Generative Flow Networks for Discrete Probabilistic Modeling

Dinghuai Zhang [1] **Nikolay Malkin** [1] **Zhen Liu** [1] **Alexandra Volokhova** [1] **Aaron Courville** [1] **Yoshua Bengio** [1]

Abstract

We present energy-based generative flow networks (EB-GFN), a novel probabilistic modeling algorithm for high-dimensional discrete data. Building upon the theory of generative flow networks (GFlowNets; Bengio et al., 2021b), we model the generation process by a stochastic data construction policy and thus amortize expensive MCMC exploration into a fixed number of actions sampled from a GFlowNet. We show how GFlowNets can approximately perform large-block Gibbs sampling to mix between modes. We propose a framework to jointly train a GFlowNet with an energy function, so that the GFlowNet learns to sample from the energy distribution, while the energy learns with an approximate MLE objective with negative samples from the GFlowNet. We demonstrate EB-GFN's effectiveness on various probabilistic modeling tasks. Code is publicly available at `github.com/zdhNarsil/EB_GFN`.

1. Introduction

Probabilistic modeling in discrete spaces, especially those with compositional structure, is important due to the universality of applications of discrete data structures, such as in natural language processing (Tai et al., 2015) or in symbolic reasoning (Besold et al., 2017). However, distributions in high-dimensional discrete spaces are generally hard to model, as they may feature rapid combinatorial growth of modes. These modes can be well separated from each other, which poses a challenge for Markov Chain Monte Carlo (MCMC) methods (Salakhutdinov, 2009). Mixing between modes is generally slow without a priori knowledge of the specific latent structure of the distribution.

Furthermore, generative modeling methods such as energy-based models (EBMs; LeCun et al., 2006) also suffer from these mixing issues when generating negative samples with MCMC (Tieleman & Hinton, 2009). The incapability of MCMC to capture the energy landscape results in the spurious mode problem (Desjardins et al., 2010; Bengio et al., 2013), in which new modes that do not occur in the true data distribution appear in the learned energy distribution.

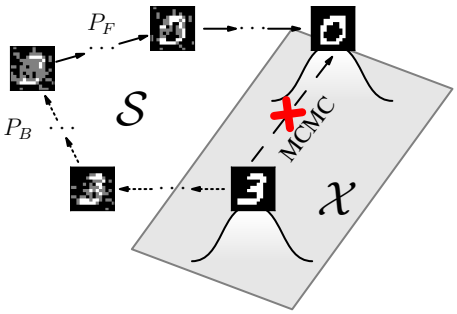

Figure 1. **The EB-GFN framework for learning an energy function on a discrete space** $\mathcal{X}$, in this case the set of all binary images, to maximize likelihood of a dataset (*e.g.*, MNIST). An energy landscape with well-separated modes is difficult to explore with local search methods. A GFlowNet, parametrized with a pair of stochastic policies P_F (painting) and P_B (erasure), exploits a broader state space $\mathcal{S}$ with partial choices, thus aiding the exploration of $\mathcal{X}$ that is necessary for updating the energy function with contrastive-divergence-like objectives. The trained GFlowNet is also a generative model on its own; see Figs. 2 and 6.

In this paper, we propose to take advantage of generative flow networks, or GFlowNets (Bengio et al., 2021a;b), instead of MCMC methods in order to simultaneously learn a sampler and train an energy function from data. GFlowNets are generative models for compositional objects, *i.e.*, they learn a stochastic policy that iteratively constructs the sampled object through a sequence of simpler steps. In past work, GFlowNets were trained to query a given energy function, rather than from a dataset (like typical generative models). We propose to adapt the GFlowNet methodology to the problem of learning from data, and jointly train the GFlowNet sampler and the energy function.

Because a GFlowNet can *learn from the low-energy configurations it has already encountered*, it has a chance to guess and sample yet-unvisited modes if there are learnable regularities in the underlying data distribution. The compositional generative structure of GFlowNet further enhances its ability to discover regularities and thus jump between

[1]Mila - Quebec AI Institute and Université de Montréal, Montreal, Quebec, Canada. Correspondence to: Dinghuai Zhang <dinghuai.zhang@mila.quebec>.

Proceedings of the 39^{th} International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022.

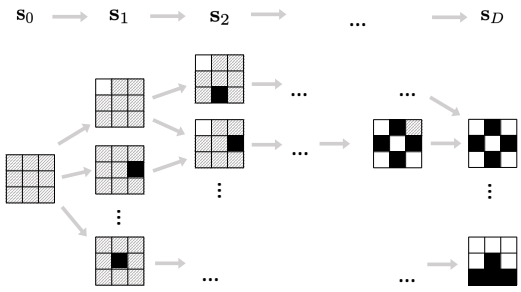

Figure 2. The state space $\mathcal{S}$ and the GFlowNet's forward modeling process in a 9-dimensional discrete data space. The states are the vertices of a DAG whose edges are the transitions – actions of painting a grey pixel into black (1) or white (0).

modes without having to travel long low-probability paths in between (Fig. 1). Instead of having to exploit a priori structure to design long jumps in the MCMC, GFlowNets can discover that structure when it is not explicitly known.

This paper makes the following key contributions:

(1) We cast a non-autoregressive sequential generation model for high-dimensional discrete data as a generative flow network (§3.1).

(2) We introduce a GFlowNet-based MCMC proposal enabling efficient large jumps with low probability of rejection, taking advantage of the compositional structure learned by the GFlowNet generative policy (§3.3).

(3) We describe a procedure based on such proposals for jointly training the GFlowNet sampler with an energy model given a dataset and state conditions under which this estimates the true data log-likelihood gradient with respect to the energy function's parameters (§3.2, §3.3).

(4) We test the algorithm on a variety of synthetic and real tasks, achieving competitive results (§4).

2. Preliminaries

2.1. GFlowNets

Generative flow networks, or GFlowNets for short (Bengio et al., 2021a;b), are trainable generative policies on which this paper is built. They model the generation process of objects $\mathbf{x} \in \mathcal{X}$ by a sequence of discrete *actions* that incrementally modify a partially constructed object (*state*). The space of possible action sequences is represented by a directed acyclic graph (DAG, see Fig. 2) $G = (\mathcal{S}, \mathcal{A})$, where the vertices in $\mathcal{S}$ are states and the edges in $\mathcal{A}$ are actions that modify one state to another. We use the terms *parents* and *children*, respectively, for the tails of incoming edges and the heads of outgoing edges of a state. Generation begins at a special *initial state* $\mathbf{s}_0$ and terminates upon a transition to any *terminal state*, which is a state with no outgoing actions. The set of terminal states is identified with the target space $\mathcal{X}$. Note that multiple possible action sequences may lead to the same terminal state.

A *complete trajectory* is a sequence of states $\mathbf{s}_0 \to \mathbf{s}_1 \to \ldots \to \mathbf{s}_n$, where each transition $\mathbf{s}_t \to \mathbf{s}_{t+1}$ is an action in $\mathcal{A}$ and $\mathbf{s}_n \in \mathcal{X}$ is terminal. A *trajectory flow* is a measure (unnormalized density) on the set of all complete trajectories $\mathcal{T}$, *i.e.*, a non-negative function $F : \mathcal{T} \to \mathbb{R}_{\geq 0}$, which can be thought of as a number of particles flowing from $\mathbf{s}_0$ to terminal states along each route. The flow is called *Markovian* if there exist distributions $P_F(\cdot|\mathbf{s})$ over the children of every non-terminal state $\mathbf{s}$, and a constant Z, such that for any complete trajectory $\tau = (\mathbf{s}_0 \to \mathbf{s}_1 \to \ldots \to \ldots \to \mathbf{s}_n)$, we have $P_F(\tau) = F(\tau)/Z$ with

$$P_F(\tau) = P_F(\mathbf{s}_1|\mathbf{s}_0)P_F(\mathbf{s}_2|\mathbf{s}_1)\ldots P_F(\mathbf{s}_n|\mathbf{s}_{n-1}). \quad (1)$$

In this case, $P_F(\mathbf{s}_{t+1}|\mathbf{s}_t)$ is called a *forward policy*, and can be used to sample complete trajectories from the density F and thus also their terminal states, objects $\mathbf{x} \in \mathcal{X}$.[1] We write $P_T(\mathbf{x})$ for the probability that a trajectory sampled from P_F terminates in $\mathbf{x}$.

