# Stop the Sampler! Classifier-Based Adaptive Stopping for Sampling Kernels

**Kirill Korolev**
HSE University
kkorolev@hse.ru

**Nikita Morozov**
HSE University
nvmorozov@hse.ru

**Stepan Pavlenko**
HSE University
spavlenko@hse.ru

**Esmeralda S. Whitammer**
University of Edinburgh & CIFAR
esmeralda.whitammer@ed.ac.uk

**Sergey Samsonov**
HSE University
svsamsonov@hse.ru

## Abstract

Sampling from complex, unnormalized probability densities is a fundamental challenge in Bayesian inference and probabilistic modeling. While Markov chain Monte Carlo (MCMC) methods provide asymptotic guarantees, they often suffer from slow mixing and high computational costs due to fixed or manually tuned trajectory lengths. In this work, we propose a novel framework that treats trajectory termination as a learnable component of the sampling dynamics. By framing MCMC within the theory of non-acyclic generative flow networks (GFlowNets), we train state-dependent neural classifiers to decide when a trajectory has reached a high-density region and should terminate. We theoretically establish the connection between optimal classifiers and the target density via detailed balance conditions and introduce a multilevel training scheme to facilitate exploration in complex geometries. Experimental results across various benchmark densities demonstrate that our approach significantly reduces average trajectory lengths while improving mode coverage and mixing compared to standard MCMC baselines.

## 1 Introduction

In this paper, we consider the problem of sampling from a probability distribution given by its unnormalized density:

$$\pi(x) = r(x)/Z, \quad Z = \int_{\mathbb{R}^d} r(x)\,dx \tag{1}$$

where $r : \mathbb{R}^d \to \mathbb{R}^+$ is a positive function and $Z$ is an unknown normalizing constant. The problem arises in many domains, including Bayesian statistics, probabilistic deep learning, and natural sciences [23, 37, 44, 32, 18]. The main two approaches to solve the sampling problem (1) are either to perform (asymptotically) exact sampling from $\pi$ based on Monte Carlo or MCMC algorithms [30, 31, 17], or to bypass exact sampling in favor of *amortized* solutions to this problem. MCMC algorithms operate by constructing an ergodic Markov chain (see, e.g., [12]) $\{Y_n\}_{n\in\mathbb{N}}$ with invariant distribution equal to $\pi$, such that the distribution of $n$-th iterate $Y_n$ becomes close to $\pi$ regardless of the starting distribution of $Y_0$. MCMC-based approaches are known to suffer from slow convergence to the target distribution [2, 31], especially when the target dimension $d$ is large. In addition to slow convergence, the important question related to this family of methods is whether the length of the generated trajectory is sufficient for the marginal distribution of the last element to be close to $\pi$. Known approaches to this problem consider some statistics based on generated samples [34, 45], which do not always provide affirmative stopping criteria.

Contrary to exact sampling approaches, amortized solutions to (1) train a generative model to approximately sample from $\pi$. Contrary to MCMC, they shift the computational burden to the

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

training phase, enabling faster inference at the price of bypassing the exact sampling. Models of this class include normalizing flows [32, 27, 13], diffusion samplers [47, 40, 7], and generative flow networks (GFlowNets) [3, 4, 22]. While GFlowNets were initially developed for discrete domains, they were further extended to continuous problems in [22] and offer another family of training algorithms for diffusion samplers [46, 36, 19, 15, 5]. GFlowNets operate by training a stochastic policy to approximate the target distribution over terminal states in an appropriately constructed Markov decision process (MDP), see, e.g., [38]. Recent theoretical developments in GFlowNets [22, 10, 8] present a broad and convenient theoretical framework for developing sampling algorithms.

In this work, we utilize GFlowNet methodology to train neural network classifiers for early stopping of MCMC kernels, creating a new family of amortized samplers based on algorithms from both worlds. Our key idea is to treat stopping as a learnable component of the sampling dynamics: instead of running a Markov chain for a fixed or externally determined number of steps, we learn state-dependent classifiers that decide when a trajectory should terminate. This leads to a unified framework where sampling trajectories are dynamically shortened in high-density regions while remaining exploratory elsewhere. Our main contributions are:

1. We theoretically show how sampling dynamics in $\mathbb{R}^d$ with an addition of a possibility to stop in any point $x$ can be viewed as a non-acyclic GFlowNet environment [8], establish connections between the learned stopping policies and the target density via flow and detailed balance conditions, and derive a characterization of optimal classifiers in terms of underlying MCMC dynamics.
2. Building on this foundation and prior non-acyclic GFlowNet training methodology [8, 29], we introduce practical algorithms that combine learned stopping with adaptive forward and backward kernels, as well as a multilevel extension that improves training and exploration.
3. We provide experimental evaluation on a number of density functions, demonstrating that our approach yields substantially shorter trajectories compared to standard MCMC while better capturing the geometry of complex target distributions and improving mixing across modes through learned drift corrections.

Source code: github.com/kkorolev1/stop-the-sampler.

## 2 Background

### 2.1 GFlowNets on general state spaces

We briefly describe the setting of the sampling problem on the general state space. Given a measurable space $(\mathcal{X}, \Sigma_{\mathcal{X}})$, and a finite reward measure $R$ on this space, we aim to generate samples from the distribution $\pi$ defined by

$$\pi(A) = R(A)/R(\mathcal{X}), \quad A \in \Sigma_{\mathcal{X}}\,.$$

To formulate this sampling problem on general state spaces, [22] extend GFlowNets by introducing *measurable pointed graphs*, a measure-theoretic generalization of the directed acyclic graphs used in discrete GFlowNets. The underlying state space is $\bar{\mathcal{S}} = \{s_0\} \cup \mathcal{S} \cup \{s_f\}$, equipped with its $\sigma$-algebra $\Sigma$. A trajectory starts at the source $s_0$, visits internal states in $\mathcal{S}$, and ends upon reaching the absorbing sink $s_f$. The supports of the forward and backward reference kernels $\kappa_F(s, \cdot)$ and $\kappa_B(s, \cdot)$ generalize the notions of children and parents in a discrete DAG, respectively, defining the set of states reachable from $s$ (resp. from which $s$ is reachable) in one step. The reference kernel is used to define policy transition densities and does not itself serve as a sampling policy. The terminal set $\mathcal{X}$ consists of all internal states from which the graph permits a direct transition to the sink $s_f$.

[22, Eq. (7)] assume that the measurable pointed graph is *finitely absorbing*. Specifically, there exists a finite $N \in \mathbb{N}$, such that $\mathrm{supp}(\kappa_F^N(s_0, \cdot)) = \{s_f\}$. This assumption is the analogue of acyclicity in a finite directed graph. In this work we utilize the *non-acyclic GFlowNets* formulation and replace the finitely absorbing condition with the weaker assumption of finite expected trajectory length (Assumption 3.1) and discuss it in §3.1.

**Trajectory distribution and sampling problem.** Suppose that we are given a *forward* Markov kernel $P_F$ on $(\bar{\mathcal{S}}, \Sigma)$ such that $P_F(s, \cdot) \ll \kappa_F(s, \cdot)$. The forward kernel $P_F$ therefore admits a transition density $p_F$ with respect to the reference kernel $\kappa_F$, satisfying

$$P_F(s, A) = \int_A p_F(s' \mid s)\, \kappa_F(s, \mathrm{d}s'), \qquad A \in \Sigma.$$

Although $\kappa_F$ need not be a Markov kernel, the convention $\kappa_F(x, \{s_f\}) = 1$ for $x \in \mathcal{X}$ [22, Eq. (6)] ensures that $p_F(s_f \mid x)$ equals the probability of terminating at $x$. For each fixed $n \geq 1$, let

$\tau = (s_0, s_1, \ldots, s_n = x, s_{n+1} = s_f)$ be a complete trajectory of length $n$ that terminates at $s_n = x \in \mathcal{X}$. Trajectory density with respect to the path measure induced by $\kappa_F$ is $p_F(\tau) = \prod_{t=1}^{n+1} p_F(s_t \mid s_{t-1})$. Marginalizing the trajectory distribution over the states preceding $s_n$ yields the terminating distribution $P_T$ on $\mathcal{X}$. The sampling objective is to construct a forward kernel $P_F$ such that

$$P_T(A) = \pi(A), \qquad A \in \Sigma_{\mathcal{X}}. \tag{2}$$

The following balance relations provide sufficient conditions for achieving the sampling objective (2).

**Local balance conditions.** As in the discrete case, the construction also includes a flow measure $F$, where $F(A)$ is proportional to the expected number of visits to a measurable set $A$ along a complete trajectory generated by $P_F$. Flow and reward densities are defined with respect to a reference measure $\nu$, typically the counting measure on a discrete space or the Lebesgue measure on a Euclidean space. Let $F \ll \nu$ be a $\sigma$-finite measure on $(\bar{\mathcal{S}}, \Sigma)$ with density $f = \mathrm{d}F/\mathrm{d}\nu$, and let $P_F$ be a forward Markov kernel with transition density $p_F(\cdot \mid s)$ with respect to $\kappa_F(s, \cdot)$ for each state $s$. The pair $(f, p_F)$ satisfies the *flow-matching condition* if:

$$f(s') = \int_{\bar{\mathcal{S}} \setminus \{s_f\}} f(s)\, p_F(s' \mid s)\, \kappa_B(s', \mathrm{d}s), \qquad \text{for } \nu\text{-almost every } s' \neq s_0. \tag{3}$$

Assume $R \ll \nu$ and denote the reward density by $r = \mathrm{d}R/\mathrm{d}\nu$. The tuple $(f, p_F)$ satisfies the *reward-matching conditions* if:

$$f(x)\, p_F(s_f \mid x) = r(x), \qquad \text{for } \nu\text{-almost every } x \in \mathcal{X}. \tag{4}$$

Specifically, [22] showed that when the flow-matching and reward-matching conditions are satisfied, the terminating distribution $P_{\mathrm{T}}$ induced by $P_F$ is proportional to $R$, and (2) holds.

**Detailed balance.** Let $P_B$ be a *backward* Markov kernel on $(\bar{\mathcal{S}}, \Sigma)$ that assigns to each state $s' \neq s_0$ a distribution over its predecessors. Assume that $P_B(s', \cdot) \ll \kappa_B(s', \cdot)$ for each such $s'$, and denote the corresponding transition density by $p_B(\cdot \mid s')$. The triple $(f, p_F, p_B)$ satisfies *detailed balance* if

$$f(s)\, p_F(s' \mid s) = f(s')\, p_B(s \mid s') \tag{5}$$

for $\nu(\mathrm{d}s)\, \kappa_F(s, \mathrm{d}s')$-almost every pair $(s, s')$ with $s \neq s_f$ and $s' \neq s_0$. As shown by [22, Proposition 1], detailed balance implies that $(f, p_F)$ satisfies the flow-matching condition (3).

**Trajectory balance.** The triple $(Z, p_F, p_B)$ satisfies *trajectory balance* if, for every $n \geq 1$,

$$\prod_{t=1}^{n+1} p_F(s_t \mid s_{t-1}) = \frac{r(s_n)}{Z} \prod_{t=0}^{n-1} p_B(s_t \mid s_{t+1}) \tag{6}$$

for almost every complete trajectory $\tau$ of length $n$. Under this condition, the flow induced by $p_F$ satisfies both flow-matching (3) and reward-matching (4) [22, Proposition 2]. A detailed presentation of GFlowNets on general state spaces is provided in §A.

### 2.2 Generalizations to non-acyclic setting

The theory of non-acyclic GFlowNets was developed in [8], extending the standard framework to settings where the graph may contain cycles. In this case, additional considerations arise due to the presence of cycles, which may lead to unnecessarily long trajectories. It was shown that the expected trajectory length is controlled by the total flow through non-terminal states, yielding the bound

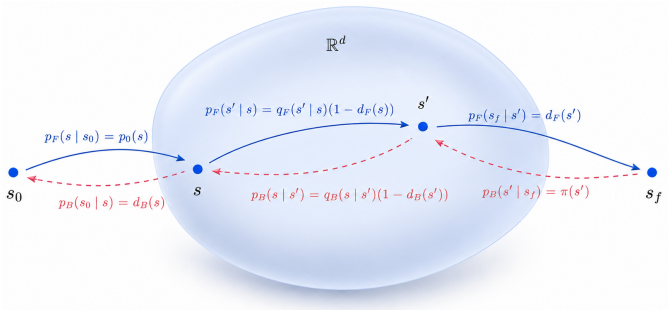
$$\mathbb{E}_{\tau \sim P}[n_\tau] \leq F(\mathcal{S})/F(\{s_0\}). \tag{7}$$

Subsequently, [28] refined this result in discrete spaces by showing that, when $F$ corresponds to the occupancy measure induced by the backward process, the inequality becomes an equality. This highlights that minimizing the total flow can be an effective strategy for reducing expected trajectory lengths in non-acyclic GFlowNets, providing the motivation for using the flow as a training regularizer.

### 2.3 Related work on adaptive kernels and early-stopping in MCMC

Another line of work in Monte Carlo studies stopping rules for Markov chains, i.e., criteria for deciding whether a partial trajectory is sufficient for estimation. These rules are typically based on convergence diagnostics or precision criteria computed from the generated samples [34, 45]. For

Figure 1: Structured state space for the non-acyclic continuous GFlowNet. The generation process starts from the source state $s_0$, enters the continuous space $\mathbb{R}^d$ according to $p_0$, evolves between continuous states using $q_F(s' \mid s)(1 - d_F(s))$, and terminates at $s_f$ with probability $d_F(s)$. Backward transitions are defined analogously in the reverse direction.




example, [21] compare Gelman–Rubin, Geweke diagnostics, and effective sample size, showing that aggressive thresholds may lead to premature termination and biased estimates. Thus, stopping in MCMC is usually an online diagnostic applied to the generated trajectory. We provide the corresponding formulas for these diagnostic approaches in §C.

A complementary direction focuses on adaptive kernels, where the transition mechanism is adjusted according to local geometry or performance. In adaptive MCMC [16, 1, 41, 39], proposal parameters are optimized within families such as Gaussian random-walk or MALA. Extensions include position-dependent preconditioning and state-dependent step sizes [35, 6, 24]. More recent work formulates proposal adaptation as a reinforcement learning problem [42, 43].

## 3 Non-acyclic GFlowNets with adaptive sampling time

We introduce a learnable stopping rule into the sampling dynamics. Starting from an initial distribution $p_0$, the process repeatedly either applies a transition kernel $q_F$ or terminates according to a state-dependent classifier $d_F$. Learning this stopping rule allows us to control trajectory length and, as shown in Theorem 3.6, can also correct the terminal distribution of a fixed kernel that does not preserve the target.

### 3.1 Classifier-based adaptive stopping

**State space and transition dynamics.** In the context of continuous acyclic GFlowNets, it is common to encode the state space $\mathcal{S}$ as pairs $(s, n)$, where $s \in \mathbb{R}^d$ and $n$ indicates a discrete timestep along a trajectory [22]. In contrast, in this work we remove the notion of time, moving instead to the non-acyclic GFlowNet setting.

