 2.60GHz i7-10750H CPU. It takes approximately 1 hour to train a continuous GFlowNet in this environment with $B = 10^5$ batches. The additional computational expense of running Adaptive Metadynamics was negligible compared to the training time of the models.

E.2. Results

On-policy distributions Figure 7 shows the forward policy and replay buffer distributions (with bias sampling) after training for 10^5 iterations with TB loss. MetaGFN is the only method that is able to uniformly populate the replay buffer and consistently learn all three peaks.

Adaptive Metadynamics Figure 9 shows the L1 error between the density implied by the kde potential and the true reward distribution during a typical training run. Figure 10 shows the resulting $\hat{V}$ and V_{bias} at the end of the training. By (1), Adaptive Metadynamics has fully-explored the central peaks. At (2), the third peak is discovered, prompting rapid adjustment of the KDE potential. By 2.5×10^4 iterations, steady state is reached and the algorithm is sampling the domain uniformly.

Comparing MetaGFN Variants We consider three MetaGFN variants. The first variant, *always backwards sample*, regenerates the entire trajectory using the current backward policy when pulling from the replay buffer. The second variant, *reuse initial backwards sample*, generates the trajectory when first added to the replay buffer and reuses the entire trajectory if subsequently sampled. The third variant, *with noise*, is always backwards sample with noisy exploration as per equation (13). We plot the L1 policy errors in Figure 8. We observe that *always backward sample* is better than *reuse initial backwards sample* for all loss functions. For DB and TB losses, there is no evidence for any benefit of adding noise, whereas noise improves training for STB loss, performing very similarly to TB loss without noise.

F. Experiment details: alanine dipeptide environment

Computing the Free Energy Surface To obtained a ground-truth free energy surface (FES) in ϕ-ψ space, we ran a 250ns NPT well-tempered metadynamics MD simulation of alanine dipeptide at temperature 300K ($\beta = 0.4009$), pressure 1bar with the TIP3P explicit water model (17). We used the PLUMMED plugin (4) for OpenMM (10) with the AMBER14 force field (31).

Parameterisation We parameterise $\hat{p}_F$, $\hat{p}_B$ and $\hat{f}$ through three heads of an MLP with 3 hidden layers with 512 hidden units per layer, with GeLU activations and dropout probability 0.2, similar to the Line Environment. The first two heads have output dimension 15, parameterising the 6 means, 6 concentrations and 3 weights of the mixture of von Mises policy. The third head has output dimension 1 and parameterises the flow function $\hat{f}$. The means are mapped to the range $(-\pi, \pi)$ through $2\arctan(\cdot)$. Concentrations are parameterised in log-space and are passed through a sigmoid so that they map to the range $\ln(\kappa) \in (0, 5)$. Mixture

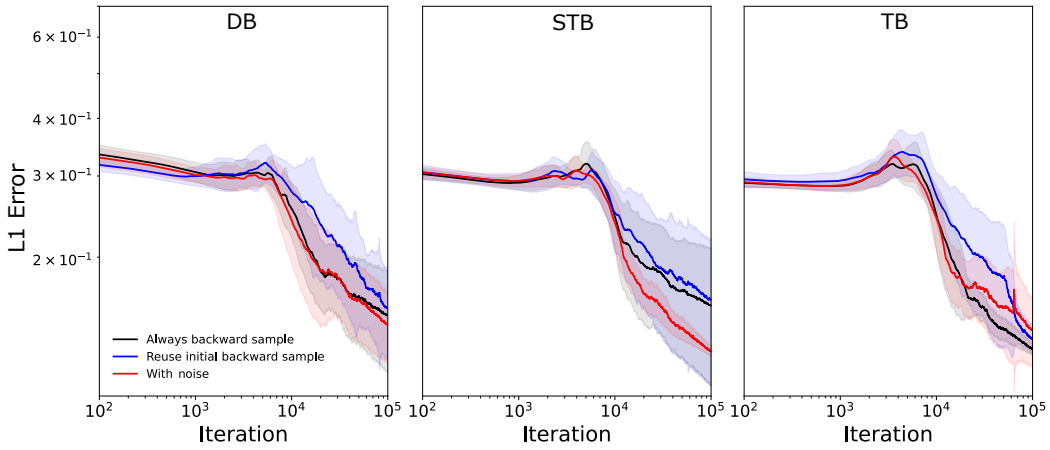

Figure 8: Comparing MetaGFN variants.