Past work on GFlowNets (Bengio et al., 2021a;b; Malkin et al., 2022; Jain et al., 2022) has considered the problem of fitting a Markovian flow to a fixed *reward function* on $\mathcal{X}$. Given a non-negative reward function $R : \mathcal{X} \to \mathbb{R}_{\geq 0}$, one seeks a Markovian flow F such that the likelihood of a trajectory sampled from F terminating in a given $\mathbf{x} \in \mathcal{X}$ is proportional to $R(\mathbf{x})$, *i.e.*, $P_T(\mathbf{x}) \propto R(\mathbf{x})$. This is accomplished by the reward matching condition:

$$R(\mathbf{x}) = \sum_{\tau=(\mathbf{s}_0\to\ldots\to\mathbf{s}_n),\mathbf{s}_n=\mathbf{x}} F(\tau). \quad (2)$$

By Eq. (1), one can specify a Markovian flow by a learned scalar $\log Z_{\boldsymbol{\theta}}$ and a neural net with parameters $\boldsymbol{\theta}$ that outputs $P_F(\cdot|\mathbf{s};\boldsymbol{\theta})$ for any input state $\mathbf{s}$. Algorithms for learning P_F from maximum-entropy reinforcement learning (Haarnoja et al., 2017) would solve this problem if there was only one way to construct an object $\mathbf{x}$ (the DAG is a tree) while GFlowNets are applicable in the more general DAG setting (Bengio et al., 2021a). The solution originally proposed by Bengio et al. (2021a) recasts an *unnormalized* forward policy as a network flow on G (in the classical sense of Ford & Fulkerson (1956)) and performs gradient descent on the error in a flow conservation constraint at states sampled from a training policy, amounting to a generalization of temporal difference objectives from Sutton (2005).

Alternative objectives proposed by Bengio et al. (2021b); Malkin et al. (2022) require a model to produce 3 outputs: the scalar $\log Z_{\boldsymbol{\theta}}$, a forward policy $P_F(\cdot|\cdot;\boldsymbol{\theta})$, and a *backward policy* that produces distributions $P_B(\cdot|\mathbf{s};\boldsymbol{\theta})$ over the parents of an input state $\mathbf{s}$. (When action sequences are sampled in reverse from P_B, we will use dashed arrows $\mathbf{s}' \dashrightarrow \mathbf{s}$

[1] It is helpful to think of the particle analogy. For a Markovian flow, the distribution over complete trajectories ($P_F(\tau) \propto F(\tau)$) satisfies a history-independence property: a particle's choice of route after reaching a state $\mathbf{s}$ is independent of how it reached $\mathbf{s}$.

to indicate that the action $\mathbf{s}\to\mathbf{s}'$ has been sampled *against* the direction of the DAG edges. The $\boldsymbol{\theta}$ will be dropped from P_F and P_B notation when it does not cause ambiguity.)

In modeling tasks with a fixed reward (Bengio et al., 2021a; Malkin et al., 2022), only the forward policy P_F is used for sampling from the GFlowNet, and the backward policy P_B is simply an training artifact. However, our approach makes use of the backward policy to perform local exploration in a contrastive divergence-like algorithm (§3.3, Figure 1).

2.2. Energy-based models

Energy-based models (EBMs) (LeCun et al., 2006; Song & Kingma, 2021) are a popular approach for probabilistic inference and modeling. An EBM specifies a distribution over a space $\mathcal{X}$ by a density $p_{\boldsymbol{\phi}}(\mathbf{x}) = \frac{1}{Z_{\boldsymbol{\phi}}}\exp(-\mathcal{E}_{\boldsymbol{\phi}}(\mathbf{x}))$, where $\mathcal{E}_{\boldsymbol{\phi}}$ is the energy function, with model parameter $\boldsymbol{\phi}$, and $Z_{\boldsymbol{\phi}}$ is a normalizing constant independent of $\mathbf{x}$. The normalizing factor is not explicitly parametrized, but, for finite spaces, can theoretically be computed as $Z_{\boldsymbol{\phi}} = \sum_{\mathbf{x}\in\mathcal{X}}\exp(-\mathcal{E}_{\boldsymbol{\phi}}(\mathbf{x}))$. This summation is finite, and thus the energy function $\mathcal{E}_{\boldsymbol{\phi}}(\mathbf{x})$ can take an arbitrary form. However, this flexibility comes with a price: calculating $Z_{\boldsymbol{\phi}}$ can involve an exponentially large summation, making evaluation of the exact probability intractable. To avoid such expensive calculations, it is desirable to amortize sampling from such models by training a generative model.

We can train $\mathcal{E}_{\boldsymbol{\phi}}$ through maximum likelihood estimation (MLE), *i.e.* seeking $\arg\max_{\boldsymbol{\phi}}\mathbb{E}_{\mathbf{x}\sim p_{\text{data}}(\mathbf{x})}[\log p_{\boldsymbol{\phi}}(\mathbf{x})]$, where p_{data} is the empirical distribution associated with the training data. The gradient of negative log-likelihood (NLL) with respect to the model parameter $\boldsymbol{\phi}$ is given by

$$\begin{aligned}-\nabla_{\boldsymbol{\phi}}\log p_{\boldsymbol{\phi}}(\mathbf{x}) &= \nabla_{\boldsymbol{\phi}}\mathcal{E}_{\boldsymbol{\phi}}(\mathbf{x}) + \nabla_{\boldsymbol{\phi}}\log Z_{\boldsymbol{\phi}}\\ &= \nabla_{\boldsymbol{\phi}}\mathcal{E}_{\boldsymbol{\phi}}(\mathbf{x}) - \mathbb{E}_{\mathbf{x}'\sim p_{\boldsymbol{\phi}}(\mathbf{x}')}[\nabla_{\boldsymbol{\phi}}\mathcal{E}_{\boldsymbol{\phi}}(\mathbf{x}')].\end{aligned}\tag{3}$$

Evaluating or estimating the second term in (3) involves taking *negative samples* $\mathbf{x}'$ from the EBM distribution, which can require an expensive search. The classical Contrastive Divergence (CD) algorithm (Hinton, 2002) approximates this gradient update by changing the energy function parameter with the following stochastic approximation:

$$\mathbb{E}_{\mathbf{x}\sim p_{\text{data}}(\mathbf{x})}[\nabla_{\boldsymbol{\phi}}\mathcal{E}_{\boldsymbol{\phi}}(\mathbf{x}) - \mathbb{E}_{\mathbf{x}'\sim q_K(\mathbf{x}'|\mathbf{x})}\nabla_{\boldsymbol{\phi}}\mathcal{E}_{\boldsymbol{\phi}}(\mathbf{x}')]\tag{4}$$

where $q_K(\mathbf{x}'|\mathbf{x})$ is the distribution obtained by using a K-step MCMC initialized at $\mathbf{x}$ to *approximately* sample from $p_{\boldsymbol{\phi}}$. For example, $\mathbf{x}'$ can be taken from a K-step Metropolis-Hastings chain starting at a true data sample $\mathbf{x}\sim p_{\text{data}}(\mathbf{x})$, and the parameters updated with $\nabla_{\boldsymbol{\phi}}(\mathcal{E}_{\boldsymbol{\phi}}(\mathbf{x}) - \mathcal{E}_{\boldsymbol{\phi}}(\mathbf{x}'))$. As $K\to\infty$, assuming mixing of MCMC chains, the distribution over negative samples $q_K(\mathbf{x}'|\mathbf{x})$ converges to $p_{\boldsymbol{\phi}}(\mathbf{x}')$, and we recover, in expectation, the true gradient (3).