We define the state space as $\bar{\mathcal{S}} = \{s_0\} \cup \mathcal{S} \cup \{s_f\}$, where $s_0$ and $s_f$ denote the source and sink states, respectively, and $\mathcal{S} = \mathbb{R}^d$. Forward transitions are allowed only from $s_0$ to $\mathcal{S}$ and from $\mathcal{S}$ to $\mathcal{S} \cup \{s_f\}$, and $s_f$ is absorbing. Backward transitions are defined analogously in the reverse direction. Following [22], we model the forward and backward policies as Markov kernels on $\bar{\mathcal{S}}$. To work with densities on this mixed discrete–continuous state space, we equip $\bar{\mathcal{S}}$ with the reference measure $\nu = \lambda + \delta_{s_0} + \delta_{s_f}$, where $\lambda$ is the Lebesgue measure on $\mathbb{R}^d$ and $\delta_s$ is the Dirac measure at $s$. Throughout, we assume that the relevant measures are absolutely continuous with respect to the reference measure $\nu$.

To handle trajectory termination without fixed length, we introduce forward and backward classifiers, $d_F, d_B : \mathbb{R}^d \to [0, 1]$, which map a state $s \in \mathbb{R}^d$ to the probability of stopping the trajectory. The forward classifier $d_F(s)$ determines the probability of transitioning directly to the sink state $s_f$, while the backward classifier $d_B(s)$ controls the probability of moving to the source state $s_0$. The forward transition density on $\bar{\mathcal{S}}$ is defined by

$$p_F(s \mid s_0) = p_0(s), \qquad p_F(s' \mid s) = (1 - d_F(s))q_F(s' \mid s), \qquad p_F(s_f \mid s) = d_F(s), \tag{8}$$

where $p_0$ is a fixed initial distribution and $q_F$ is a transition density on $\mathbb{R}^d$. The backward transition density $p_B$ on $\bar{\mathcal{S}}$ is defined analogously in the reverse direction using transition density $q_B$ on $\mathbb{R}^d$ and a classifier $d_B$, with the sink transition fixed by reward-matching:

$$p_B(s \mid s_f) = \pi(s), \qquad p_B(s \mid s') = (1 - d_B(s'))q_B(s \mid s'), \qquad p_B(s_0 \mid s) = d_B(s). \tag{9}$$

Figure 1 illustrates the proposed non-acyclic state space. A complete trajectory is a sequence $\tau = (s_0 \to s_1 \to \ldots \to s_{n_\tau} \to s_{n_\tau+1} = s_f)$ is called a trajectory of length $n_\tau \in \mathbb{N}$ if $s_t \in$

$\mathbb{R}^d, t \in \{1, \ldots, n_\tau\}$ and $\mathcal{T}$ denotes the set of such complete trajectories. To ensure that the forward process defines a probability distribution over complete trajectories and induces a finite flow measure, terminating in a finite expected number of steps, we impose the following assumption.

**Assumption 3.1.** Let $P$ be the trajectory distribution induced by the forward kernel $P_F$, then we assume $\mathbb{E}_{\tau\sim P}[n_\tau] < \infty$.

Assumption 3.1 guarantees almost-sure termination and a finite expected number of sampling steps. It is weaker than the finite-absorption condition of [22], which imposes a uniform deterministic bound on trajectory length. In our setup trajectories may be arbitrarily long, provided that their expected length is finite. This requirement is consistent with earlier non-acyclic GFlowNet formulations [8, 28, 29]. In §B.1 we argue why it holds in our case in practice.

**Flow measure and reward-matching.** To formulate balance conditions that guarantee sampling from $\pi$, we need to characterize a flow measure, a standard object in GFlowNet theory [4, 22]. In contrast to [22], where the flow is defined as a measure satisfying the flow-matching condition, we define it directly in terms of the forward transition kernel $P_F$. This is complementary to the approach of [28], where the flow is constructed via a backward transition kernel $P_B$.

**Definition 3.2** (Flow measure)**.** Under Assumption 3.1, the flow measure $F$ is defined as

$$F(\{s_0\}) = F(\{s_f\}) = Z, \qquad F(A) = F(\{s_0\}) \sum_{n=0}^{\infty} P_F^n(s_0, A), \quad A \in \Sigma_{|\mathcal{S}}. \tag{10}$$

Intuitively, for a measurable set $A \subseteq \mathbb{R}^d$, the ratio $F(A)/F(\{s_0\})$ is the expected number of visits to $A$ along a complete trajectory generated by the forward kernel $P_F$. Consequently, as in the discrete non-acyclic case [28], this definition of a flow measure yields flow-matching.

**Proposition 3.3.** *The flow measure $F$ from Def. 3.2 and the Markov kernel $P_F$ satisfy the flow-matching conditions (3).*

The proof is in §E.2. Note that our structured state space is a particular instance of the measurable non-acyclic setting in [8]. Hence, their general sampling result applies directly in our case.

**Theorem 3.4.** *If a flow measure $F$ and a Markov kernel $P_F$ satisfy both flow-matching and reward-matching conditions, then the sampling distribution $P_T(A) = \pi(A)$.*

In contrast to [8], where the flow is arbitrary and yields only an upper bound on the expected trajectory length, our flow is defined as the occupancy measure induced by $P_F$, and the bound becomes tight.

**Proposition 3.5.** *Let $P$ be the trajectory distribution induced by the forward kernel $P_F$. Let $F$ be the flow measure (Def. 3.2). Then the expected trajectory length is a constant multiple of the total flow:*

$$\mathbb{E}_{\tau\sim P}[n_\tau] = F(\mathcal{S})/F(\{s_0\}). \tag{11}$$

The proof can be found in §E.3. We also show that detailed balance implies the following relations:

$$f(s) = \frac{r(s)}{d_F(s)} = \frac{Z\, p_0(s)}{d_B(s)}, \qquad d_F(s) = \frac{\pi(s)}{p_0(s)} d_B(s), \tag{12}$$

These identities allow us to parameterize the flow density through a stopping classifier, avoiding a separate learnable model for $f$. The derivation can be found int §E.1.

### 3.2 Connection with MCMC

In this section, we relate the generation kernel restricted to transitions within $\mathcal{S}$ to early stopping of time-homogeneous chains that use this kernel as the their transition operator, which establishes a precise link between MCMC and our framework.

Let $Q_F$ be a fixed forward kernel with a transition density $q_F$. Assume that $Q_F$ has full support on $\mathbb{R}^d$ and is uniformly geometrically ergodic. That is, $Q_F$ admits a unique stationary measure $\pi_Q$, and there exist constants $C < \infty$ and $\rho \in (0, 1)$ such that

$$\|\nu Q_F^n - \pi_Q\|_{\mathrm{TV}} \leq C\rho^n \tag{13}$$

for all $n \in \mathbb{N}$ and any probability measure $\nu$, where $\|\cdot\|_{\mathrm{TV}}$ denotes the total variation norm. Let $\pi(s) = r(s)/Z$ be a target distribution, $p_0$ be the initial distribution of a Markov chain, and $P$ be the trajectory distribution induced by $p_F = (q_F, d_F)$. By a slight abuse of notation, we use $p_0, \pi$ and $\pi_Q$ to denote both the corresponding probability measures and their densities, depending on the context.

**Theorem 3.6.** *Assume $Q_F$ is a kernel with a density $q_F$ satisfying (13). The tuple $(f, q_F, d_F, q_B, d_B)$ is a solution to detailed balance conditions (5) if and only if the following two statements hold:*

*1. The flow measure $F$ satisfies for any measurable $A \subseteq \mathbb{R}^d$*

$$F(A) = Z\big(U(A) + n_Q \pi_Q(A)\big), \tag{14}$$

*where*

$$U(A) = \sum_{n=0}^{\infty} \big(p_0 Q_F^n - \pi Q_F^{n+1}\big)(A), \qquad n_Q \geq n_Q^* := \sup_{s \in \mathbb{R}^d} \frac{\max\{\pi(s), p_0(s)\} - u(s)}{\pi_Q(s)}, \tag{15}$$

*and $u(s)$ is the density of the measure $U$.*

*2. The classifiers $d_F$, $d_B$ and the backward transition density $q_B$ satisfy*

$$d_F(s) = \frac{r(s)}{f(s)}, \qquad d_B(s) = \frac{Z\, p_0(s)}{f(s)}, \qquad q_B(s \mid s') = \frac{f(s) - r(s)}{f(s') - Z\, p_0(s')} q_F(s' \mid s), \tag{16}$$

*where $f(s) = Z(u(s) + n_Q \pi_Q(s))$ is the density of the flow measure $F$.*

**Corollary 3.7.** *The constant $n_Q$ from Theorem 3.6 coincides with the mean trajectory length $\mathbb{E}_{\tau \sim P}[n_\tau]$. The total flow $F(\mathcal{S})$ is minimized if and only if $n_Q = n_Q^*$.*

We provide the proof of these statements in §E.4. The measure $U(A)$ admits an interpretation as the accumulated discrepancy between injected and removed mass from a measurable set $A$, when we run a Markov chain with the fixed kernel $Q_F$. Additional discussion and interpretation are provided in §D. This result shows that, once the forward kernel $Q_F$ is fixed, we can potentially de-bias it to obtain samples from a target distribution $\pi(x)$ using a classifier. However, doing so requires at least $n_Q^*$ steps on average, given that this quantity is finite.

If the forward kernel $Q_F$ is already invariant with respect to the target distribution, i.e., $\pi_Q = \pi$, then the measure $U$ simplifies to $U(A) = \sum_{n=0}^{\infty}(p_0 Q_F^n - \pi)(A)$, which can be intuitively viewed as an accumulated discrepancy between distributions at step $n$ of the Markov chain $p_0 Q_F^n$ and the target $\pi$.

### 3.3 Training algorithm

In practice, our model includes parametric forward and backward policies $p_F$ and $p_B$ with learnable classifiers $d_F, d_B$ and learnable kernels $q_F, q_B$. Based on our theory, various GFlowNet objectives can be used for training. Detailed balance (DB) [4] is a local objective, and [28] showed that it can be effectively used with state flow regularization to train non-acyclic GFlowNets. However, we found DB to perform poorly in continuous environments, which is consistent with the findings of [5]. Alternatively, subtrajectory balance (SubTB) [25] objective can be used, which incorporates both local and global trajectory information, but introduces a quadratic number of terms in the trajectory length, making it less computationally efficient for training in non-acyclic environments. Recently [29] introduced a computationally efficient prefix trajectory balance objective, which we found well-suited for training in non-acyclic continuous spaces, and use it in our experiments. We provide an ablation of the discussed training objectives in §B.7.

At each iteration, we sample partial trajectories of fixed length $N_{\max}$ with the classifier disabled to encourage exploration. Each prefix of length $i$ is extended with a terminal transition, yielding a complete trajectory $\tau_{0:i}$. For each prefix, we minimize

$$\mathcal{L}_{\mathrm{TB}}(\theta, \tau_{0:i}) = \left(\log \frac{p_0(s_1) \prod_{t=1}^{i} p_F(s_{t+1} \mid s_t; \theta)}{r(s_i)/Z_\theta \prod_{t=0}^{i-1} p_B(s_t \mid s_{t+1}; \theta)}\right)^2, \tag{17}$$

where the normalizing constant $Z_\theta$ is treated as a learnable scalar parameter and allows the model to match the partition function $Z = R(\mathcal{X})$.

To encourage shorter trajectories, we add flow regularization following prior approaches [8, 28], which can be computed directly from the classifier using (12). We sum the loss over all prefixes, weighted by the model's stopping probability at step $i$: $w_i = \mathrm{sg}\big[d_F(s_i; \theta) \prod_{t=1}^{i-1}(1 - d_F(s_t; \theta))\big]$, where sg denotes stopgrad. The weights focus training on prefixes that are likely under the current policy and thus improve stability. The final objective is

$$\mathcal{L}(\theta, \tau, \rho) = \sum_{i=1}^{N_{\max}} w_i \left(\mathcal{L}_{\mathrm{TB}}(\theta, \tau_{0:i}) + \rho \frac{r(s_i)}{d_F(s_i; \theta)}\right), \tag{18}$$

**Algorithm 1** Training non-acyclic continuous GFlowNet with prefix trajectory balance (simplified)

**Input:** Initial parameters $\theta$, reward $R$, initial distribution $p_0$, maximum trajectory length $N_{\max}$, batch size $B$, number of training steps $K$, regularization coefficient $\rho$
**Output:** Trained model parameters $\theta$
**for** $k = 1$ **to** $K$ **do**
  Sample trajectories $\widetilde{\tau} = \{(s_0 \to s_1^b \to \cdots \to s_{N_{\max}}^b)\}_{b=1}^B$ with a classifier disabled
  Set $\tau_{0:i}^b = \widetilde{\tau}_{0:i}^b \cup \{s_f\}$ for $i = 1, \ldots, N_{\max}, b = 1, \ldots, B$
  For all complete prefixes $\{\tau_{0:i}^b\}_{b=1,i=1}^{B,N_{\max}}$, compute TB loss $\mathcal{L}_{\text{TB}}(\theta, \tau_{0:i}^b)$ and weights $\{w_i^b\}_{b=1}^B$
  Compute

$$\mathcal{L}(\theta) = \frac{1}{B}\sum_{b=1}^{B}\sum_{i=1}^{N_{\max}} w_i^b \left[\mathcal{L}_{\text{TB}}(\theta, \tau_{0:i}^b) + \rho \frac{r(s_i^b)}{d_F(s_i^b;\theta)}\right]$$

  Update $\theta$ by a gradient step on $\mathcal{L}(\theta)$
**end for**

where $\rho > 0$ controls flow regularization. The training algorithm is presented in Algorithm 1, with full training details provided in §B.3.

In §B.1, we provide the concrete parameterizations of the continuous kernels $q_F$ and $q_B$. Importantly, $q_F$ can either be fixed, for example to a kernel of some MCMC algorithm like ULA, or learned to additionally improve sampling quality. We extensively study this design choice in our experiments in §4. $q_B$, however, has to be learned in every setup, since the backward policy is a crucial part of the whole mechanism that encourages shorter trajectories (see [28] for a detailed discussion).

### 3.4 Multilevel generalization

We extend the state space with levels variables, allowing the model to switch between regimes during generation. In the basic formulation from §3.1, the transition kernels and classifiers are homogeneous and do not explicitly depend on the generation step. To introduce additional flexibility, we augment the state space and consider continuous states of the form $(s, \ell)$, where $s \in \mathbb{R}^d$ and $\ell \in \{1, \ldots, L\}$, with $L$ denoting the number of levels.

Starting from $s_0$, we sample $s \sim p_0$ at level $\ell = 1$ and evolve the process using $P_F$. The classifier decides whether to continue within the current level or transition to the next one, until level $L$ is reached and the trajectory terminates at $s_f$. The backward dynamics are defined analogously. The resulting process remains non-acyclic, as cycles may occur within each level. Inspired by annealed importance sampling (AIS) [30] and the use of geometric interpolatoins in dynamics-based samplers (e.g., [26, 9], we define intermediate rewards

$$r(s, \ell) = p_0(s)^{1-\beta_\ell} r(s)^{\beta_\ell}, \qquad Z^{(\ell)} = \int_{\mathbb{R}^d} r(s, \ell)\mathrm{d}s, \tag{19}$$

where $0 = \beta_0 < \cdots < \beta_L = 1$ and $Z^{(\ell)}$ is the corresponding normalizing constant for the level $\ell$. This yields a sequence of distributions interpolating between $p_0$ and $r$. For each pair $(\ell - 1, \ell)$, where $l \in \{1, \ldots, L\}$, we solve a GFlowNet problem with initial and target distributions

$$p_0^{(\ell)}(s) \propto r(s, \ell - 1), \qquad \pi^{(\ell)}(s) \propto r(s, \ell). \tag{20}$$

These are implemented using shared forward and backward policies conditioned on the level $\ell$, with learnable constants $Z_\theta^{(\ell)}$. Training follows §3.1, where we sample trajectories of length $N_{\max}$, allocate $N_L = N_{\max}/L$ steps per level, and apply prefix trajectory balance within each segment. The final objective sums losses across all levels with the same weighting and flow regularization.