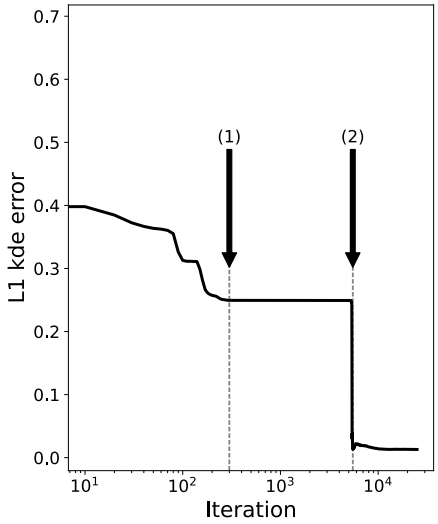

Figure 9: L1 error between $\hat{\rho} = \exp(-\beta \hat{V}(x))/Z$ and the reward distribution, $r(x)/Z$.

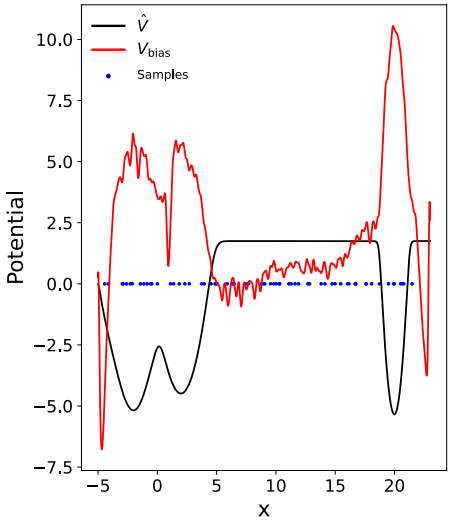

Figure 10: KDE potential, bias potential, and positions of final samples after 2.5×10^4 training iterations.

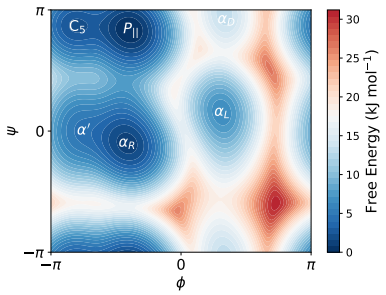

Figure 11: The KDE potential $\hat{V}$ learnt using Adapted Metadynamics with von Mises kernel.

weights are normalised with the softmax function.

Replay buffer The replay buffer has capacity for 10^4 trajectories. Trajectories are stored in the replay buffer only if terminal state's reward exceeds 10^{-10}. When drawing a replay buffer batch, trajectories are bias-sampled: 50% randomly drawn from the upper 30% of trajectories with the highest rewards, the remaining 50% randomly drawn from the lower 70%.

Noisy exploration We use the same noise profile as the Line Environment, equation (13). The noise $\bar{\sigma}$ is now added to the concentration parameter κ. Concentration is related to standard deviation through $\sigma = \frac{1}{\kappa^2}$.

Thompson Sampling We use 10 heads with the bootstrap probability parameter set to 0.3.

MetaGFN We use $\texttt{freqRB} = 2$, $\texttt{freqMD} = 10$, $\Delta t = 0.01$, $n = 2$, $\beta = 0.4009$, $\gamma = 0.1$, $w = 10^{-5}$, $\kappa = 10$, $\epsilon = 10^{-6}$. The bias and KDE potentials are stored on a uniform grid with grid spacing 0.1. Initial samples are drawn from a Gaussian distribution, mean centered $P_{||}$, variance $\sigma^2 = (0.1, 0.1)$, and initial momenta from a Gaussian, mean $\mu = (0, 0)$, variance $\sigma^2 = (0.05, 0.05)$. The resulting KDE potential learnt during Adapted Metadynamics is shown in Figure 11.

Training parameters The same as for the Line Environment, see Appendix E.1.

Evaluation The L1 error of a histogram of on-policy samples, $\hat{\rho}(\phi, \psi)$, is computed via a two-dimensional generalisation of (14); error $= \frac{1}{2}\int_{-\pi}^{\pi}\int_{-\pi}^{\pi}\left|\hat{\rho}(\phi,\psi) - \frac{r(\phi,\psi)}{Z}\right| \mathrm{d}\phi\mathrm{d}\psi$, estimated by a discrete sum with grid spacing 0.1.

Compute resources Experiments are performed in PyTorch a desktop with 32Gb of RAM and a 12-core 2.60GHz i7-10750H CPU. It takes approximately 10 hours to train a continuous GFlowNet in this environment with $B = 10^5$ batches. The additional computational expense of running Adaptive Metadynamics was less than 10%.