Tieleman (2008) later proposed persistent CD (PCD), where the MCMC chains that give negative samples do not restart at true data at every training step, but are initialized with a previous state. While CD-type algorithms are efficient in computation and can accelerate learning, their gradient estimation is biased and thus may not model the true data distribution faithfully (Nijkamp et al., 2020).

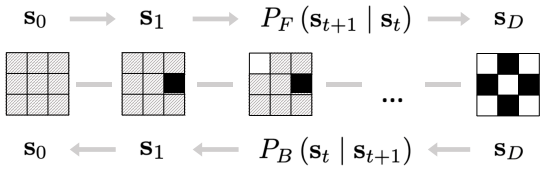

Figure 3. An illustration of the forward and backward GFlowNet policies in a 9-dimensional discrete space of the kind studied here. The forward policy transforms a state $\mathbf{s}_t$ into $\mathbf{s}_{t+1}$, while the backward policy does the opposite operation. We represent $0/1$ with black / white patches, and use grey patches to denote unspecified entries $\oslash$ in incomplete (non-terminal) states.

3. Methodology

3.1. GFlowNet generative process

In this work, we aim to model a target distribution over discrete data with a GFlowNet. In the domains we consider, the data is in D-dimensional binary space, *i.e.*, $\mathbf{x}\in\mathcal{X}\triangleq\{0,1\}^D$. As an example, $\mathbf{x}$ could be an image with D pixels taking binary values. We model the generation of vectors in $\mathcal{X}$ by a GFlowNet. The state space $\mathcal{S}$ of the GFlowNet consists of vectors of length d with entries in $\{0,1,\oslash\}$, where the void symbol $\oslash$ represents a yet unspecified entry that may be turned to 0 or 1 by a future action. To be precise:

$$\mathcal{S}\triangleq\{(\mathbf{s}^1,\dots,\mathbf{s}^D)\mid\mathbf{s}^d\in\{0,1,\oslash\},d=1,\dots,D\}.\tag{5}$$

The DAG structure on $\mathcal{S}$ is the D-th Cartesian power of the DAG with states $\{\oslash,0,1\}$, where 0 and 1 are children of $\oslash$. Concretely, the children of a state $\mathbf{s}=(\mathbf{s}^1,\dots,\mathbf{s}^D)$ are vectors that can be obtained from $\mathbf{s}$ by changing any one entry $\mathbf{s}^d$ from $\oslash$ to 0 or 1, and its parents are states that can be obtained by changing a single entry $\mathbf{s}^d\in\{0,1\}$ to $\oslash$.

We define $|\mathbf{s}|\triangleq\#\{\mathbf{s}^d\mid\mathbf{s}^d\in\{0,1\},d=1,\dots,D\}$, the number of non-void entries in $\mathbf{s}$, so $\mathcal{X}$ is naturally identified with $\{\mathbf{s}\in\mathcal{S}:|\mathbf{s}|=D\}$. There is an initial state $\mathbf{s}_0\triangleq(\oslash,\oslash,\dots,\oslash)$. Any trajectory from $\mathbf{s}_0$ to $\mathbf{x}\in\mathcal{X}$ has exactly D actions. A choice of trajectory from $\mathbf{s}_0$ to $\mathbf{x}$ amounts to a choice of the order in which the entries of $\mathbf{x}$ are assigned.

In this setting, the forward policy $P_F(\cdot|\mathbf{s};\boldsymbol{\theta})$ of a GFlowNet, introduced in §2.1, is a distribution over all ways to select a position with a void entry in $\mathbf{s}$ and a value (0 or 1) to assign to this entry. Thus the action space for a state $\mathbf{s}$ has size $2(D-|\mathbf{s}|)$. Correspondingly, the backward policy $P_B(\cdot|\mathbf{s};\boldsymbol{\theta})$ is a distribution over the $|\mathbf{s}|$ ways to select a position with a nonvoid entry in $\mathbf{s}$. We illustrate the mechanism of the forward and backward policies in Figure 3.

In our experiments, we take $P_F(\cdot|\mathbf{s};\boldsymbol{\theta})$ and $P_B(\cdot|\mathbf{s};\boldsymbol{\theta})$ to

be neural networks with a multilayer perceptron (MLP) architecture, where the input is a vector $\mathbf{s} \in \{\oslash, 0, 1\}^D$ encoded using a value of -1 for $\oslash$, and P_F and P_B share all weights except the final weight matrix that outputs logits for the forward and backward actions. The scalar $Z_{\boldsymbol{\theta}}$ is parametrized in the log domain, as suggested by Malkin et al. (2022). Specific implementation details are given in §4.

3.2. GFlowNet training towards a target distribution

Recall from §2.1 that, given a non-negative reward function $R : \mathcal{X} \to \mathbb{R}_{\geq 0}$, a GFlowNet can be trained so that its terminating probability distribution matches the reward distribution. To be precise, the marginal likelihood that a trajectory sampled from the GFlowNet's forward policy $P_F(\cdot|\cdot;\boldsymbol{\theta})$ terminates at a given state is proportional to the state's reward, $P_T(\mathbf{x}) \propto R(\mathbf{x})$. We now describe how GFlowNets could be trained towards matching a given reward.

Trajectory balance. To train the parameters $\boldsymbol{\theta}$ of the GFlowNet, we use the trajectory balance objective proposed by Malkin et al. (2022). Trajectory balance optimizes the following objective along complete trajectories $\tau = (\mathbf{s}_0 \to \mathbf{s}_1 \to \ldots \to \ldots \to \mathbf{s}_n)$:

$$\mathcal{L}_{\boldsymbol{\theta}}(\tau) = \left[\log \frac{Z_{\boldsymbol{\theta}} \prod_{t=0}^{n-1} P_F(\mathbf{s}_{t+1} \mid s_t; \boldsymbol{\theta})}{R(\mathbf{s}_n) \prod_{t=0}^{n-1} P_B(\mathbf{s}_t \mid \mathbf{s}_{t+1}; \boldsymbol{\theta})}\right]^2. \quad (6)$$

Proposition 1 of Malkin et al. (2022) shows that if this objective is globally minimized (*i.e.*, zeroed out) for all complete trajectories τ, then $P_T(\mathbf{x}) \propto R(\mathbf{x})$, *i.e.*, the forward policy samples proportionally to the reward. Trajectory balance improves training of GFlowNets under various metrics and characteristics of the reward landscape (Malkin et al., 2022) relative to previously proposed objectives.

Training policy. With the trajectory balance objective, we train the GFlowNet with stochastic gradient

$$\mathbb{E}_{\tau \sim \pi_{\boldsymbol{\theta}}(\tau)} \left[\nabla_{\boldsymbol{\theta}} \mathcal{L}_{\boldsymbol{\theta}}(\tau)\right] \quad (7)$$

with some training trajectory distribution $\pi_{\boldsymbol{\theta}}(\tau)$. Akin to on-policy RL settings, Malkin et al. (2022) took $\pi_{\boldsymbol{\theta}}$ to be the distribution over trajectories sampled from the current policy $P_F(\cdot \mid \cdot; \boldsymbol{\theta})$, or a perturbed / tempered version of it. That is, τ is sampled with $\mathbf{s}_{t+1} \sim P_F(\cdot|\mathbf{s}_t; \boldsymbol{\theta})$ starting from $\mathbf{s}_0$, perhaps raised to a power or mixed with a uniform action policy to ensure $\pi_{\boldsymbol{\theta}}$ has full support, which is a condition for obtaining the desired distribution (Malkin et al., 2022).