The multilevel scheme decomposes a difficult generation problem into a sequence of easier ones. Intermediate rewards $r(s, \ell)$ are smoother and closer to each other than the original pair $(p_0, r)$, stabilizing learning and exploration. Correctness follows from the following evident statement.

**Proposition 3.8.** *If each pair of levels $(\ell - 1, \ell)$, where $l \in \{1, \ldots, L\}$, satisfies trajectory balance with initial distribution $p_0^{(\ell)}$ and target $\pi^{(\ell)}$, then the terminating distribution $P_T(A) = \pi(A)$.*

*Remark* 3.9. A related line of work is diffusion-based sampling [47, 36], where samples evolve through a sequence of intermediate distributions, moving to the next level after each transition. In

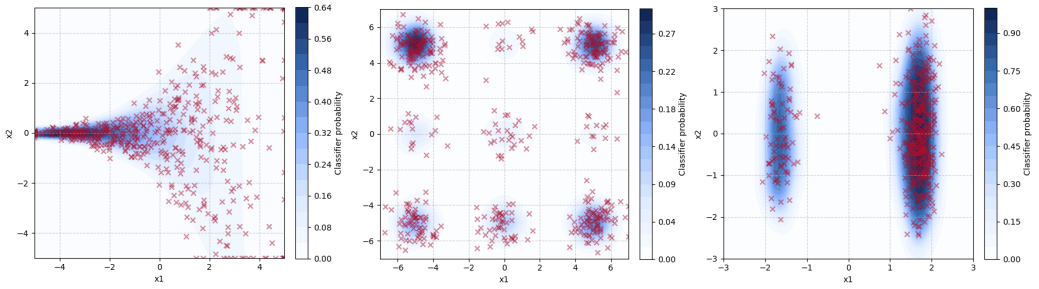



Figure 3: Forward classifier $d_F(s)$ visualizations for different environments (from left to right: Funnel, GMM9, ManyWell, with the first and last projected onto the first two coordinates).

contrast, our model can apply the same transition kernel multiple times per level, using a learned classifier to decide when to advance. This added flexibility allows the sampler to allocate more steps to difficult regions and fewer to simpler ones.

## 4 Experiments

**Improvement over ULA.** In this section, we evaluate the proposed method against the Unadjusted Langevin Algorithm (ULA) and study the effects of classifier-based early stopping and learned corrections to the forward dynamics. With a constant step size, ULA is an inexact MCMC kernel whose invariant distribution generally differs from the target due to discretization bias. This makes it a natural test case for Theorem 3.6, which characterizes when a learned stopping rule can correct the terminal distribution of a fixed kernel. We first evaluate learned stopping with the ULA kernel held fixed, and then investigate how learned drift and variance corrections further improve exploration and approximation quality. Experiments are performed on several synthetic environments described in §B.2. We evaluate the algorithms using standard metrics commonly adopted in the sampling literature, following [7]. Additional details on the metrics are provided in §B.8.

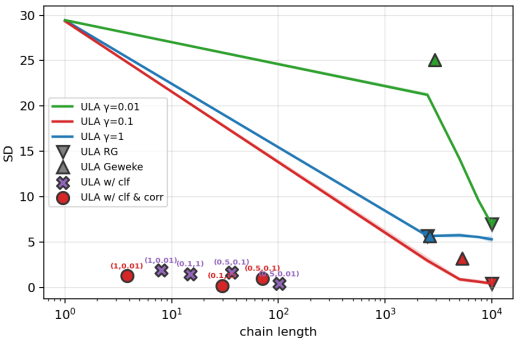



Figure 2: Comparison of our approach to ULA and ULA with stopping based on the Gelman–Rubin and Geweke convergence diagnostics on GMM9 task, where modes are imbalanced. Pairs of two values next to points corresponding to our models denote $(\gamma, \rho)$, step size and regularization coefficient respectively. Baselines (ULA, ULA RG, ULA Geweke) with equal step sizes share the same color. *ULA w/ clf* denotes our model with $q_F$ fixed to the ULA kernel. *ULA w/ clf & corr* denotes our model with learnable $q_F$.

We first consider *ULA w/ clf*, where the forward kernel $q_F$ is fixed to ULA, while $d_F$, $d_B$, and $q_B$ are learned. This configuration isolates the effect of learned stopping with fixed forward dynamics. In this setting, the learned classifier significantly reduces the average trajectory length relative to fixed-step ULA, as shown in Table 1. However, the improvement in sample quality is generally limited.

Next, we study the model *ULA w/ clf & corr*, where the forward kernel $q_F$ is also learned (as a correction to the ULA kernel, see §B.1) in addition to $d_F$, $d_B$, and $q_B$. In this setting, the sampler is able to make substantially more effective transitions between distant high-density regions, leading to improved approximation quality. The learned classifier in this model is visualized in Figure 3. The classifier assigns higher stopping probabilities in regions of high density and lower probabilities elsewhere, effectively recovering the geometry of the target distribution. This behavior is also consistent with the relation $d_F(s) = r(s)/f(s)$, since the flow typically becomes close to uniform within a mode, the classifier tends to resemble the target density itself.

Figure 9 in the Appendix illustrates that ULA tends to remain trapped within one mode, whereas the model with learnable corrections moves between distant modes. For a fixed kernel, the standard ULA typically requires a small step size to control discretization bias, which makes mode transitions rare. Our framework allows to use ULA with larger step size to improve exploration. The classifier can

Table 1: Comparison of various ULA extensions across four target densities. The best result in each row is highlighted in gray. **ULA w/ clf** denotes our model with $q_F$ fixed to the ULA kernel. **ULA w/ clf & corr** denotes our full model with learnable $q_F$. Mean±std over three runs.

| Target → | GMM9 ($d = 2$) | | Funnel ($d = 2$) | | GMM9 ($d = 40$) | | ManyWell ($d = 40$) | |
|---|---|---|---|---|---|---|---|---|
| Algorithm ↓ Metric → | SD ↓ | MMD ↓ | SD ↓ | MMD ↓ | SD ↓ | MMD ↓ | SD ↓ | MMD ↓ |
| ULA | 0.463±0.131 | 0.002±0.005 | 14.101±0.243 | 0.142±0.014 | 376.406±2.165 | 0.347±0.011 | 71.508±0.876 | 0.332±0.018 |
| ULA RG | 0.463±0.154 | 0.002±0.001 | 11.686±0.165 | 0.112±0.002 | 376.652±2.377 | 0.345±0.002 | 70.614±0.189 | 0.328±0.001 |
| ULA Geweke | 3.234±0.243 | 0.010±0.001 | 16.262±0.133 | 0.216±0.004 | 376.407±1.935 | 0.346±0.001 | 73.531±0.081 | 0.338±0.001 |
| **ULA w/ clf** | 0.435±0.038 | 0.032±0.001 | 8.849±0.126 | 0.128±0.003 | 321.803±2.038 | 0.314±0.017 | 54.804±4.203 | 0.298±0.019 |
| **ULA w/ clf & corr** | 0.221±0.026 | 0.027±0.006 | 4.947±0.097 | 0.089±0.013 | 243.809±1.984 | 0.251±0.019 | 62.472±1.237 | 0.314±0.017 |
| Algorithm ↓ Metric → | ELBO ↑ | EUBO ↓ | ELBO ↑ | EUBO ↓ | ELBO ↑ | EUBO ↓ | ELBO ↑ | EUBO ↓ |
| **ULA w/ clf** | −0.631±0.041 | 2.390±0.075 | −1.264±0.038 | 14.462±0.312 | −64.581±0.912 | 174.390±1.726 | 92.507±10.109 | 317.601±15.301 |
| **ULA w/ clf & corr** | −0.157±0.025 | 2.278±0.065 | −0.392±0.023 | 1.005±0.053 | −51.701±0.778 | 168.609±1.618 | 150.008±8.421 | 248.303±9.152 |
| Algorithm ↓ Metric → | $\mathbb{E}[n_\tau]$ ↓ | | $\mathbb{E}[n_\tau]$ ↓ | | $\mathbb{E}[n_\tau]$ ↓ | | $\mathbb{E}[n_\tau]$ ↓ | |
| ULA | $10^4$ | | $5 \times 10^2$ | | $5 \times 10^2$ | | $1.5 \times 10^3$ | |
| ULA RG | $10^4$ | | $10^3$ | | $10^3$ | | $2 \times 10^3$ | |
| ULA Geweke | 5307.304±29.689 | | 406.579±0.662 | | 437.228±1.370 | | 1468.161±8.509 | |
| **ULA w/ clf** | 101.702±0.853 | | 21.842±0.330 | | 37.659±2.589 | | 79.123±4.203 | |
| **ULA w/ clf & corr** | 29.591±0.345 | | 36.337±0.421 | | 27.837±3.494 | | 75.383±2.918 | |

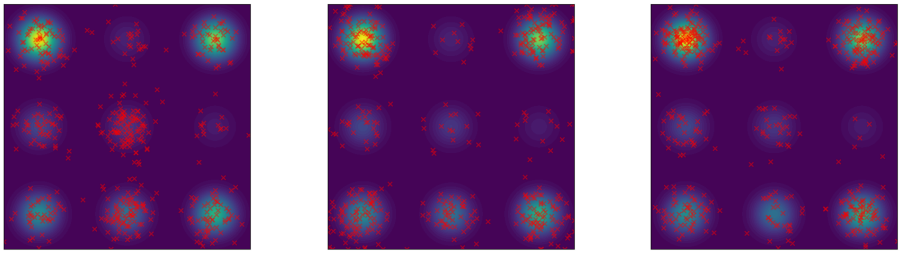


Figure 4: Samples on the imbalanced GMM9 target. Left to right: ULA, ULA with classifier, ULA with classifier and drift correction. True density shown in background. Additional images in §B.9.

then assign low stopping probability to transient or poorly represented regions and terminate primarily in states consistent with the target distribution. For a learnable kernel case, state-dependent drift and variance corrections can make large directed transitions in regions where local ULA dynamics would mix slowly, while retaining small transitions where appropriate.

The quantitative results are summarized in Table 1. Evaluation metrics are Sinkhorn distance (SD), RBF maximum mean discrepancy (MMD), evidence lower and upper bounds (ELBO, EUBO), and the mean trajectory length $\mathbb{E}[n_\tau]$. ULA RG and ULA Geweke denote ULA with stopping based on the Gelman–Rubin and Geweke convergence diagnostics, respectively. For ULA, we report metrics after the chain reaches a stationary regime with respect to the Sinkhorn metric. Overall, classifier-based stopping substantially improves sampling efficiency by reducing the average trajectory length, while learned forward corrections further improve approximation quality, typically achieving lower Sinkhorn distance and MMD, and tighter ELBO/EUBO estimates.

In Figure 2, we compare our methods against ULA and standard stopping diagnostics across different step sizes and numbers of steps on a gaussian mixture target where modes have different weights. Even on the relatively simple two-dimensional imbalanced mixture task, MCMC-based approaches require extremely long mixing times to correctly recover the target distribution, while the learned classifier helps to correctly adjust to the weights of different mixture components. This behavior is also clearly seen in the visualization of the samples in Figure 4.

Finally, one can note that the trade-off between approximation quality and trajectory length in our method depends on the ULA step size $\gamma$ and flow regularization coefficient $\rho$. Larger steps can improve exploration but can also make learning the finer details of the distribution geometry more difficult. Stronger flow regularization encourages earlier termination, although excessive regularization can reduce approximation quality because of the larger errors in balance conditions.

Additional experiments, hyperparameters and visualizations are provided in §B.

Table 2: Comparison of the multilevel GFlowNet with $L \in \{1, 2, 5\}$ against diffusion sampling baselines PIS and DDS on GMM9 ($d = 40$), ManyWell ($d = 40$), and Digits ($d = 196$). The best mean value for each metric and target is highlighted in gray. Mean±std over three runs.

| Target → | GMM9 ($d = 40$) | | ManyWell ($d = 40$) | | Digits ($d = 196$) | |
|---|---|---|---|---|---|---|
| Algorithm ↓ Metric → | SD ↓ | MMD ↓ | SD ↓ | MMD ↓ | SD ↓ | MMD ↓ |
| PIS | 292.878±1.710 | 0.185±0.010 | 38.740±1.329 | 0.258±0.011 | 1728.000±10.351 | 0.540±0.023 |
| DDS | 168.391±1.624 | 0.178±0.014 | 38.810±1.329 | 0.259±0.013 | 202.400±4.189 | 0.346±0.021 |
| **ULA w/ clf & corr** | 243.809±1.984 | 0.251±0.019 | 62.472±1.237 | 0.314±0.017 | 469.500±1.532 | 0.410±0.036 |
| **Multilevel ($L = 2$)** | 186.200±1.723 | 0.203±0.015 | 36.330±0.561 | 0.108±0.021 | 343.600±0.386 | 0.394±0.042 |
| **Multilevel ($L = 5$)** | 152.347±2.281 | 0.184±0.011 | 28.400±4.843 | 0.095±0.020 | 196.938±3.519 | 0.304±0.028 |
| Algorithm ↓ Metric → | ELBO ↑ | EUBO ↓ | ELBO ↑ | EUBO ↓ | ELBO ↑ | EUBO ↓ |
| PIS | −164.103±3.481 | 39.507±4.900 | 200.900±7.924 | 278.900±5.118 | −3671.000±21.931 | 925.200±7.815 |
| DDS | −4.782±1.301 | 4.598±2.218 | 201.100±8.914 | 331.100±7.185 | −149.300±3.194 | 149.900±8.917 |
| **ULA w/ clf & corr** | −51.701±0.778 | 168.609±1.618 | 150.008±8.421 | 248.303±9.152 | −210.340±3.591 | 270.913±5.951 |
| **Multilevel ($L = 2$)** | −34.735±6.192 | 71.100±6.813 | 143.500±4.254 | 254.900±3.138 | −184.391±8.138 | 138.200±6.984 |
| **Multilevel ($L = 5$)** | −28.421±2.156 | 58.734±3.021 | 154.267±3.845 | 221.456±4.932 | −121.570±4.193 | 102.630±3.903 |
| Algorithm ↓ Metric → | $\mathbb{E}[n_\tau]$ ↓ | | $\mathbb{E}[n_\tau]$ ↓ | | $\mathbb{E}[n_\tau]$ ↓ | |
| PIS | 73 | | 75 | | 65 | |
| DDS | 73 | | 75 | | 65 | |
| **ULA w/ clf & corr** | 27.837±3.494 | | 75.383±2.918 | | 71.600±6.139 | |
| **Multilevel ($L = 2$)** | 44.600±3.613 | | 74.173±2.590 | | 62.300±5.058 | |
| **Multilevel ($L = 5$)** | 72.900±4.152 | | 68.730±2.741 | | 68.281±5.300 | |

**Experiments with the multilevel scheme.** We now evaluate the proposed multilevel extension described in §3.4. We consider configurations with $L \in \{1, 2, 5\}$ levels and compare them against established diffusion sampling methods, namely PIS [47] and DDS [40]. Note that a multilevel scheme with $L = 1$ reduces to the single-level scheme with learnable $q_F$.

Table 2 reports results on the GMM9 and ManyWell targets with $d = 40$, as well as on the more challenging $d = 196$ dimensional Digits task from [7], corresponding to sampling MNIST images from a NICE normalizing flow model [11]. For a trained multilevel sampler, we measure it mean trajectory length and train diffusion samplers with the approximately this number of steps. Overall, the multilevel scheme consistently improves sample quality relative to the single-level model, especially on more difficult and high-dimensional tasks. Increasing the number of levels further improves approximation quality, yielding lower SD, MMD, and tighter ELBO/EUBO estimates.

The multilevel construction decomposes the sampling problem into transitions between closer intermediate distributions and allows the transition kernels and stopping classifiers to adapt to each level. Additional levels may increase the total trajectory length. Nevertheless, the sampling effort within each level can be adjusted through flow regularization and a more gradual temperature schedule. However, in all experiments, we use the linear schedule $\beta_\ell = \ell/L$, $\ell = 0, \ldots, L$.

On GMM9 and Digits, the multilevel models significantly outperform the single-level baseline and become competitive with diffusion samplers. In particular, the $L = 5$ configuration achieves the best overall performance among the proposed methods. Additional experiments on the Sonar Bayesian logistic-regression posterior are presented in §B.6, where we also evaluate Laplace-based preconditioning of the multilevel transition kernels.

## 5 Conclusion

We have proposed a novel framework for adaptive sampling that treats trajectory termination as a learnable component of the dynamics. By casting MCMC with stopping decisions into the framework of non-acyclic GFlowNets, we established a connection between optimal stopping policies and the target distribution via flow and detailed balance conditions. Empirically, the method significantly reduces trajectory lengths while improving mode coverage and mixing across a range of standard benchmarks. The proposed approach introduces additional learning components, which increases optimization complexity and can require careful balancing between exploration and regularization during training, posing typical challenges that arise in learning-based sampling methods. Future work could focus on improving training objectives for continuous non-acyclic GFlowNets, as well as extending the framework to more expressive sampling dynamics, e.g., Hamiltonian Monte Carlo [31].

## Acknowledgments and Disclosure of Funding

The work of K. Korolev, N. Morozov, S. Pavlenko, and S. Samsonov was supported by the grant for research centers in the field of AI provided by the Ministry of Economic Development of the Russian Federation in accordance with the agreement 000000C313925P4E0002 and the agreement with HSE University № 139-15-2025-009. This research was supported in part through computational resources of HPC facilities at HSE University [20].

## References

[1] Andrieu, C. and Thoms, J. A tutorial on adaptive MCMC. *Statistics and Computing*, 18(4): 343–373, 2008.

[2] Andrieu, C., De Freitas, N., Doucet, A., and Jordan, M. I. An introduction to MCMC for machine learning. *Machine learning*, 50(1):5–43, 2003.

[3] Bengio, E., Jain, M., Korablyov, M., Precup, D., and Bengio, Y. Flow network based generative models for non-iterative diverse candidate generation. *Neural Information Processing Systems (NeurIPS)*, 2021.

[4] Bengio, Y., Lahlou, S., Deleu, T., Hu, E. J., Tiwari, M., and Bengio, E. GFlowNet foundations. *Journal of Machine Learning Research*, 24(210):1–55, 2023.

[5] Berner, J., Richter, L., Sendera, M., Rector-Brooks, J., and Malkin, N. From discrete-time policies to continuous-time diffusion samplers: Asymptotic equivalences and faster training. *Transactions on Machine Learning Research*, 2026.

[6] Biron-Lattes, M., Surjanovic, N., Syed, S., Campbell, T., and Bouchard-Cote, A. autoMALA: Locally adaptive Metropolis-adjusted Langevin algorithm. *Artificial Intelligence and Statistics (AISTATS)*, 2023.

[7] Blessing, D., Jia, X., Esslinger, J., Vargas, F., and Neumann, G. Beyond ELBOs: A large-scale evaluation of variational methods for sampling. *International Conference on Machine Learning (ICML)*, 2024.

[8] Brunswic, L., Li, Y., Xu, Y., Feng, Y., Jui, S., and Ma, L. A theory of non-acyclic generative flow networks. *Association for the Advancement of Artificial Intelligence (AAAI)*, 2024.

[9] Choi, S., Mittal, S., Elvira, V., Park, J., and Whitammer, E. S. Reinforced sequential Monte Carlo for amortised sampling. *International Conference on Machine Learning (ICML)*, 2026.

[10] Deleu, T. and Bengio, Y. Generative flow networks: a markov chain perspective. *arXiv preprint arXiv:2307.01422*, 2023.

[11] Dinh, L., Krueger, D., and Bengio, Y. NICE: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*, 2014.

[12] Douc, R., Moulines, E., Priouret, P., and Soulier, P. *Markov chains*. Springer Series in Operations Research and Financial Engineering. Springer, 2018.

[13] Gabrié, M., Rotskoff, G. M., and Vanden-Eijnden, E. Adaptive Monte Carlo augmented with normalizing flows. *Proceedings of the National Academy of Sciences*, 119(10):e2109420119, 2022.

[14] Gretton, A., Borgwardt, K. M., Rasch, M. J., Schölkopf, B., and Smola, A. A kernel two-sample test. *Journal of Machine Learning Research*, 13(1):723–773, 2012.

[15] Gritsaev, T., Morozov, N., Tamogashev, K., Tiapkin, D., Samsonov, S., Naumov, A., Vetrov, D., and Malkin, N. Adaptive destruction processes for diffusion samplers. *arXiv preprint arXiv:2506.01541*, 2025.

[16] Haario, H., Saksman, E., and Tamminen, J. An adaptive Metropolis algorithm. *Bernoulli*, 7(2): 223 – 242, 2001.

[17] Hoffman, M. D., Gelman, A., et al. The No-U-Turn sampler: adaptively setting path lengths in Hamiltonian Monte Carlo. *Journal of Machine Learning Research*, 15(1):1593–1623, 2014.

[18] Izmailov, P., Vikram, S., Hoffman, M. D., and Wilson, A. G. G. What are Bayesian neural network posteriors really like? *International Conference on Machine Learning (ICML)*, 2021.

[19] Kim, M., Seong, K., Woo, D., Ahn, S., and Kim, M. On scalable and efficient training of diffusion samplers. *Neural Information Processing Systems (NeurIPS)*, 2025.

[20] Kostenetskiy, P., Chulkevich, R., and Kozyrev, V. Hpc resources of the higher school of economics. In *Journal of Physics: Conference Series*, volume 1740, pp. 012050. IOP Publishing, 2021.

[21] Kwon, S., Zhang, S., Köhn, H. F., and Zhang, B. MCMC stopping rules in latent variable modelling, 2025.

[22] Lahlou, S., Deleu, T., Lemos, P., Zhang, D., Volokhova, A., Hernández-Garcıa, A., Ezzine, L. N., Bengio, Y., and Malkin, N. A theory of continuous generative flow networks. *International Conference on Machine Learning (ICML)*, 2023.

[23] Liu, J. S. and Liu, J. S. *Monte Carlo strategies in scientific computing*, volume 10. Springer, 2001.

[24] Liu, T., Surjanovic, N., Biron-Lattes, M., Bouchard-Cote, A., and Campbell, T. AutoStep: Locally adaptive involutive MCMC. *International Conference on Machine Learning (ICML)*, 2025.

[25] Madan, K., Rector-Brooks, J., Korablyov, M., Bengio, E., Jain, M., Nica, A. C., Bosc, T., Bengio, Y., and Malkin, N. Learning GFlowNets from partial episodes for improved convergence and stability. *International Conference on Machine Learning (ICML)*, 2023.

[26] Máté, B. and Fleuret, F. Learning interpolations between Boltzmann densities. *Transactions on Machine Learning Research (TMLR)*, 2023.

[27] Midgley, L. I., Stimper, V., Simm, G. N. C., Schölkopf, B., and Hernández-Lobato, J. M. Flow annealed importance sampling bootstrap. *International Conference on Learning Representations (ICLR)*, 2023.

[28] Morozov, N., Maksimov, I., Tiapkin, D., and Samsonov, S. Revisiting non-acyclic GFlowNets in discrete environments. *International Conference on Machine Learning (ICML)*, 2025.

[29] Morozov, N., Maksimov, I., Tiapkin, D., and Samsonov, S. Learning shortest paths with generative flow networks. *arXiv preprint arXiv:2603.01786*, 2026.

[30] Neal, R. M. Annealed importance sampling. *Statistics and computing*, 11(2):125–139, 2001.

[31] Neal, R. M. et al. MCMC using Hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo*, 2(11):2, 2011.

[32] Noé, F., Olsson, S., Köhler, J., and Wu, H. Boltzmann generators: Sampling equilibrium states of many-body systems with deep learning. *Science*, 365(6457), 2019.

[33] Peyré, G. and Cuturi, M. *Computational optimal transport: With applications to data science*. Now Foundations and Trends, 2019.

[34] Roy, V. Convergence diagnostics for Markov chain Monte Carlo. *Annual Review of Statistics and Its Application*, 7:387–412, 2020.

[35] Roy, V. and Zhang, L. Convergence of position-dependent MALA with application to conditional simulation in GLMMs. *Journal of Computational and Graphical Statistics*, 32(2):501–512, 2023.

[36] Sendera, M., Kim, M., Mittal, S., Lemos, P., Scimeca, L., Rector-Brooks, J., Adam, A., Bengio, Y., and Malkin, N. Improved off-policy training of diffusion samplers. *Neural Information Processing Systems (NeurIPS)*, 2024.

[37] Stoltz, G., Rousset, M., et al. *Free energy computations: A mathematical perspective*. World Scientific, 2010.

[38] Tiapkin, D., Morozov, N., Naumov, A., and Vetrov, D. P. Generative flow networks as entropy-regularized RL. *Artificial Intelligence and Statistics (AISTATS)*, 2024.

[39] Titsias, M. and Dellaportas, P. Gradient-based adaptive Markov chain Monte Carlo. *Neural Information Processing Systems (NeurIPS)*, 2019.

[40] Vargas, F., Grathwohl, W., and Doucet, A. Denoising diffusion samplers. *International Conference on Learning Representations (ICLR)*, 2023.

[41] Vihola, M. Robust adaptive metropolis algorithm with coerced acceptance rate. *Statistics and Computing*, 22(5):997–1008, 2012.

[42] Wang, C., Chen, W., Kanagawa, H., and Oates, C. J. Reinforcement learning for adaptive MCMC. *Artificial Intelligence and Statistics (AISTATS)*, 2025.

[43] Wang, C., Fisher, M. A., Kanagawa, H., Chen, W., and Oates, C. J. Harnessing the power of reinforcement learning for adaptive MCMC. *arXiv preprint arXiv:2507.00671*, 2025.

[44] Welling, M. and Teh, Y. W. Bayesian learning via stochastic gradient Langevin dynamics. 2011.

[45] Wu, J. and Kawai, R. Stopping rules for Monte Carlo methods: A review. *arXiv preprint arXiv:2510.22688*, 2025.

[46] Zhang, D., Chen, R. T., Liu, C.-H., Courville, A., and Bengio, Y. Diffusion generative flow samplers: Improving learning signals through partial trajectory optimization. *International Conference on Learning Representations (ICLR)*, 2024.

[47] Zhang, Q. and Chen, Y. Path integral sampler: A stochastic control approach for sampling. *International Conference on Learning Representations (ICLR)*, 2022.

## A GFlowNets on general state spaces

This section gives a more detailed description of GFlowNets on general state spaces using the measure-theoretic notation of [22].

Given a measurable space $(\mathcal{X}, \Sigma_{\mathcal{X}})$, and a finite reward measure $R$, we aim to sample from the distribution $\pi$ defined by

$$\pi(A) = R(A)/R(\mathcal{X}), \quad A \in \Sigma_{\mathcal{X}}\,.$$

Following [22], the problem can be solved with an extension of GFlowNets, which can be defined on general state spaces using a measure-theoretic extension of directed acyclic graphs. This structure is represented by a *measurable pointed graph* $\mathcal{G} = (\bar{\mathcal{S}}, \Sigma, s_0, s_f, \kappa_F, \kappa_B, \nu)$. Here, $(\bar{\mathcal{S}}, \Sigma)$ is a measurable space with $\bar{\mathcal{S}} = \{s_0\} \cup \mathcal{S} \cup \{s_f\}$, where $s_0, s_f \in \bar{\mathcal{S}}$ are the distinguished source and sink states. We assume that $\mathcal{X} = \{s \in \mathcal{S} : \kappa_F(s, \{s_f\}) > 0\}$ denote the set of terminal states, and $\nu$ is some reference measure on $\mathcal{S}$, such that $R \ll \nu$. In this statement, $\kappa_F(s, \cdot)$, and $\kappa_B(s, \cdot)$ are the reference forward and backward kernels. The supports of $\kappa_F(s, \cdot)$ and $\kappa_B(s, \cdot)$ generalize the notions of children and parents in a discrete DAG, respectively, defining the set of states reachable from $s$ (resp. from which $s$ is reachable) in one step. The kernels $\kappa_F$ and $\kappa_B$ must satisfy several compatibility conditions, in particular, it should exist $N \in \mathbb{N}$, such that

$$\operatorname{supp}(\kappa_F^N(s_0, \cdot)) = \{s_f\}\,. \tag{21}$$

**Trajectory distribution and sampling problem.** Suppose that we are given a *forward* Markov kernel $P_F$ on $(\bar{\mathcal{S}}, \Sigma)$ such that $P_F(s, \cdot) \ll \kappa_F(s, \cdot)$. Learner should be able to sample from $P_F(s, \cdot)$. The kernel $P_F(s, \cdot)$ naturally induces a measure over $(\mathcal{S}^n, \Sigma^{\otimes n})$ given by

$$P_F^{\otimes n}(s_0, \mathrm{d}s_{1:n}) = \textstyle\prod_{i=1}^n P_F(s_{i-1}, \mathrm{d}s_i)\,. \tag{22}$$

Under appropriate regularity conditions (see [22]), $P_F^{\otimes n}$ induces the probability distribution over the complete trajectories $\tau = (s_0 \to s_1 \to \cdots \to s_n \to s_{n+1} = s_f)$ of all possible length $n \in \mathbb{N}$ given for any $C_n \in \Sigma^n$ by

$$P(\tau \in (\{s_0\} \otimes C_n \otimes \{s_f\})) = \textstyle\int_{\bar{\mathcal{S}}^{n+1}} P_F^{\otimes(n+1)}(s_0, \mathrm{d}s_{1:n+1}) \mathbb{I}\{(s_1, \ldots, s_n) \in C_n\} \mathbb{I}\{s_{n+1} = s_f\}\,.$$

Note that the condition (21) implies that a trajectory sampled starting from $s_0$ according to the probability distribution $P(\tau)$ defined above will terminate with probability 1 in the number of steps not exceeding $N$. Marginalization of the above trajectory distribution $P(\tau)$ over the terminal states $s_n$ induces the terminating distribution $P_{\mathrm{T}}(B)$ on $(\mathcal{X}, \Sigma_{\mathcal{X}})$. The goal of our sampling procedure is to ensure proper conditions on the forward Markov kernel $P_F$ such that for any $A \in \Sigma_{\mathcal{X}}$

$$P_{\mathrm{T}}(A) = \pi(A)\,. \tag{23}$$

**Local balance conditions.** Given a $\sigma$-finite measure $F \ll \nu$ over $(\bar{\mathcal{S}}, \Sigma)$ and a Markov kernel $P_F$, the tuple $(F, P_F)$ is said to satisfy *flow-matching conditions* if for any $A \subseteq \bar{\mathcal{S}} \setminus \{s_0\}$:

$$F(A) = \textstyle\int_{\bar{\mathcal{S}} \setminus \{s_f\}} F(\mathrm{d}s)\, P_F(s, A), \tag{24}$$

where $F$ and $P_F$ are said to be a flow measure and a forward kernel, respectively, over $\mathcal{G}$. The tuple $(F, P_F)$ satisfies the *reward-matching conditions* if for any $x \in \mathcal{X}$:

$$R(\mathrm{d}x) = F(\mathrm{d}x)\, P_F(x, \{s_f\}). \tag{25}$$

Specifically, [22] showed that when the flow and reward-matching conditions are satisfied, the terminating distribution $P_{\mathrm{T}}$ induced by $P_F$ is proportional to $R$, and (23) holds.