In addition to this forward sampling approach, we propose a complementary strategy to benefit from the circumstances where we are given some terminating states (data examples $\mathbf{x} \in \mathcal{X}$). For a terminating state $\mathbf{x}$, one can sample a reverse trajectory $\tau = (\mathbf{x} = \mathbf{s}_D \dashrightarrow \mathbf{s}_{D-1} \dashrightarrow \ldots \dashrightarrow \mathbf{s}_0)$, where $\mathbf{s}_t \sim P_B(\cdot|\mathbf{s}_{t+1}; \boldsymbol{\theta})$. Empirically, this backward trajectory sampling technique enables us to obtain a different trajectory distribution from the forward sampling distribution, as such backward trajectories visit regions of $\mathcal{S}$ near the true data samples that may be poorly explored by P_F, and could thus stabilize the optimization. Hereafter, we use $P_F(\tau)$ and $P_B(\tau|\mathbf{x})$ to denote the trajectory distributions that sample forward from $\mathbf{s}_0$ using P_F and backward from $\mathbf{x}$ using P_B, respectively. In experiments, we take the training trajectory distribution $\pi_{\boldsymbol{\theta}}$ to be a mixture of these two sampling methods (see steps 3-9 of Algorithm 1).

Canonical design of P_B. Bengio et al. (2021b) noted that while there may be multiple Markovian flows satisfying (2), for any choice of a *fixed* backward policy P_B, there is a unique forward policy P_F such that the corresponding $P_T(\mathbf{x})$ is proportional to the reward. Malkin et al. (2022) suggested fixing $P_B(\cdot|\mathbf{s})$ to be uniform over the parents of every state $\mathbf{s}$ as a canonical choice. We find this to be beneficial in some domains. Furthermore, in our setting, this choice also enforces a maximum-entropy property on the forward policy, as the following proposition shows.

Definition. The *entropy* of a Markovian flow F, denoted $\mathcal{H}[F]$, is the expected total entropy of its forward policy distributions along a complete trajectory:

$$\mathcal{H}[F] = \mathbb{E}_{(\mathbf{s}_0 \to \ldots \to \mathbf{s}_n) \sim P_F(\cdot|\cdot)} \left[\sum_{t=0}^{n-1} \mathcal{H}[P_F(\cdot|\mathbf{s}_t)]\right]. \quad (8)$$

Proposition 1. *Suppose G is the DAG defined in §3.1. Let R be a nonnegative reward function on $\mathcal{X}$ and let P_B° be the uniform backward policy on G. Let F° be the Markovian flow uniquely determined by P_B° and R subject to the reward matching constraint (2). Then F° has maximal entropy among all Markovian flows satisfying (2).*

Estimating GFlowNet data likelihood. The most commonly used metric in probabilistic modeling is the model's likelihood on a test set. A well-trained model should assign a high likelihood to data from the same underlying distribution as the training data, that is, the terminating probability distribution $P_T(\mathbf{x}) = \sum_{\tau=(\mathbf{s}_0 \to \ldots \to \mathbf{s}_D), \mathbf{s}_D = \mathbf{x}} P_F(\tau)$ would be close to the true data distribution. We overcome the intractability of the sum defining $P_T(\mathbf{x})$ (the number of terms is factorial in D) by importance sampling:

$$P_T(\mathbf{x}) = \mathbb{E}_{P_B(\tau|\mathbf{x})} \frac{P_F(\tau)}{P_B(\tau|\mathbf{x})} \approx \frac{1}{M} \sum_{j=1}^{M} \frac{P_F(\tau^j)}{P_B(\tau^j|\mathbf{x})}, \quad (9)$$

where $\tau^j \sim P_B(\tau|\mathbf{x})$ are trajectories sampled backward from $\mathbf{x}$ using P_B. We can then use the average GFlowNet log likelihood on test set, estimated using (9) with M large enough, as an evaluation metric.

3.3. Interleaved updates of GFlowNet and energy

The training of GFlowNets relies on a given function $R(\mathbf{x})$ to provide reward signals. In generative modeling, we typically set $R(\mathbf{x})$ to be the unnormalized target probability.

Algorithm 1 EB-GFN joint training framework

input Training dataset $\{\mathbf{x}_i\}_i$, hyperparameter $\alpha \in [0, 1]$
Initialize GFlowNet's P_F, P_B, Z with parameters $\boldsymbol{\theta}$, and the energy function $\mathcal{E}_{\boldsymbol{\phi}}$ with parameters $\boldsymbol{\phi}$.
repeat
  $X \sim \text{Bernoulli}(\alpha)$
  if $X = 1$ **then**
    Sample forward trajectory $\tau \sim P_F(\tau)$.
  else
    Uniformly sample $\mathbf{x}_i$ from dataset.
    Sample backward trajectory $\tau \sim P_B(\tau|\mathbf{x}_i)$.
  end if
  Update the GFlowNet via gradient step on $\mathcal{L}_{\boldsymbol{\theta}}(\tau)$ with reward $R(\mathbf{x}) = e^{-\mathcal{E}(\mathbf{x};\boldsymbol{\phi})}$ (Eq. (6)).
  Update energy function with Algorithm 2.
until some convergence condition

Algorithm 2 GFlowNet-guided energy function update

input Training dataset $\{\mathbf{x}_i\}_i$, GFlowNet providing $\{P_F, P_B, Z\}$, energy function $\mathcal{E}_{\boldsymbol{\phi}}$, horizon K.
Uniformly sample $\mathbf{x}$ from dataset.
Sample a K-step backward trajectory from $P_B(\cdot|\cdot;\boldsymbol{\theta})$: $\tau = (\mathbf{x} = \mathbf{s}_D \dashrightarrow \mathbf{s}_{D-1} \dashrightarrow \ldots \dashrightarrow \mathbf{s}_{D-K})$.
Sample a K-step forward trajectory from $P_F(\cdot|\cdot;\boldsymbol{\theta})$: $\tau' = (\mathbf{s}_{D-K} \to \mathbf{s}'_{D-K+1} \to \ldots \to \mathbf{s}'_D = \mathbf{x}')$.
Accept or reject $\mathbf{x}'$ via Eq. (11); set $\mathbf{x}' \leftarrow \mathbf{x}$ if reject.
Update $\boldsymbol{\phi}$ with gradient of $\mathcal{E}_{\boldsymbol{\phi}}(\mathbf{x}) - \mathcal{E}_{\boldsymbol{\phi}}(\mathbf{x}')$.

However, in many settings, we do not have access to this exact quantity, but only to a collection of data samples from a target distribution.

To address this issue, we propose to introduce an energy-based model $\mathcal{E}_{\boldsymbol{\phi}}(\mathbf{x})$ as an intermediate object between the data and the GFlowNet, to serve as the reward function with which the GFlowNet can be trained. The "final products" of training are then twofold: the trained energy model $\mathcal{E}_{\boldsymbol{\phi}}(\mathbf{x})$ and the GFlowNet sampling policy P_F.