**Detailed balance.** Given a backward Markov kernel $P_B$ on $(\bar{\mathcal{S}}, \Sigma)$ with $P_B(s', \cdot) \ll \kappa_B(s', \cdot)$, the tuple $(F, P_F, P_B)$ is said to satisfy *detailed balance conditions* if:

$$F(\mathrm{d}s)\, P_F(s, \mathrm{d}s') = F(\mathrm{d}s')\, P_B(s', \mathrm{d}s). \tag{26}$$

Furthermore, [22] demonstrated that if $(F, P_F, P_B)$ satisfies the detailed balance conditions, then $(F, P_F)$ satisfies the flow-matching conditions. Equations (25) and (26) generalize the familiar equations in the discrete case [4] for a reward function $R : \mathcal{X} \to \mathbb{R}^+$, a flow function $F : \bar{\mathcal{S}} \to \mathbb{R}_{\geq 0}$ and policies $P_F, P_B$ represented as transition probability masses:

$$R(x) = F(x) P_F(s_f \mid x), \quad F(s) P_F(s' \mid s) = F(s') P_B(s \mid s'). \tag{27}$$

If one defines $r(s), f(s), p_F(s' \mid s)$ and $p_B(s \mid s')$ as the Radon–Nikodym derivatives of the reward measure $R$, flow measure $F$, forward kernel $P_F$, and backward kernel $P_B$ with respect to the corresponding reference measures and kernels, then (25) and (26) can be rewritten in terms of densities to resemble (27) directly:

$$\begin{aligned} r(x) &= f(x)p_F(s_f \mid x) \quad \text{for } \nu\text{-a.e. } x \in \mathcal{X}, \\ f(s)p_F(s' \mid s) &= f(s')p_B(s \mid s') \quad \text{for } (\nu \otimes \kappa_F)\text{-a.e. } (s, s'). \end{aligned} \tag{28}$$

**Trajectory balance.** Fix $n \in \mathbb{N}$ and let $\tau = (s_0 \to s_1 \to \cdots \to s_n = x \to s_{n+1} = s_f)$ denote a complete trajectory of length $n$ starting at $s_0$, passing through non-terminal states in $\mathcal{S}$, reaching a terminal state $s_n = x \in \mathcal{X}$, and then transitioning to $s_f$. Given $Z \in \mathbb{R}^+$, the tuple $(Z, P_F, P_B)$ is said to satisfy trajectory balance conditions if for any $n \in \mathbb{N}$,

$$Z \prod_{t=0}^{n} p_F(s_{t+1} \mid s_t) = r(s_n) \prod_{t=0}^{n-1} p_B(s_t \mid s_{t+1}) \quad \text{for } \kappa_F^{\otimes n}\text{-a.e. } (s_0, s_1, \ldots, s_n, s_f)\,. \tag{29}$$

Moreover, [22] showed that the trajectory balance conditions imply both flow-matching and reward-matching.

Note that the framework described above corresponds to the finitely absorbing setting (21), when the trajectories sampled according to $P_F(s, \cdot)$ are guaranteed to have finite length. We show how this setting can be generalized under Assumption 3.1, see the detailed discussion in §3.1.

# B Experimental details

## B.1 Continuous kernels parameterizations

We initialize trajectories from the Gaussian distribution

$$p_0(s) = \mathcal{N}(s \mid 0, \sigma_0^2 I).$$

For transitions between continuous states, we use the forward density

$$\begin{aligned} q_F(s' \mid s; \theta) &= \mathcal{N}(s' \,|\, \mu_F(s;\theta),\, \Sigma_F(s;\theta))\,, \\ \mu_F(s;\theta) &= s + \gamma\Big(u_F(s;\theta) + \big(\mathbf{1} + v_F(s;\theta)\big) \odot \nabla \log r(s)\Big), \\ \Sigma_F(s;\theta) &= 2\gamma\, \mathrm{diag}\big(\exp\{C_F \tanh(w_F(s;\theta))\}\big), \end{aligned} \tag{30}$$

where $u_F, v_F, w_F : \mathbb{R}^d \to \mathbb{R}^d$ are learnable functions, $\mathbf{1}$ is a vector of $d$ ones, $\odot$ is a Hadamard product, $\gamma$ is the step size and we set $C_F = 4$ across all experiments. The forward transition density $q_F$ is chosen following the Langevin-type parameterization used in prior works [47], [36]. In particular, it resembles a discretization of Langevin dynamics, augmented with a learned correction to the drift term and a state-dependent scaling of the target score $\nabla \log r(s)$. We also found it beneficial to learn the variance of the forward process, which provides additional adaptability compared to fixed-noise parameterizations. Note that when $u_F \equiv 0$, $v_F \equiv 0$, and $w_F \equiv 0$, the forward transition reduces to the standard Unadjusted Langevin Algorithm (ULA) kernel.

For the backward transitions, we use the density

$$\begin{aligned} q_B(s \mid s'; \theta) &= \mathcal{N}(s \,|\, \mu_B(s';\theta),\, \Sigma_B(s';\theta))\,, \\ \mu_B(s';\theta) &= s' - \gamma\, \mathrm{softplus}(u_B(s';\theta))\, s', \\ \Sigma_B(s';\theta) &= \gamma\, \mathrm{diag}\big(\exp\{C_B \tanh(w_B(s';\theta))\}\big), \end{aligned} \tag{31}$$

where $u_B, w_B : \mathbb{R}^d \to \mathbb{R}^d$ are learnable functions and $C_B = 4$. Here, we employ a homogeneous kernel, as our formulation does not rely on an explicit notion of time. The backward density is inspired by an Ornstein–Uhlenbeck-type process targeting a zero-mean distribution, effectively acting as a mechanism for destructing or reversing the forward generative process, similar in spirit to diffusion samplers [47]. We further introduce a learned correction $u_B$ to the drift of the backward kernel, following [15], where the backward process is enhanced by learnable adjustments to both its drift and variance.

We now justify why Assumption 3.1 holds in our setting in practice. Provided that $p = \inf_{s \in \mathbb{R}^d} d_F(s) > 0$, we have $\mathbb{E}_{\tau \sim P}[n_\tau] \le \mathbb{E}[\xi] = 1/p < \infty$, where $\xi \sim \mathrm{Geom}(p)$. We impose a lower-bound for a classifier $d_F(s) \ge \varepsilon$ for any $s \in \mathbb{R}^d$, where $\varepsilon = 10^{-5}$.

### B.2 Target densities

We evaluate the proposed method on a set of synthetic benchmark distributions commonly used to assess sampling algorithms in continuous spaces. These environments are designed to capture different challenges such as multimodality or complex geometry.

**GMM9.** The GMM9 environment is a Gaussian mixture model with nine components arranged randomly in [-5,5]. Each component has diagonal covariance and the mixture weights are non-uniform, making the distribution both multimodal and imbalanced. Formally, the density is given by a mixture of Gaussian components

$$\pi(x) = \sum_{k=1}^{9} w_k \, \mathcal{N}(x \mid \mu_k, \Sigma_k),$$

where $\Sigma_k = 0.55 \, I_d$ and $\sum_{k=1}^{9} w_k = 1$. This environment tests the ability of the sampler to correctly capture multiple modes and their relative probabilities.

**Funnel.** The Funnel distribution is a hierarchical model with strong scale variation across dimensions. The first coordinate $x_1$ controls the variance of the remaining coordinates:

$$x_1 \sim \mathcal{N}(0, 3^2), \qquad x_{2:d} \sim \mathcal{N}(0, \exp(x_1) I).$$

This leads to regions of very different scales, creating a narrow "funnel" structure. Such distributions are well known to be challenging for MCMC methods due to poor mixing between regions of different variance.

**ManyWell.** The ManyWell density consists of multiple independent double-well potentials, leading to an exponentially large number of modes in higher dimensions. The energy is constructed as a sum of double-well energies:

$$U(x_1, x_2) = a x_1 + b x_1^2 + c x_1^4 + \frac{x_2^2}{2},$$

which induces a multimodal distribution with well-separated modes, where we set $a = -0.5, b = -6.0, c = 1.0$. This environment is particularly challenging because it requires efficient transitions between a large number of modes.

**Digits.** The Digits environment is adopted from [7] and corresponds to sampling MNIST images from a density defined by a NICE normalizing flow model [11]. The target distribution is a continuous density over $\mathbb{R}^{196}$, obtained by training an invertible flow on $14 \times 14$ MNIST images. Although the underlying dataset consists of images, the resulting target distribution has full support on $\mathbb{R}^{196}$. Samples from this environment therefore correspond to continuous vectors that can be reshaped into grayscale digit images. This task provides a substantially more challenging high-dimensional benchmark than the synthetic multimodal targets considered in the main text.

### B.3 Algorithm details and hyperparameters

We train the proposed non-acyclic GFlowNet using the prefix trajectory balance objective described in §3.1. The full version of the training algorithm is summarized in Algorithm 2. In this section, we provide additional implementation details.

At each iteration, the algorithm alternates between forward (on-policy) and backward (off-policy) updates. During forward updates, trajectories are sampled from the forward policy starting from the initial distribution $p_0$. During backward updates, terminal states are sampled from a replay buffer and trajectories are generated using the backward policy. This alternating scheme allows the model to combine on-policy exploration with off-policy refinement and was inspired by [36]. Across all experiments, we used a replay ratio of $N_{\text{RR}} = 2$, which specifies the number of off-policy iterations performed within each on-policy iteration.

Practically, collecting data with early termination hinders exploration of the continuous space. This issue motivates us to disable classifiers during trajectory collection and instead sample trajectories of a fixed length $N_{\max}$. We choose $N_{\max}$ based on the number of iterations required for ULA with the same step-size $\gamma$ to reach a stationary regime, monitored via the Wasserstein metric. If $N_{\max}$ is chosen too small, trajectories may fail to reach proper high-density regions, resulting in poor exploration of

**Algorithm 2** Training non-acyclic continuous GFlowNet with prefix trajectory balance

**Input:** Initial parameters $\theta$, reward $R$, initial distribution $p_0$, maximum trajectory length $N_{\max}$, batch size $B$, number of training steps $K$, replay buffer $\mathcal{B}$, regularization coefficient $\rho$, replay ratio $N_{\text{RR}}$, local search frequency $N_{\text{LS}}$
**Output:** Trained model parameters $\theta$
Pre-fill buffer $\mathcal{B}$ with terminal states from $p_F$
**for** $k = 1$ **to** $K$ **do**
  **if** $(k-1) \mod N_{\text{RR}} \equiv 0$ **then**
    Sample initial states $\{s_1^b\}_{b=1}^B \sim p_0$
    Sample trajectories $\widetilde{\tau} = \{(s_0 \to s_1^b \to \cdots \to s_{N_{\max}}^b = x^b)\}_{b=1}^B$ using $p_F$ with $d_F$ disabled
    Add $\{x^b\}_{b=1}^B$ to $\mathcal{B}$ with priorities $r(x^b)$
  **else**
    Sample terminal states $\{x^b\}_{b=1}^B \sim \mathcal{B}$
    Sample trajectories $\widetilde{\tau} = \{(s_0 \to s_1^b \to \cdots \to s_{N_{\max}}^b = x^b)\}_{b=1}^B$ using $p_B$ with $d_B$ disabled
  **end if**
  Set $\tau_{0:i}^b = \widetilde{\tau}_{0:i}^b \cup \{s_f\}$ for $i = 1, \ldots, N_{\max}, b = 1, \ldots, B$
  For all complete prefixes $\{\tau_{0:i}^b\}_{b=1,i=1}^{B,N_{\max}}$, compute TB loss $\mathcal{L}_{\text{TB}}(\theta, \tau_{0:i}^b)$ and weights $\{w_i^b\}_{b=1}^B$
  Compute

$$\mathcal{L}(\theta) = \frac{1}{B} \sum_{b=1}^{B} \sum_{i=1}^{N_{\max}} w_i^b \left[ \mathcal{L}_{\text{TB}}(\theta, \tau_{0:i}^b) + \rho \frac{r(s_i^b)}{d_F(s_i^b; \theta)} \right]$$

  Update $\theta$ by a gradient step on $\mathcal{L}(\theta)$
  **if** $(k-1) \mod N_{\text{LS}} \equiv 0$ **then**
    Sample $\{z^b\}_{b=1}^B \sim \mathcal{B}$, run MALA to obtain $\{\bar{z}^b\}_{b=1}^B$, and add $\{\bar{z}^b\}_{b=1}^B$ to $\mathcal{B}$
  **end if**
**end for**

the state space. Conversely, setting a very large $N_{\max}$ slows down training and is impractical, though it remains functional because we apply flow regularization to encourage shorter trajectories.

Following [36], to improve sample efficiency, we employ a prioritized replay buffer that stores terminal states. Each state is assigned a priority based on its reward, which biases sampling toward high-reward regions of the state space. The buffer is updated continuously as new samples are generated from the forward policy. Before training, we pre-fill the buffer with $100 \times B$ terminal samples from a forward policy initialized as Langevin dynamics, where $B$ is the batch size. This initial pre-filling improves training at the beginning.

We further incorporate a local search procedure based on the Metropolis-adjusted Langevin algorithm (MALA) [36]. At regular intervals $N_{\text{LS}}$, states sampled from the replay buffer are refined using a small number of MALA steps, and the resulting samples are added back to the buffer. This improves exploration by enabling transitions between distant high-density regions and helps mitigate the limitations of purely learned dynamics. In our implementation, we used 300 MALA steps with $\gamma = 10^{-3}$ and set $N_{\text{LS}} = 25$.