We train the EBM associated with the GFlowNet in an approximate MLE manner similar to Eq. (4), but using the GFlowNet policy to generate negative examples, $\mathbf{x}' \sim P_T(\mathbf{x}')$. In the basic form of the energy function training procedure, updates to $\boldsymbol{\phi}$ are made proportionally to

$$\mathbb{E}_{\mathbf{x}\sim p_{\text{data}}(\mathbf{x})} \nabla_{\boldsymbol{\phi}} \mathcal{E}_{\boldsymbol{\phi}}(\mathbf{x}) - \mathbb{E}_{\mathbf{x}'\sim P_T(\mathbf{x}')} \nabla_{\boldsymbol{\phi}} \mathcal{E}_{\boldsymbol{\phi}}(\mathbf{x}'), \tag{10}$$

where the GFlowNet terminating probability distribution $P_T(\mathbf{x})$ corresponds to marginalizing the forward trajectory distribution $P_F(\tau)$ on its non-terminating states. If the GFlowNet is perfectly trained (with zero training loss), its terminating probability distribution will be equal to the energy distribution, and thus this approximate MLE objective is an unbiased estimate of the maximum likelihood training of EBM. In this way, we amortize the MCMC sampling computation into the GFlowNet training process.

GFlowNet as an MCMC transition kernel. In contrastive divergence training of EBMs (4), K-step MCMC is used to generate the negative samples. For each step of traditional MCMC methods, one first proposes a local random perturbation of the current sample $\mathbf{x}$ (seen as a state of the Markov chain) to some nearby point $\mathbf{x}'$, and then decides whether to accept this transition[2] according to the Metropolis-Hastings (MH) rejection rule (Hastings, 1970).

[2]This should be distinguished from the 'transition' (action) in GFlowNets, which happens between two states $\mathbf{s}_t$ and $\mathbf{s}_{t+1}$ in $\mathcal{S}$.

Here, we show that a GFlowNet can also be used for generating a proposal $\mathbf{x}'$ from a given point $\mathbf{x}$ (Fig. 1). Given $\mathbf{x}$ and a fixed number of steps K ($1 \le K \le D$), we first sample a K-step trajectory with the backward policy P_B:

$$\tau = (\mathbf{x} = \mathbf{s}_D \dashrightarrow \mathbf{s}_{D-1} \dashrightarrow \ldots \dashrightarrow \mathbf{s}_{D-K}), \mathbf{s}_t \sim P_B(\mathbf{s}_t|\mathbf{s}_{t+1}),$$

and then sample with the forward policy P_F, starting at $\mathbf{s}_{D-K}$ until a new terminal state $\mathbf{x}'$ is reached:

$$\tau' = (\mathbf{s}'_{D-K} \to \ldots \to \mathbf{s}'_D = \mathbf{x}'), \quad \mathbf{s}'_{t+1} \sim P_F(\mathbf{s}'_{t+1}|\mathbf{s}'_t),$$

where we have $\mathbf{s}'_{D-K} = \mathbf{s}_{D-K}$. For convenience, we denote this back-and-forth trajectory by (τ, τ') and the reverse trajectory by

$$(\tau'_-, \tau_-) = (\mathbf{s}'_D \dashrightarrow \ldots \dashrightarrow \mathbf{s}'_{D-K} = \mathbf{s}_{D-K} \to \ldots \to \mathbf{s}_D).$$

Similar to K-step MCMC proposals, this GFlowNet proposal only changes the values of at most K different entries.

With τ, τ' as above, we extend the definitions for complete trajectories (Eq. 1) to $P_F(\tau) = \prod_{t=D-K}^{D-1} P_F(\mathbf{s}_{t+1}|\mathbf{s}_t)$ and $P_B(\tau|\mathbf{x}) = \prod_{t=D-K}^{D-1} P_B(\mathbf{s}_t|\mathbf{s}_{t+1})$. The probability of a transition from $\mathbf{x}$ to $\mathbf{x}'$ along the back-and-forth trajectory (τ, τ') is $P_B(\tau|\mathbf{x})P_F(\tau')$. Similarly, the probability of going $\mathbf{x}'$ to $\mathbf{x}$ along the reverse trajectory (τ'_-, τ_-) is $P_B(\tau'|\mathbf{x}')P_F(\tau)$. With the MH rule, if the move from $\mathbf{x}$ to $\mathbf{x}'$ is accepted with probability

$$A_{\tau,\tau'}(\mathbf{x} \to \mathbf{x}') \triangleq \min\left[1, \frac{e^{-\mathcal{E}_{\boldsymbol{\phi}}(\mathbf{x}')}}{e^{-\mathcal{E}_{\boldsymbol{\phi}}(\mathbf{x})}} \frac{P_B(\tau|\mathbf{x})P_F(\tau')}{P_B(\tau'|\mathbf{x}')P_F(\tau)}\right], \tag{11}$$

then the stationary distribution of an iterated application of such steps is equal to the desired reward distribution.

The following proposition shows that with a perfectly trained GFlowNet, we can cheaply obtain a high-dimensional form of Gibbs sampling, where K variables are updated at each step[3]:

Proposition 2. *If a GFlowNet fits the reward perfectly,* i.e., *satisfies (2), then $A_{\tau,\tau'}(\mathbf{x} \to \mathbf{x}') = 1$, so the MH rejection step will always accept the proposal.*

[3]Vanilla block Gibbs sampling would require computation exponential in K in order to compute 2^K possible K-bit changes, with their energies and their normalizing constant.

Table 1. Mean negative log-RMSE (higher is better) between data-generating matrix J and learned matrix J_ϕ for different values of σ. We find standard deviation < 0.1 between runs for each setting.

	$D = 10^2$					$D = 9^2$	
Method \ σ	0.1	0.2	0.3	0.4	0.5	-0.1	-0.2
Gibbs	4.8	4.7	**3.4**	**2.6**	**2.3**	4.8	4.7
GWG	4.8	4.7	**3.4**	**2.6**	**2.3**	4.8	4.7
EB-GFN	**6.1**	**5.1**	3.3	**2.6**	**2.3**	**5.7**	**5.1**

The proof is in §B. Notice that if $K = D$, the proposed transition is equivalent to directly sampling from $\mathbf{s}_{D-K} = \mathbf{s}_0$ with P_F, which is independent of the backward policy and of the starting sample $\mathbf{x}$.

As a relaxation of (10) that can accelerate learning, we propose to generate negative samples using this back-and-forth GFlowNet transition proposal. Unlike MCMC methods, we do not iterate this kernel, but perform only one single step to generate negative samples $\mathbf{x}'$ for each EBM parameter update, similarly to one-step contrastive divergence. As with MCMC-based contrastive divergence, it may be beneficial to begin with a small K and gradually increase it over the course of training. In all of our experiments, we either use a constant $K = D$ (unconditional samples from the GFlowNet are used as negative examples, as in Eq. (10)) or gradually increase K from 1 to D. We summarize our use of the GFlowNet proposal in EBM training in Algorithm 2.

Summary. We propose a joint training framework (Algorithm 1), where the EBM and the GFlowNet are optimized alternately: the energy function serves as the (negative log-) reward function for the GFlowNet, which is trained with the trajectory balance objective to sample from the evolving energy model, while the energy function is trained with an approximate MLE gradient, where the GFlowNet provides negative samples $\mathbf{x}'$ through an MCMC transition proposal that approximates block Gibbs sampling.

4. Experiments

4.1. Ising models

We validate the EB-GFN algorithm on the Ising model (Ising, 1925). The Ising model is an elementary example of a Markov random field and is widely studied in mathematics and physics (see MacKay, 2003, §31). A D-particle Ising model is a distribution over D-dimensional binary vectors, with entries called *spins*. To keep with established conventions, we call the two possible values of each spin $\{+1, -1\}$ rather than $\{0, 1\}$. The distribution is given by an energy model, where the energy is a quadratic form (with symmetric $D \times D$ matrix J) evaluated on the spin vector:

$$P(\mathbf{x}) \propto \exp(-\mathcal{E}_J(\mathbf{x})), \quad \mathbf{x} \in \{\pm 1\}^D,$$
$$\mathcal{E}_J(\mathbf{x}) \triangleq -\mathbf{x}^\top J \mathbf{x} = -\sum_{i=1}^{D}\sum_{j=1}^{D} J_{ij}\mathbf{x}^i\mathbf{x}^j.$$

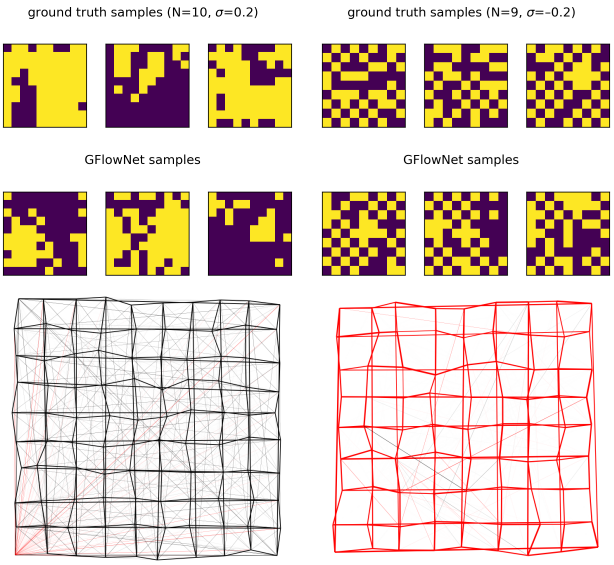

Figure 4. Top: Samples from an Ising model and samples from a GFlowNet trained jointly with an energy function on 2000 such samples. The generated samples display similar characteristics to the ground truth samples: contiguous regions with the same spin for $\sigma > 0$ and checkerboard texture for $\sigma < 0$. *Bottom:* A visualization of the learned matrix J_ϕ as a graph, where the thickness of the edge between nodes i and j is proportional to $(J_\phi)_{ij}$ and red/black edges represent negative/positive entries. Despite EB-GFN having no a priori knowledge of the grid structure, it almost perfectly recovers the full $D \times D$ matrix – 4950 (left) or 4851 (right) degrees of freedom – from 2000 discrete samples.

Positive entries J_{ij} encourage $\mathbf{x}^i$ and $\mathbf{x}^j$ to have the same spin, while negative J_{ij} push them to have opposite spins. The matrix J is often taken to be a multiple of the adjacency matrix of a graph. Here we consider models of the form $J = \sigma A_N$, where $\sigma \in \mathbb{R}$ and A_N is the adjacency matrix of a $N \times N$ grid with toroidal wrap-around ($D = N^2$).

We consider 10×10 grids with $\sigma = 0.1, 0.2, \ldots, 0.5$ and 9×9 grids with $\sigma = -0.1, -0.2$.[4] For each setting of σ and N, we use standard methods (Wang & Swendsen, 1990) to generate 2000 samples $\{\mathbf{x}_i\}_{i=1}^{2000}$ from the Ising model with energy $\mathcal{E}_J$. We then use the EB-GFN algorithm to jointly fit a symmetric matrix J_ϕ, giving an estimated Ising EBM $\mathcal{E}_{J_\phi}$, and a GFlowNet that samples from this Ising EBM. Note that the EB-GFN algorithm does not have access to the true data-generating matrix J, only to the collection of samples $\{\mathbf{x}_i\}$. This is a simple test case for EB-GFN, since the energy is parametrized by a single matrix J_ϕ, not by a deep model. We evaluate the discrepancy (RMSE) between the true matrix J and the learned matrix J_ϕ.

For simplicity, we set $K = D$ for the negative sampling step and use a training policy with $\alpha = 1$ (no backward paths from training examples). Details can be found in §C.1. We

[4] We use an odd grid size when $\sigma < 0$ because such a model has many modes, each resembling a checkerboard with 'seams' where the checkerboard pattern is violated.

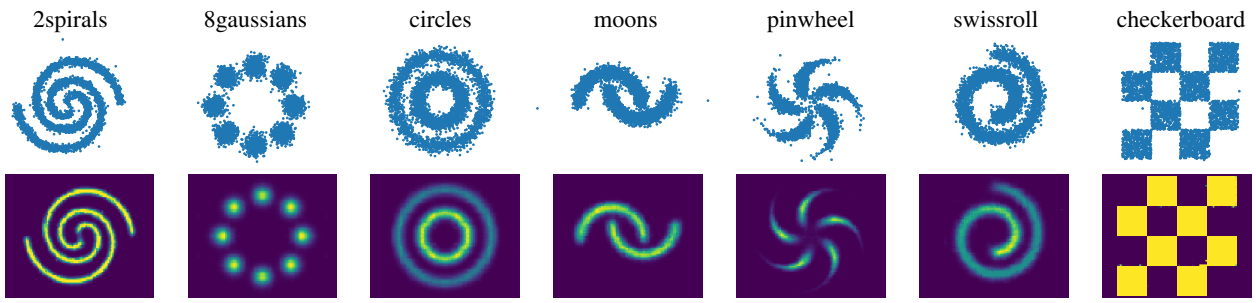

Figure 5. Top: Visualization of the samples generated with a learned GFlowNet. *Bottom:* Visualization of the energy function learned jointly with the GFlowNet. Due to limited space, we defer the visualization of baselines to Fig. C.2.

Table 2. Experiment results with seven 2D synthetic problems. We display the negative log-likelihood (NLL) and MMD (in units of 1×10^{-4}). Note that ALOE+ uses a thirty times larger parametrization than ALOE and EB-GFN.

Metric	Method	2spirals	8gaussians	circles	moons	pinwheel	swissroll	checkerboard
NLL↓	PCD	20.094	19.991	20.565	19.763	19.593	20.172	21.214
	ALOE	20.295	20.350	20.565	19.287	19.821	20.160	54.653
	ALOE+	20.062	19.984	20.570	19.743	19.576	20.170	21.142
	EB-GFN	**20.050**	**19.982**	**20.546**	**19.732**	**19.554**	**20.146**	**20.696**
MMD↓	PCD	2.160	0.954	0.188	0.962	0.505	1.382	2.831
	ALOE	21.926	107.320	0.497	26.894	39.091	0.471	61.562
	ALOE+	**0.149**	**0.078**	0.636	0.516	1.746	0.718	12.138
	EB-GFN	0.583	0.531	**0.305**	**0.121**	**0.492**	**0.274**	**1.206**

compare EB-GFN with the Gibbs and Gibbs-With-Gradients (GWG) PCD algorithms (Grathwohl et al., 2021b). Table 1 shows the advantage of EB-GFN. Fig. 4 shows how faithful both the samples and the energy function obtained by the GFlowNet are, suggesting that the GFlowNet is able to discover generalizable structure from the data.

4.2. Synthetic tasks

The experiment described in this subsection follows the setup of Dai et al. (2020). We aim to model seven different distributions over 32-dimensional binary data that are discretizations of continuous distributions over the plane (shown in Fig. C.1). To convert planar data $(x, y) \in \mathbb{R}^2$ to 32-dimensional binary data, we quantize both x and y into 2^{16} equal-width buckets, then remap them to 16-bit representations via a Gray code (Gray, 1953), so that any two neighbouring buckets differ in exactly one bit. As a result, the methods are required to model data in $\{0, 1\}^{32}$.