The forward and backward policies are parameterized by neural networks as described in §B.1. We use a shared backbone consisting of a 3-layer MLP for forward and backward embeddings, followed by a 2-layer MLP for their respective specific predictions, with a hidden dimension of 128. The forward continuous kernel $q_F$ is either fixed (in the ULA-based setup) or augmented with learned drift and variance corrections, while the backward kernel is always learned. The forward and backward classifiers control the stopping behavior and are trained jointly with the transition kernels. The normalizing constant $\log Z_\theta$ is also learned as a scalar parameter with a learning rate $10^{-1}$. We set the variance of the initial distribution $\sigma_0^2 = 1$ for low-dimensional environments and $\sigma_0^2 = 5$ for high-dimensional ones.

Optimization is performed using Adam with a batch size of 2000 with gradient clipping. Environment-specific hyperparameters are summarized in the next sections, where we discuss experiments in details. All experiments were performed on NVIDIA A100 GPUs. Our implementations are based upon the published code of [7].

### B.4 Details on the improvement over ULA

As additional stopping-based baselines, we consider ULA with the Gelman–Rubin convergence diagnostic (ULA RG) and ULA with the Geweke convergence diagnostic (ULA Geweke). Formal definitions of the corresponding diagnostics are provided in §C. For ULA RG, we run several parallel ULA chains and evaluate the multivariate $\hat{R}$ statistic on a predefined grid of trajectory lengths $t$ using only trajectory prefixes of length $t$. Sampling is stopped at the first checkpoint where the maximum $\hat{R}$ value across coordinates falls below a fixed threshold. In our experiments, we use a threshold of 1.1. If the criterion is never satisfied before the final checkpoint, the trajectory is truncated at the maximal allowed length. For ULA Geweke, we independently analyze each trajectory by comparing statistics of an early trajectory window (first 10% of samples) and a late window (last 50% of samples) within the prefix of length $t$. A Geweke $Z$-score is computed for each coordinate, and the stopping time is defined as the first checkpoint where the maximum absolute $Z$-score across coordinates falls below a threshold of 4.0. Unlike ULA RG, the stopping time for ULA Geweke is determined independently for each chain.

In Table 3 we specify hyperparameters for the runs in Table 1.

Table 3: Environment-specific hyperparameters for ULA experiments, where lr is a learning rate of the optimizer, $\rho$ is a regularization coefficient, $\gamma$ is the step-size, $N_{\max}$ is trajectory length during training and grad clip specifies values of the gradient clipping with respect to parameters in the optimizer.

| Environment ↓ Parameter → | lr | $\rho$ | $\gamma$ | $N_{\max}$ | grad clip |
|---|---|---|---|---|---|
| GMM9 ($d = 2$) | $10^{-3}$ | $10^{-1}$ | $10^{-1}$ | $10^{2}$ | 1.0 |
| Funnel ($d = 2$) | $10^{-4}$ | $10^{-2}$ | $10^{-2}$ | $10^{2}$ | $10^{-3}$ |
| GMM9 ($d = 40$) | $10^{-4}$ | $10^{0}$ | $10^{-1}$ | $10^{2}$ | $10^{-3}$ |
| ManyWell ($d = 40$) | $10^{-5}$ | $10^{-1}$ | $10^{-2}$ | $10^{2}$ | $10^{-5}$ |

### B.5 Details on the multilevel scheme experiments

In Table 4 we specify hyperparameters for the runs in Table 2.

Table 4: Environment-specific hyperparameters for multi-level experiments, where lr is a learning rate of the optimizer, $\rho$ is a regularization coefficient, $\gamma$ is the step-size, $N_{\max}$ is trajectory length during training and grad clip specifies values of the gradient clipping with respect to parameters in the optimizer.

| Environment ↓ Parameter → | lr | $\rho$ | $\gamma$ | $N_{\max}$ | grad clip |
|---|---|---|---|---|---|
| GMM9 ($d = 40$) | $10^{-4}$ | $10^{0}$ | $10^{-1}$ | $10^{2}$ | $10^{-3}$ |
| ManyWell ($d = 40$) | $10^{-5}$ | $10^{-1}$ | $10^{-2}$ | $10^{2}$ | $10^{-5}$ |
| Digits ($d = 196$) | $10^{-4}$ | $10^{0}$ | $10^{-1}$ | $5 \times 10^{1}$ | $10^{-3}$ |

### B.6 Experiment on a Bayesian logistic regression

We evaluate our method on the Sonar Bayesian logistic-regression target from the benchmark used by [36]. The posterior is 61-dimensional and is defined using 208 labeled observations. Given $(X, Y) \in \mathbb{R}^{208\times 61} \times \{0,1\}^{208}$, the prior and likelihood are

$$w \sim \mathcal{N}(0, I_d), \qquad y_i \mid w \sim \text{Bernoulli}\big(\sigma(x_i^\top w)\big), \qquad d = 61, \quad i = 1, \dots, 208,$$

where $\sigma$ denotes the sigmoid function and the observations are conditionally independent given $w$. Our objective is to sample from the posterior $\pi(w) = p(w \mid X, Y)$, whose normalizing constant and exact samples are unavailable.

**Experimental setup.** We compare PIS and DDS with our single-level model (*ULA w/ clf & corr*) and multilevel models with $L \in \{2, 5\}$. The diffusion baselines use the publicly available implementation of [36]. Following the comparison protocol in the main text, we first measure the

Table 5: Comparison on the Sonar Bayesian logistic-regression target. PIS and DDS use a fixed horizon of 64 steps. The final row removes replay and MALA local search from the preconditioned $L = 5$ model. The best mean value for each metric is highlighted in gray. Mean±std over three runs.

| Target → | Sonar ($d = 61$) | |
|---|---|---|
| Algorithm ↓ Metric → | ELBO ↑ | $\mathbb{E}[n_\tau]$ ↓ |
| PIS | $-142.916_{\pm 3.395}$ | 64 |
| DDS | $-121.365_{\pm 5.093}$ | 64 |
| **ULA w/ clf & corr** | $-166.200_{\pm 4.381}$ | $59.610_{\pm 5.729}$ |
| **Multilevel ($L = 2$)** | $-133.891_{\pm 5.913}$ | $61.973_{\pm 3.184}$ |
| **Multilevel ($L = 5$)** | $-113.903_{\pm 3.810}$ | $63.503_{\pm 4.031}$ |
| **Multilevel ($L = 5$), preconditioned** | $-109.905_{\pm 1.289}$ | $65.793_{\pm 2.602}$ |
| **w/o replay and local search** | $-116.549_{\pm 2.093}$ | $65.014_{\pm 2.785}$ |

mean trajectory length of the trained multilevel sampler and then train the diffusion baselines with approximately the same number of steps. For Sonar, PIS and DDS use 64 steps, approximately matching the mean trajectory length of the unpreconditioned $L = 5$ model.

For our algorithms, we use a learning rate of $3 \times 10^{-4}$, step size $\gamma = 0.1$, regularization coefficient $\rho = 0.01$, and training trajectory length $N_{\max} = 100$. We select $N_{\max}$ using pilot ULA runs with the same step size, choosing the smallest tested horizon after which the empirical effective sample size estimates stabilize.

Since reference posterior samples are unavailable, we omit SD, MMD, and EUBO, which require samples from the target. Following the evaluation protocol for Bayesian posterior targets in [36], we report ELBO, together with the mean trajectory length. Results are reported as mean and standard deviation over three runs.

**Preconditioned Langevin parameterization.** To study the benefits of incorporating posterior geometry into the transition kernel, we additionally consider an explicitly preconditioned multilevel model. For each intermediate target

$$\pi_\ell(s) \propto p_0(s)^{1-\beta_\ell} \pi(s)^{\beta_\ell},$$

we construct a preconditioning matrix from a regularized Laplace approximation,

$$M_\ell = \left(-\nabla^2 \log \pi_\ell(\widehat{s}_\ell) + \lambda I\right)^{-1}, \qquad \lambda = 10^{-4},$$

where $\widehat{s}_\ell$ is a numerically estimated mode of $\pi_\ell$. The matrices are computed before training and held fixed.

Let $L_\ell$ denote the Cholesky factor of $M_\ell$, so that $M_\ell = L_\ell L_\ell^\top$. Using the notation of §B.1, the preconditioned forward kernel has mean and covariance

$$\begin{aligned} \mu_F(s, \ell; \theta) &= s + \gamma \Big[ u_F(s, \ell; \theta) + \big(\mathbf{1} + v_F(s, \ell; \theta)\big) \odot \big(M_\ell \nabla \log \pi_\ell(s)\big) \Big], \\ \Sigma_F(s, \ell; \theta) &= 2\gamma L_\ell \, \mathrm{diag}(\exp\{C_F \tanh(w_F(s, \ell; \theta))\}) \, L_\ell^\top. \end{aligned} \tag{32}$$

We initialize all three learned corrections $u_F$, $v_F$, and $w_F$ to zero. The initial forward kernel therefore coincides with preconditioned ULA,

$$q_F^{(\ell)}(s' \mid s) = \mathcal{N}(s' \mid s + \gamma M_\ell \nabla \log \pi_\ell(s), \, 2\gamma M_\ell).$$

We found this initialization to improve training stability on Sonar. Explicit preconditioning also incorporates off-diagonal correlations into the transition covariance, which cannot be represented by the diagonal covariance parameterization in §B.1.

**Ablation of replay and local search.** We additionally train the preconditioned $L = 5$ model without the replay buffer and MALA local search. In this variant, training uses only forward trajectories of fixed length, collected with the stopping classifier disabled. The learned stopping rule is used at inference, as in the full method. This experiment assesses the joint contribution of replay and local search to the resulting sampler.

Table 6: Comparison of training objectives for the *ULA w/ clf & corr* model on GMM9 ($d = 10$). All objectives use identical architectures, hyperparameters, and iteration budgets. Values are means and standard deviations. The best mean value for each metric is highlighted in gray.

| Target → | GMM9 ($d = 10$) | | | |
|---|---|---|---|---|
| Loss function ↓ Metric → | SD ↓ | MMD ↓ | $\mathbb{E}[n_\tau]$ ↓ | Wall-clock time ↓ |
| DB | 11.240±1.491 | 0.070±0.010 | 89.350±3.590 | 3.324±0.014 |
| SubTB | 5.752±2.832 | 0.043±0.013 | 148.201±5.131 | 5.825±0.023 |
| **Prefix-TB** | 7.301±1.702 | 0.056±0.011 | 98.437±6.042 | 3.320±0.027 |

**Results.** Table 5 shows that the multilevel construction improves the ELBO relative to the single-level model. The unpreconditioned $L = 2$ model achieves a higher mean ELBO than PIS, while the $L = 5$ model outperforms both diffusion baselines in this metric. Explicit preconditioning further improves the mean ELBO, with a modest increase in mean trajectory length.

The preconditioned model trained without replay and local search also achieves a higher mean ELBO than PIS and DDS. Thus, competitive performance is possible without these auxiliary training mechanisms, although the full training procedure achieves a higher mean ELBO in this experiment.

### B.7 Ablation of a training objective

We compare detailed balance (DB), subtrajectory balance (SubTB), and prefix trajectory balance (Prefix-TB) on GMM9 ($d = 10$), using the *ULA w/ clf & corr* configuration. All objectives use identical model architectures, hyperparameters, and iteration budgets. We set the learning rate to $3 \times 10^{-4}$, the step size to $\gamma = 0.1$, the flow regularization coefficient to $\rho = 0.01$, and the training trajectory length to $N_{\max} = 100$. Here, $N_{\max}$ specifies the rollout length during training, while inference follows the learned stopping rule.

In addition to SD, MMD, and mean trajectory length, we report wall-clock time aggregated over training iterations. The results are presented in Table 6.

SubTB achieves the lowest mean SD and MMD, but also produces the longest trajectories and incurs the highest training time. Its full objective evaluates balance constraints over all subtrajectories, whose number grows quadratically with the training trajectory length. In this experiment, its reported wall-clock time is approximately 75% higher than that of Prefix-TB.

Prefix-TB improves both mean sample-quality metrics relative to DB at essentially the same training time, with a moderate increase in mean trajectory length. Its number of balance constraints grows linearly with the training trajectory length. These results support its use as a compromise between approximation quality, sampling length, and training cost.

### B.8 Metrics

Let $\pi(x) = r(x)/Z$ denote the target distribution, and let $p_T$ be the marginal distribution of terminal states produced by the model. Following [7], to assess the discrepancy between $p_T$ and $\pi$, we report the following sample-based metrics that are widely used in the evaluation of generative samplers. Additionally, to measure performance of our method we report ELBO and EUBO [7]. For these metrics we utilize the implementation from `https://github.com/DenisBless/variational_sampling_methods`. Furthermore, we calculate the mean trajectory length of our sampler and denote this quantity by $\overline{n_\tau}$.

#### Maximum Mean Discrepancy

Maximum Mean Discrepancy [14, 7] is a kernel-based integral probability metric that compares two distributions through their embeddings in a reproducing kernel Hilbert space (RKHS). Let $\mathcal{H}_k$ be the RKHS associated with a positive definite kernel $k$. The MMD between $p_T$ and $\pi$ is defined as

$$\mathrm{MMD}_k(p_T, \pi) = \sup_{\substack{f \in \mathcal{H}_k \\ \|f\|_{\mathcal{H}_k} \le 1}} \left( \mathbb{E}_{x \sim p_T}[f(x)] - \mathbb{E}_{y \sim \pi}[f(y)] \right). \tag{33}$$

For characteristic kernels, this quantity is nonnegative and vanishes if and only if $p_T = \pi$. Given samples $X = \{x_i\}_{i=1}^n \sim p_T$ and $Y = \{y_j\}_{j=1}^m \sim \pi$, we use the usual unbiased estimator of the squared

MMD:

$$\widehat{\mathrm{MMD}}_k^2(X,Y) = \frac{1}{n(n-1)}\sum_{i\neq j} k(x_i,x_j) + \frac{1}{m(m-1)}\sum_{i\neq j} k(y_i,y_j) - \frac{2}{nm}\sum_{i=1}^{n}\sum_{j=1}^{m} k(x_i,y_j). \tag{34}$$

In our experiments, the squared exponential kernel

$$k(x,y) = \exp\left(-\frac{\|x-y\|_2^2}{\alpha}\right) \tag{35}$$

is used with bandwidth $\alpha$ selected by the median heuristic [7].

**Entropic optimal transport distance**

As a complementary metric, we use an entropy-regularized optimal transport distance [33, 7] based on the quadratic cost. The underlying unregularized quantity is the 2-Wasserstein distance,

$$W_{2,\varepsilon}(p_T,\pi) = \inf_{\xi\in\Gamma(p_T,\pi)}\left(\int_{\mathbb{R}^d\times\mathbb{R}^d}\|x-y\|_2^2\,\xi(dx,dy) - \epsilon\mathcal{H}(\xi)\right)^{1/2}, \tag{36}$$

where $\Gamma(p_T,\pi)$ denotes the set of couplings with marginals $p_T$ and $\pi$, and $\mathcal{H}(\xi) = -\int_{\mathbb{R}^d\times\mathbb{R}^d}\xi(x,y)\log\xi(x,y)\,\mathrm{d}x\mathrm{d}y$. In practice, we compute its entropically regularized counterpart using Sinkhorn iterations, which is substantially more tractable numerically. Throughout all experiments, we set $\epsilon = 10^{-3}$ [7].