We compare with PCD-10 and ALOE (Dai et al., 2020), two baselines for energy modeling in discrete spaces. We use the same energy function architecture and training protocol as Dai et al. (2020). The PCD-10 baseline utilizes Gibbs sampling with a replay buffer and the random restart mechanism (Du & Mordatch, 2019) for negative sample generation. ALOE learns three neural networks for negative sampling: a proposal model to provide initial samples and a pair of models (local search policy and stop policy) used to iteratively refine these samples. The initial proposal model could be either a simple multinomial distribution or a large autoregressive network. We name them ALOE and ALOE+ respectively, as the former has a similar total number of parameters to our GFlowNet, while ALOE+ is $32\times$ larger. For GFlowNet training, we use a mixed training policy with $\alpha = 0.5$ and a schedule with linearly increasing K for the back-and-forth proposal. See §C.2 for details.

For qualitative evaluation, we first visualize in Fig. 5 the heatmaps of the learned energy functions and some GFlowNet-generated samples by remapping the Gray code representations of samples back to 2-D space. As a comparison, we also visualize the energy model baselines in Fig. C.2. We observe that for multimodal tasks such as *checkerboard* and *8gaussians*, GFlowNets are much better at capturing the modes and their structure than the baselines (as illustrated schematically in Fig. 1). We hypothesize that *multimodality is easier to handle with GFlowNets than with MCMC if there are generalizable regularities that make it possible for the GFlowNet to guess new modes from those already visited and from which it has learned.*

We quantitatively evaluate the algorithms in Table 2 by showing for each method the NLL of a large independent sample of ground truth data and the exponential Hamming MMD (Gretton et al., 2012) between ground truth data and generated samples as performance metrics. Our method outperforms the baselines on all datasets and metrics except MMD on *2spirals* and *8gaussians*, where it still exceeds comparable-size baselines. Of note, ALOE does not surpass

Table 3. Experiments on discrete image modeling. We display the negative log likelihood (NLL) per sample for different algorithms.

Dataset \ Method	Gibbs	GWG	EB-GFN
Omniglot	133.92	114.96	**112.59**
Silhouettes	475.55	188.82	**185.57**
Static MNIST	173.61	**99.36**	102.43
Dynamic MNIST	162.25	108.29	**105.75**

PCD for many tasks without a large initial proposal network, which shows a potential weakness of its local search strategy. We defer more results and related details to §C.2.

4.3. Discrete image modeling

Here, we aim to generatively model previously studied image datasets in discrete high-dimensional spaces. These are generally hard problems as we lose the information of continuous pixel values, and prevalent scalable methods (Welling & Teh, 2011; Ma et al., 2015) are not applicable to training deep EBMs on discrete data. An MLP is utilized as the energy function, and the baseline training methods are PCD with Gibbs sampling and the Gibbs-With-Gradients sampling method (GWG; Grathwohl et al., 2021b). Experiments are performed on four different binary image datasets. Following the experimental settings in Grathwohl et al. (2021b), the EBM is trained via PCD-100 with different negative sampling methods, and a replay buffer is adopted as in Du & Mordatch (2019). The validation and evaluation protocol is also kept aligned with Grathwohl et al. (2021b), where the checkpoint with the best negative log-likelihood on the validation set is reported.

The test set NLLs are displayed in Table 3. The results indicates that EB-GFN reaches state-of-the-art energy modeling performance, surpassing GWG, on three of the four datasets. As a supplement, a visualization of Dynamic MNIST data is shown in Figure 6. More details are in §C.3.

5. Related Work

Energy-based models. EBM, one of the central methods in generative modeling, has proved effective with energy functions parametrized by deep nets (Hinton et al., 2006; Salakhutdinov & Hinton, 2009). To avoid costly MCMC simulation with deep models, contrastive divergence-type methods (Hinton, 2002; Tieleman, 2008; Xie et al., 2016; Du et al., 2021) were proposed to approximate the energy gradient. It has also been shown that better objectives beyond vanilla CD helps EBM training (Yu et al., 2020). Training methods have been proposed for better stability, shorter mixing time, faster training (Nijkamp et al., 2019; Du & Mordatch, 2019; Grathwohl et al., 2021a; Gao et al., 2021). Recent work shows that it can be beneficial to learn the sampler or the proposal distribution as well (Dai et al., 2019; Arbel et al., 2021), a finding that this work extends to discrete spaces.

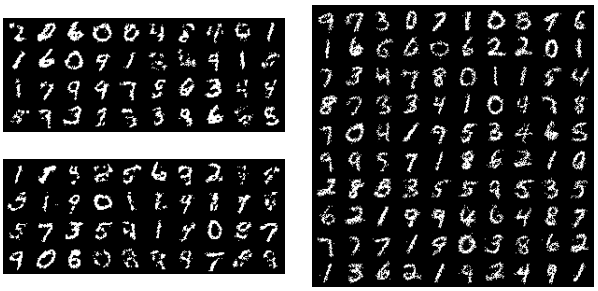

Figure 6. Visualization of the Dynamic MNIST samples generated from Gibbs *(top left)*, GWG *(bottom left)* and GFlowNet *(right)*. EB-GFN models the details of certain modes better (see, *e.g.*, the bottom right corner of the GWG samples). Note that the original images were binarized, which explains in part why all samples look noisier than the familiar gray-level MNIST images.

Related methods. Autoregressive models (Sutskever et al., 2011; Graves, 2013; Germain et al., 2015), like our GFlowNets, generate each entry of a data vector sequentially, but in a fixed order. Autoregressive models can also be used for data without a natural order (Uria et al., 2016; van den Oord et al., 2016b;a; Meng et al., 2021). Some recent work (Emelianenko et al., 2019; Li et al., 2021) allows generation order to be learned. We are the first to interpret learning of generation order as a joint inference of forward (construction) and reverse (erasure) Markov processes.

Discrete inference. Probabilistic inference in discrete settings is generally harder than in continuous spaces. Many optimized sampling methods for discrete spaces have been developed (Titsias & Yau, 2017; Zanella, 2019; Han et al., 2020; Zhang et al., 2022). Most applicable to our experiment domains, Grathwohl et al. (2021b) uses a continuous relaxation to approximate the local energy landscape, while Dai et al. (2020) introduces a local search strategy in the variational distribution to initialize negative sample generation.

6. Conclusion

We have extended GFlowNets to the setting where one is given a dataset rather than a fixed energy function, and we learn both the GFlowNet sampler and the energy function. In doing so, we introduced a new proposal for MH MCMC that approaches block Gibbs sampling as the GFlowNet training converges. The main advantage of this proposal is that it can perform large jumps in the state space, unlike simple Gibbs sampling, taking advantage of the compositional structure that the GFlowNet may have uncovered that allows it to generalize across modes of the distribution and more easily jump between them (Fig. 1). Future work can consider iterating such proposals from a trained GFlowNet for efficient exploration of the space, rather than just sampling complete trajectories ($K = D$) from the GFlowNet.

The cost of such an approach is that in addition to train-

ing the energy function, we also have to train the sampler. However, this cost can be amortized if we intend to use the sampler later, since generating samples from the GFlowNet is often much cheaper than from a MCMC (if we want to make sure to cover the modes well). We hypothesize that the above advantages explain the good comparative results obtained here, and expect that the proposed approach may be extended to many other types of generative tasks.

Acknowledgement

We thank Yilun Du, Ricky T. Q. Chen, and Mila GFlowNet group for helpful discussion. Dinghuai Zhang thanks the never-ending snow storm in Montreal for preventing him from any form of outdoor activity :). Zhen Liu thanks miHoYo for the joy from their awesome games in the tough winters. Aaron Courville thanks the support of Microsoft Research, Hitachi and CIFAR. Yoshua Bengio acknowledges the funding from CIFAR, Samsung, IBM and Microsoft.