**Evidence lower and upper bounds**

For a given terminal point $x\in\mathbb{R}^d$, let $\mathcal{T}_x = \{\tau\in\mathcal{T} \mid s_{n_\tau} = x\}$. For $\tau\in\mathcal{T}_x$ one can write trajectory distributions induced by the transition densities $p_F$ and $p_B$ from §3.1 as $p_F(\tau)$ and $p_B(\tau) = p_B(\tau \mid s_{n_\tau} = x)\pi(x)$. then we have:

$$\begin{aligned} Z &= \int_{\mathbb{R}^d} r(x)dx = \int_{\mathbb{R}^d} r(x)\left(\int_{\mathcal{T}_x} p_B(\tau \mid s_{n_\tau}=x)d\tau\right)dx \\ &= \int_{\mathcal{T}} r(x)p_B(\tau \mid s_{n_\tau}=x)d\tau = \mathbb{E}_{\tau\sim p_F}\left[\frac{r(x)p_B(\tau \mid s_{n_\tau}=x)}{p_F(\tau)}\right]. \end{aligned} \tag{37}$$

One way to construct an estimator for $\log Z$ is to use the evidence lower-bound (ELBO) [7]:

$$\log Z = \log\mathbb{E}_{\tau\sim p_F}\left[\frac{r(x)p_B(\tau \mid s_{n_\tau}=x)}{p_F(\tau)}\right] \geq \mathbb{E}_{\tau\sim p_F}\left[\log r(x) + \log p_B(\tau \mid s_{n_\tau}=x) - \log p_F(\tau)\right]. \tag{38}$$

To estimate the ELBO, we draw $M = 2000$ samples from the current policy and compute the average of the estimated $\log Z$ values:

$$\mathrm{ELBO} = \frac{1}{M}\sum_{i=1}^{M}\left[\log R(x^i) + \log p_B(\tau^i \mid s_{n_\tau}=x^i) - \log p_F(\tau^i)\right], \quad \tau^i\sim p_F, \tau^i \rightsquigarrow x^i. \tag{39}$$

To measure mode coverage, evidence upper-bound (EUBO) metric [7] was introduced, but it requires samples from $\pi$:

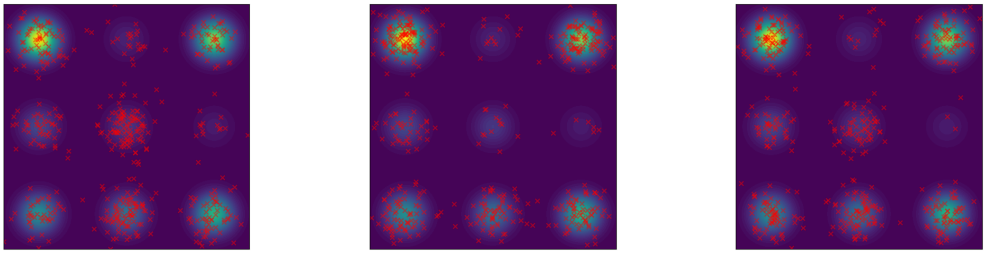
$$\begin{aligned} \log Z &= \mathbb{E}_{x\sim\pi,\tau\sim p_B(\cdot\mid s_{n_\tau}=x)}\left[\log\frac{r(x)p_B(\tau \mid s_{n_\tau}=x)}{p_F(\tau)}\right] \\ &\quad - \mathbb{KL}(\pi\cdot p_B \mid\mid p_F) \leq \mathbb{E}_{x\sim\pi,\tau\sim p_B(\cdot\mid x)}\left[\log\frac{r(x)p_B(\tau \mid s_{n_\tau}=x)}{p_F(\tau)}\right]. \end{aligned} \tag{40}$$

We draw $M = 2000$ samples from $\pi$ and then simulate trajectories using a backward policy:

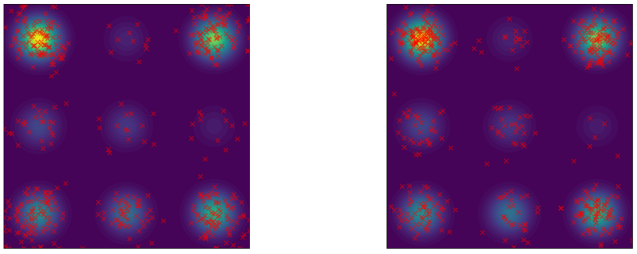
$$\mathrm{EUBO} = \frac{1}{M}\sum_{i=1}^{M}\left[\log R(x^i) + \log p_B(\tau^i \mid s_{n_\tau}=x^i) - \log p_F(\tau^i)\right], \quad x^i\sim\pi, \tau^i\sim p_B(\cdot \mid s_{n_\tau}=x^i). \tag{41}$$

### B.9 Additional visualizations

We provide qualitative sample visualizations for all environments studied in the experiments in §4. For each environment, we show samples generated by ULA, ULA RG, ULA Geweke, ULA with a learned classifier, and the full model.

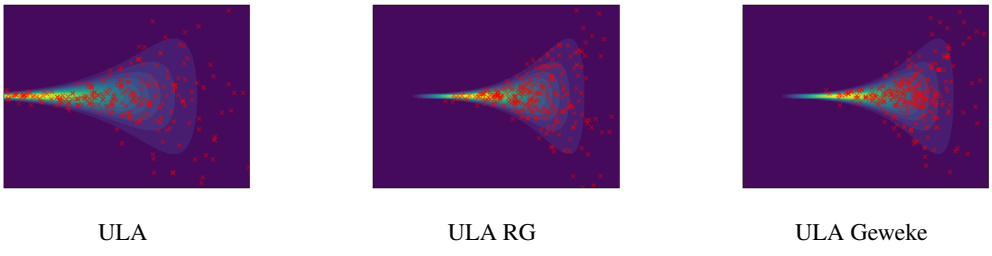


ULA ULA RG ULA Geweke

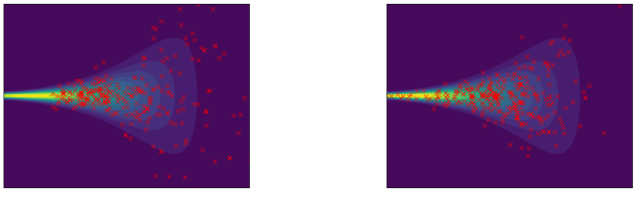


ULA + clf ULA w/ clf & corr

Figure 5: Samples on 9GMM ($d = 2$) with step-size $\gamma = 0.1$.

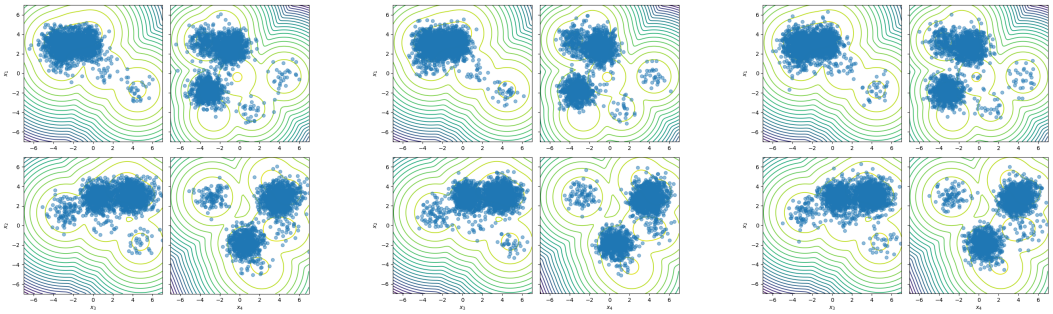


ULA ULA RG ULA Geweke

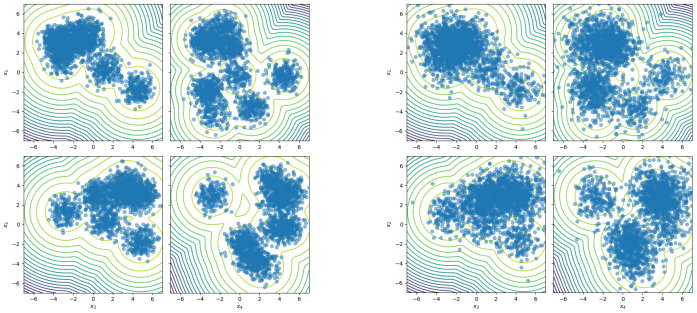


ULA + clf ULA w/ clf & corr

Figure 6: Samples on Funnel ($d = 2$) with step-size $\gamma = 0.01$.

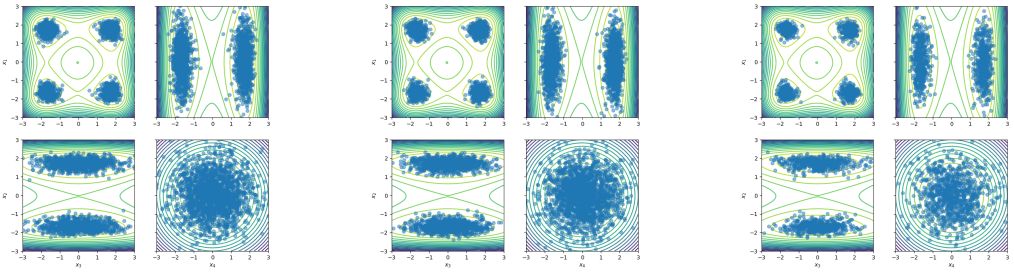


ULA ULA RG ULA Geweke

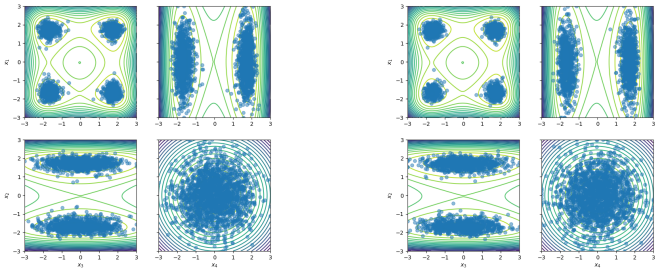


ULA + clf ULA w/ clf & corr

Figure 7: Samples on GMM9 ($d = 40$) using marginal projections, step-size $\gamma = 0.1$.



ULA ULA RG ULA Geweke



ULA + clf ULA w/ clf & corr

Figure 8: Marginal projections of samples from ManyWell ($d = 40$), step-size $\gamma = 0.1$.

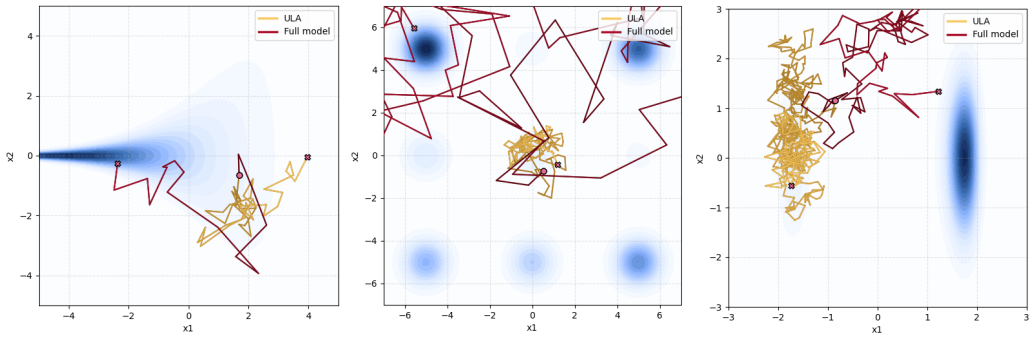



Figure 9: Comparison of trajectories produced by ULA and the full model in different environments (from left to right: Funnel, GMM9, ManyWell). ULA tends to remain trapped within a single mode or local region of the target and therefore requires a large mixing time to move between modes. In contrast, the proposed model is able to make effective transitions between modes and more distant high-density regions, resulting in substantially more efficient exploration of the target distribution.

## C Classical stopping diagnostics for MCMC

For $m$ chains of length $n$, let $\bar{x}_j$ be the mean of chain $j$, let

$$\bar{x} = \frac{1}{m}\sum_{j=1}^{m} \bar{x}_j$$

denote the overall mean across chains, and let $s_j^2$ be the within-chain variance. The Gelman–Rubin potential scale reduction factor is based on

$$B = \frac{n}{m-1}\sum_{j=1}^{m}(\bar{x}_j - \bar{x})^2, \qquad W = \frac{1}{m}\sum_{j=1}^{m} s_j^2, \tag{42}$$

and

$$\hat{R} = \sqrt{\frac{\hat{V}}{W}}, \qquad \hat{V} = \frac{n-1}{n}W + \frac{1}{n}B. \tag{43}$$

Values of $\hat{R}$ close to one indicate approximate agreement between between-chain and within-chain variability.

Another commonly used quantity is the effective sample size,

$$\mathrm{ESS} = \frac{N}{1 + 2\sum_{k\geq 1}\rho_k}, \tag{44}$$

where $N$ is the total number of retained samples and $\rho_k$ denotes the lag-$k$ autocorrelation.

A related diagnostic is Geweke's statistic,

$$Z = \frac{\bar{g}_A - \bar{g}_B}{\sqrt{\hat{S}_A(0)/n_A + \hat{S}_B(0)/n_B}}, \tag{45}$$

which compares the averages of a scalar summary over an early window $A$ and a late window $B$, with $\hat{S}_A(0)$ and $\hat{S}_B(0)$ denoting corresponding long-run variance estimates.

## D Interpretation of Theorem 3.6

In this section we discuss and interpret results of Theorem 3.6. At first, let us define useful objects for this matter. Following [8], given a flow measure $F$ from Def. 3.2, we define an edge flow

$$F(A \times B) = \int_A F(\mathrm{d}s) P_F(s, B), \tag{46}$$

for any measurable $A \subseteq \bar{\mathcal{S}} \setminus \{s_f\}, B \subseteq \bar{\mathcal{S}} \setminus \{s_0\}$. Then using an edge flow we can define an incoming and outcoming flow measures $F_{\text{in}}$ and $F_{\text{out}}$, respectively:

$$\begin{aligned} F_{\text{in}}(B) &= F(\bar{\mathcal{S}} \setminus \{s_f\} \times B), \\ F_{\text{out}}(A) &= F(A \times \bar{\mathcal{S}} \setminus \{s_0\}). \end{aligned} \tag{47}$$

Consider a measurable set $A \subseteq \mathbb{R}^d$. Then from the definitions of transitions in our state space

$$\begin{aligned} P_F(s_0, A) &= p_0(A), \\ P_F(s, A) &= (1 - d_F(s))Q_F(s, A) + d_F(s)\delta_{s_f}(A), \end{aligned} \tag{48}$$

we obtain

$$\begin{aligned} F_{\text{in}}(A) &= \int_{\mathcal{S} \cup \{s_0\}} F(\mathrm{d}s) P_F(s, B) = F(\{s_0\}) p_0(A) + \int_{\mathbb{R}^d} F(\mathrm{d}s)(1 - d_F(s)) Q_F(s, A) \\ &= Z p_0(A) + \int_{\mathbb{R}^d} (F(\mathrm{d}s) - R(\mathrm{d}s)) Q_F(s, A) = Z p_0(A) - (F - R) Q_F(A), \end{aligned} \tag{49}$$

where we used that from Proposition 3.3 the flow measure $F$ and the Markov kernel $P_F$ satisfy flow-matching conditions (3). Therefore, detailed balance conditions (5) must hold and we can express a classifier $d_F$ from a relation (12). In turn, a flow measure we defined in Def. 3.2 coincides with the outcoming flow measure $F_{\text{out}}$:

$$F_{\text{out}}(A) = \int_A F(\mathrm{d}s) P_F(s, \bar{S} \setminus \{s_0\}) = \int_A F(\mathrm{d}s) = F(A). \tag{50}$$

In Proof of the Theorem 3.6 we show that detailed balance conditions for continuous transitions imply a Poisson equation:

$$F(I - Q_F)(A) = Z p_0(A), \tag{51}$$

for a measurable set $A \subseteq \mathbb{R}^d$. By substituting $F_{\text{out}} = F$ and $F_{\text{in}} = (F_{\text{out}} - R)Q_F + Z p_0$, then

$$(Z p_0 - R Q_F)(A) = F_{\text{in}}(A) - F_{\text{out}} Q_F(A). \tag{52}$$

The left hand side is exactly the unnormalized term in the series of the measure $U$. Therefore, we can interpret $U(A)$ as an accumulated discrepancy between injected flow $F_{\text{in}}(A)$ and removed flow $F_{\text{out}} Q_F(A)$, when we applied a kernel $Q_F$.

# E Proofs of results in the main text

## E.1 Derivation of flow representations

In this section, we derive key relations from the detailed balance conditions, which implies flow-matching as shown in [22]. Since this implication does not depend on acyclicity, it remains valid in our non-acyclic setting. Moreover, since all relevant measures and kernels in our framework admit densities with respect to the chosen reference measure, we may express these conditions in density form. Applying detailed balance for transitions from $s_0$ to $s \in \mathbb{R}^d$, yields

$$f(s_0)\, p_F(s \mid s_0) = f(s)\, p_B(s_0 \mid s). \tag{53}$$

Substituting $p_F(s \mid s_0) = p_0(s)$, $f(s_0) = Z$, and $p_B(s_0 \mid s) = d_B(s)$, we obtain

$$f(s) = \frac{Z\, p_0(s)}{d_B(s)}. \tag{54}$$

Applying detailed balance to transitions from $s \in \mathbb{R}^d$ to $s_f$, we obtain

$$f(s)\, p_F(s_f \mid s) = f(s_f)\, p_B(s \mid s_f). \tag{55}$$

Substituting $p_F(s_f \mid s) = d_F(s)$, $f(s_f) = Z$, and $p_B(s \mid s_f) = r(s)/Z$, we arrive at the alternative representation

$$f(s) = \frac{r(s)}{d_F(s)}. \tag{56}$$

Therefore, we have the following connection between classifiers:

$$d_F(s) = \frac{\pi(s)}{p_0(s)} d_B(s). \tag{57}$$

### E.2 Proof of Proposition 3.3.

From Def. 3.2, we have

$$\begin{aligned} F(\{s_0\}) &= F(\{s_f\}) = Z, \\ F(A) &= F(\{s_0\}) \sum_{n=0}^{\infty} P_F^n(s_0, A), \qquad A \subseteq \bar{\mathcal{S}} \setminus \{s_0, s_f\}. \end{aligned} \tag{58}$$

We aim to prove the flow-matching condition (3), namely that for any measurable set $A \subseteq \bar{\mathcal{S}} \setminus \{s_0\}$,

$$F(A) = \int_{\bar{\mathcal{S}}\setminus\{s_f\}} F(\mathrm{d}s)\, P_F(s, A). \tag{59}$$

Case $A \subseteq \mathbb{R}^d$. Using the definition of $F$, we obtain

$$\begin{aligned} \int_{\bar{\mathcal{S}}\setminus\{s_f\}} F(\mathrm{d}s)\, P_F(s, A) &= F(\{s_0\}) \int_{\bar{\mathcal{S}}\setminus\{s_f\}} \sum_{n=0}^{\infty} P_F^n(s_0, \mathrm{d}s)\, P_F(s, A) \\ &\overset{(a)}{=} F(\{s_0\}) \sum_{n=0}^{\infty} \int_{\bar{\mathcal{S}}\setminus\{s_f\}} P_F^n(s_0, \mathrm{d}s)\, P_F(s, A) \\ &\overset{(b)}{=} F(\{s_0\}) \sum_{n=0}^{\infty} P_F^{n+1}(s_0, A) \\ &= F(A) - F(\{s_0\})\delta_{s_0}(A) \\ &= F(A). \end{aligned} \tag{60}$$

In step (a), we interchange the integral and the series using Tonelli's theorem. In step (b), we apply the Chapman–Kolmogorov relation:

$$P_F^{n+1}(s_0, A) = \int_{\bar{\mathcal{S}}\setminus\{s_f\}} P_F^n(s_0, \mathrm{d}s)\, P_F(s, A), \tag{61}$$

because for $A \subseteq \mathbb{R}^d$, we have $P_F(s_f, A) = 0$. In the last step, noting that $P_F^0(s_0, A) = \delta_{s_0}(A) = 0$ for $A \subseteq \mathbb{R}^d$ concludes the claim for this case.

Case $A = \{s_f\}$. In this case, the absorbing nature of $s_f$ implies

$$\begin{aligned} P_F^{n+1}(s_0, \{s_f\}) &= \int_{\bar{\mathcal{S}}} P_F^n(s_0, \mathrm{d}s)\, P_F(s, \{s_f\}) \\ &= \int_{\bar{\mathcal{S}}\setminus\{s_f\}} P_F^n(s_0, \mathrm{d}s)\, P_F(s, \{s_f\}) + P_F^n(s_0, \{s_f\}), \end{aligned} \tag{62}$$

since $P_F(s_f, \{s_f\}) = 1$. Repeating the steps from the previous case and using the above equation, yields:

$$\begin{aligned} \int_{\bar{\mathcal{S}}\setminus\{s_f\}} F(\mathrm{d}s)\, P_F(s, \{s_f\}) &= F(\{s_0\}) \sum_{n=0}^{\infty} \left(P_F^{n+1}(s_0, \{s_f\}) - P_F^n(s_0, \{s_f\})\right) \\ &= F(\{s_0\}) \Big(\delta_{s_f}(\{s_f\}) - \delta_{s_0}(\{s_f\})\Big) \\ &= F(\{s_0\}) = F(\{s_f\}), \end{aligned} \tag{63}$$

where under Assumption 3.1, we have $P_F^n(s_0, \cdot) \to \delta_{s_f}$. Combining these two cases, we conclude that the flow-matching conditions hold for all measurable sets $A \subseteq \bar{\mathcal{S}} \setminus \{s_0\}$. □

### E.3 Proof of Proposition 3.5.

$$\mathbb{E}_{\tau\sim P}[n_\tau] = \mathbb{E}_{\tau\sim P}\left[\sum_{n=0}^{\infty} \mathbb{1}\{s_n \in \mathcal{S}\}\right] = \sum_{n=0}^{\infty} \mathbb{P}(s_n \in \mathcal{S}) = \sum_{n=0}^{\infty} P_F^n(s_0, \mathcal{S}) = \frac{F(\mathcal{S})}{F(\{s_0\})}. \tag{64}$$

Note that under Assumption 3.1 the series above converges. □

### E.4 Proof of Theorem 3.6 and Corollary 3.7

**Proof of Theorem 3.6.** From the detailed balance conditions at the source and sink states, we obtain for $s \in \mathbb{R}^d$ that:

$$\begin{aligned} f(s)d_B(s) &= Z\, p_0(s), \\ f(s)d_F(s) &= r(s). \end{aligned} \tag{65}$$

Now consider the detailed balance relation in the interior for $s, s' \in \mathbb{R}^d$:

$$f(s)\big(1 - d_F(s)\big)q_F(s' \mid s) = f(s')\big(1 - d_B(s')\big)q_B(s \mid s'). \tag{66}$$

Integrating both sides with respect to $s$, and using that $q_B(\cdot \mid s')$ is a probability density, yields:

$$\int_{\mathbb{R}^d} f(s)\big(1 - d_F(s)\big)q_F(s' \mid s)\, ds = f(s')\big(1 - d_B(s')\big). \tag{67}$$

Expanding both sides and using (65), gives the following:

$$\int_{\mathbb{R}^d} f(s)q_F(s' \mid s)\, ds - \int_{\mathbb{R}^d} r(s)q_F(s' \mid s)\, ds = f(s') - Z\, p_0(s'). \tag{68}$$

For the purpose of further proof, it will be convenient to write the above equation in a measure-kernel form. Define $V(A) := F(A)/Z$ for $A \subseteq \mathbb{R}^d$, and let $\pi$, $p_0$ and $\pi_Q$ denote the measures of the corresponding densities. Dividing the above equation by $Z$, we obtain the Poisson equation:

$$V(I - Q_F) = p_0 - \pi Q_F. \tag{69}$$

Since $Q_F$ is uniformly geometrically ergodic with a unique stationary measure $\pi_Q$, there exist constants $C < \infty$ and $\rho \in (0, 1)$ such that

$$\|\nu\, Q_F^n - \pi_Q\|_{\mathrm{TV}} \le C\rho^n, \tag{70}$$

for all $n \in \mathbb{N}$ and any probability measure $\nu$, where $\|\cdot\|_{\mathrm{TV}}$ is a total variation norm. Define a measure

$$U := \sum_{n=0}^{\infty} (p_0 Q_F^n - \pi Q_F^{n+1}), \tag{71}$$

where the series converges absolutely in total variation using (70) and a triangular inequality. Then by telescoping

$$U - UQ_F = \sum_{n=0}^{\infty} (p_0 Q_F^n - \pi Q_F^{n+1}) - \sum_{n=0}^{\infty} (p_0 Q_F^{n+1} - \pi Q_F^{n+2}) = p_0 - \pi Q_F, \tag{72}$$

which means that a measure $U$ is a solution to a Poisson equation (69). Suppose that $V$ is another solution to (69), then

$$(V - U)(I - Q_F) = 0, \tag{73}$$

which implies that $V - U$ is an invariant measure for the kernel $Q_F$. By the assumption that $Q_F$ admits a unique stationary measure $\pi_Q$, it follows that $V - U$ lies in the one-dimensional subspace spanned by $\pi_Q$. Therefore, all solutions of (69) are of the form

$$V = \sum_{n=0}^{\infty} \big(p_0 Q_F^n - \pi Q_F^{n+1}\big) + n_Q\, \pi_Q, \qquad n_Q \in \mathbb{R}. \tag{74}$$

Since $q_F(\cdot \mid s)$ is a transition density, the pushforward of any absolutely continuous measure remains absolutely continuous. Hence $V$ admits a density of the form

$$v(s) = u(s) + n_Q\, \pi_Q(s), \qquad n_Q \in \mathbb{R}, \tag{75}$$

where by $u(s)$ we denote the corresponding density of a measure $U$. Using the connection of a flow and a forward classifier in (65), we obtain

$$d_F(s) = \frac{r(s)}{f(s)} = \frac{\pi(s)}{v(s)} = \frac{\pi(s)}{u(s) + n_Q\, \pi_Q(s)}. \tag{76}$$

Similarly for the backward classifier from (12)

$$d_B(s) = \frac{p_0(s)}{\pi(s)} d_F(s) = \frac{p_0(s)}{u(s) + n_Q\, \pi_Q(s)}. \tag{77}$$

Thus, in order for both classifiers to define valid probabilities, it is necessary and sufficient that

$$u(s) + n_Q\, \pi_Q(s) \geq \max\{\pi(s), p_0(s)\}, \qquad \forall s \in \mathbb{R}^d. \tag{78}$$

The admissible values of $n_Q$ are

$$n_Q \geq n_Q^* = \sup_{s \in \mathbb{R}^d} \frac{\max\{\pi(s), p_0(s)\} - u(s)}{\pi_Q(s)}. \tag{79}$$

Finally, from the balance for the interior transitions

$$q_B(s \mid s') = \frac{(v(s) - \pi(s)) q_F(s' \mid s)}{v(s') - p_0(s')}. \tag{80}$$

If the detailed balance conditions are satisfied and Assumption 3.1 holds, it follows that $n_Q$ and $n_Q^*$ are finite.

Conversely, given a solution $V$ to (69) with admissible values of $n_Q$, detailed balance conditions for a source-to-continuous and continuous-to-sink transitions are satisfied by setting $d_F(s) := \pi(s)/v(s)$ and $d_B(s) := p_0(s)/v(s)$. From (80) balance conditions for the interior transitions are satisfied and by integrating over $\mathbb{R}^d$ we conclude that $q_B$ defines a valid transition density

$$\int_{\mathbb{R}^d} q_B(s \mid s')\mathrm{d}s = \frac{\int_{\mathbb{R}^d} (v(s) - \pi(s)) q_F(s' \mid s)\mathrm{d}s}{v(s') - p_0(s')} = \frac{v(s') - p_0(s')}{v(s') - p_0(s')} = 1, \tag{81}$$

where we used that $(V - \pi)Q_F = V - p_0$, since $V$ is a solution to (69). Hence, from $V$ we can uniquely recover all other objects to satisfy detailed balance conditions. □

**Proof of Corollary 3.7.** Notice that from Proposition 3.5

$$\mathbb{E}[n_\tau] = \frac{F(\mathcal{S})}{F(\{s_0\})} = V(\mathcal{S}) = U(\mathcal{S}) + n_Q \pi_Q(\mathcal{S}) = n_Q, \tag{82}$$

because we are dealing with probability measures and $\mathcal{S} = \mathbb{R}^d$. Finally, minimizing the total flow is equivalent to minimizing

$$\int_{\mathbb{R}^d} f(s)\, ds = Z \int_{\mathbb{R}^d} v(s)\, ds = Z \int_{\mathbb{R}^d} \big(u(s) + n_Q\, \pi_Q(s)\big)\, ds. \tag{83}$$

Since $\pi_Q$ is a probability density, this quantity is affine and strictly increasing in $n_Q$. Therefore, the minimum is attained at the smallest admissible value of $n_Q = n_Q^*$.

## NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction accurately describe the proposed classifier-based adaptive stopping framework, the theoretical connection to non-acyclic GFlowNets, and the empirical improvements in trajectory length and sampling quality. The main contributions are summarized in Section 1.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Yes, see Section 5.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: For each theoretical result we provide the proof in Appendix E.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: The paper specifies model parameterizations, optimization procedures, training objectives, trajectory generation procedures, replay-buffer usage, evaluation metrics, and environment definitions necessary for reproducing the reported experiments. See Appendix B.1, Appendix B.2, and Appendix B.3.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: Code is publicly available at github.com/kkorolev1/stop-the-sampler. A detailed explanation of the experimental setup can be found in Appendix B.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The paper describes optimization settings, replay strategies, local-search procedures, neural architectures, parameterizations, training schedules, and evaluation metrics. Environment-specific hyperparameters are provided in Appendix B.3.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Yes, all tables with experimental results provide error bars computed across multiple random seeds.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Yes, see Appendix B.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: To the best of our knowledge, we follow the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: The authors believe that the assets of the submission do not pose societal impacts.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.

- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The authors believe that the assets of the submission do not pose such risks.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All utilized code and data is free to use in academic research and is credited where required.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: Code is publicly available at github.com/kkorolev1/stop-the-sampler. A detailed explanation of the experimental setup can be found in Appendix B.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: No crowdsourcing nor research with human subjects was involved in this work.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: No crowdsourcing nor research with human subjects was involved in this work.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.