References

Arbel, M., Zhou, L., and Gretton, A. Generalized energy based models. *International Conference on Learning Representations (ICLR)*, 2021.

Ba, J., Kiros, J. R., and Hinton, G. E. Layer normalization. *ArXiv*, abs/1607.06450, 2016.

Bengio, E., Pineau, J., and Precup, D. Interference and generalization in temporal difference learning. *International Conference on Machine Learning (ICML)*, 2020.

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)*, 2021a.

Bengio, Y., Mesnil, G., Dauphin, Y., and Rifai, S. Better mixing via deep representations. *International Conference on Machine Learning (ICML)*, 2013.

Bengio, Y., Deleu, T., Hu, E., Lahlou, S., Tiwari, M., and Bengio, E. GFlowNet foundations. *arXiv preprint 2111.09266*, 2021b.

Besold, T. R., d'Avila Garcez, A. S., Bader, S., Bowman, H., Domingos, P. M., Hitzler, P., Kühnberger, K.-U., Lamb, L., Lowd, D., Lima, P. M. V., de Penning, L., Pinkas, G., Poon, H., and Zaverucha, G. Neural-symbolic learning and reasoning: A survey and interpretation. *arXiv preprint 1711.03902*, 2017.

Clevert, D.-A., Unterthiner, T., and Hochreiter, S. Fast and accurate deep network learning by exponential linear units (ELUs). *International Conference on Learning Representations (ICLR)*, 2016.

Dai, B., Liu, Z., Dai, H., He, N., Gretton, A., Song, L., and Schuurmans, D. Exponential family estimation via adversarial dynamics embedding. *Neural Information Processing Systems (NeurIPS)*, 2019.

Dai, H., Singh, R., Dai, B., Sutton, C., and Schuurmans, D. Learning discrete energy-based models via auxiliary-variable local exploration. *Neural Information Processing Systems (NeurIPS)*, 2020.

Desjardins, G., Courville, A. C., Bengio, Y., Vincent, P., and Delalleau, O. Tempered Markov chain Monte Carlo for training of restricted Boltzmann machines. *Artificial Intelligence and Statistics (AISTATS)*, 2010.

Du, Y. and Mordatch, I. Implicit generation and generalization in energy-based models. *Neural Information Processing Systems (NeurIPS)*, 2019.

Du, Y., Li, S., Tenenbaum, J. B., and Mordatch, I. Improved contrastive divergence training of energy based models. *International Conference on Machine Learning (ICML)*, 2021.

Emelianenko, D., Voita, E., and Serdyukov, P. Sequence modeling with unconstrained generation order. *Neural Information Processing Systems (NeurIPS)*, 2019.

Ford, L. R. and Fulkerson, D. R. Maximal flow through a network. *Canadian Journal of Mathematics*, 8:243–248, 1956.

Gao, R., Song, Y., Poole, B., Wu, Y. N., and Kingma, D. P. Learning energy-based models by diffusion recovery likelihood. *International Conference on Learning Representations (ICLR)*, 2021.

Germain, M., Gregor, K., Murray, I., and Larochelle, H. MADE: Masked autoencoder for distribution estimation. *International Conference on Machine Learning (ICML)*, 2015.

Grathwohl, W., Chen, R. T. Q., Bettencourt, J., Sutskever, I., and Duvenaud, D. K. Ffjord: Free-form continuous dynamics for scalable reversible generative models. *International Conference on Learning Representations (ICLR)*, 2019.

Grathwohl, W., Kelly, J., Hashemi, M., Norouzi, M., Swersky, K., and Duvenaud, D. K. No MCMC for me: Amortized sampling for fast and stable training of energy-based models. *International Conference on Learning Representations (ICLR)*, 2021a.

Grathwohl, W., Swersky, K., Hashemi, M., Duvenaud, D. K., and Maddison, C. J. Oops I took a gradient: Scalable sampling for discrete distributions. *International Conference on Machine Learning (ICML)*, 2021b.

Graves, A. Generating sequences with recurrent neural networks. *arXiv preprint 1308.0850*, 2013.

Gray, F. Pulse code communication. *US Patent 2,632,058*, 1953.

Gretton, A., Borgwardt, K. M., Rasch, M. J., Schölkopf, B., and Smola, A. A kernel two-sample test. *J. Mach. Learn. Res.*, 13:723–773, 2012.

Haarnoja, T., Tang, H., Abbeel, P., and Levine, S. Reinforcement learning with deep energy-based policies. *International Conference on Machine Learning (ICML)*, 2017.

Han, J., Ding, F., Liu, X., Torresani, L., Peng, J., and Liu, Q. Stein variational inference for discrete distributions. *Artificial Intelligence and Statistics (AISTATS)*, 2020.

Hastings, W. K. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57(1): 97–109, 1970.

Hinton, G. E. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14:1771–1800, 2002.

Hinton, G. E., Osindero, S., and Teh, Y. W. A fast learning algorithm for deep belief nets. *Neural Computation*, 18: 1527–1554, 2006.

Ising, E. Beitrag zur Theorie des Ferromagnetismus. *Zeitschrift für Physik*, 31(1):253–258, 1925.

Jain, M., Bengio, E., García, A., Rector-Brooks, J., Dossou, B. F. P., Ekbote, C. A., Fu, J., Zhang, T., Kilgour, M., Zhang, D., Simine, L., Das, P., and Bengio, Y. Biological sequence design with gflownets. *ArXiv*, abs/2203.04115, 2022.

LeCun, Y., Chopra, S., Hadsell, R., Ranzato, A., and Huang, F. J. A tutorial on energy-based learning. 2006.

Li, X., Trabucco, B., Park, D., Luo, M., Shen, S. M., Darrell, T., and Gao, Y. Discovering non-monotonic autoregressive orderings with variational inference. *International Conference on Learning Representations (ICLR)*, 2021.

Ma, Y., Ma, Y.-A., Chen, T., and Fox, E. B. A complete recipe for stochastic gradient MCMC. *Neural Information Processing Systems (NIPS)*, 2015.

MacKay, D. J. C. *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, 2003.

Malkin, N., Jain, M., Bengio, E., Sun, C., and Bengio, Y. Trajectory balance: Improved credit assignment in GFlowNets. *arXiv preprint 2201.13259*, 2022.

Meng, C., Song, J., Song, Y., Zhao, S., and Ermon, S. Improved autoregressive modeling with distribution smoothing. *International Conference on Learning Representations (ICLR)*, 2021.

Nijkamp, E., Hill, M., Zhu, S.-C., and Wu, Y. N. Learning non-convergent non-persistent short-run mcmc toward energy-based model. *Neural Information Processing Systems (NeurIPS)*, 2019.

Nijkamp, E., Hill, M., Han, T., Zhu, S.-C., and Wu, Y. N. On the anatomy of MCMC-based maximum likelihood learning of energy-based models. *Association for the Advancement of Artificial Intelligence (AAAI)*, 2020.

Salakhutdinov, R. Learning in Markov random fields using tempered transitions. *Neural Information Processing Systems (NIPS)*, 2009.

Salakhutdinov, R. and Hinton, G. E. Deep Boltzmann machines. *Artificial Intelligence and Statistics (AISTATS)*, 2009.

Song, Y. and Kingma, D. P. How to train your energy-based models. *arXiv preprint 2101.03288*, 2021.

Sutskever, I., Martens, J., and Hinton, G. E. Generating text with recurrent neural networks. *International Conference on Machine Learning (ICML)*, 2011.

Sutton, R. S. Learning to predict by the methods of temporal differences. *Machine Learning*, 3:9–44, 2005.

Sutton, R. S. and Barto, A. G. Reinforcement learning: An introduction. *IEEE Transactions on Neural Networks*, 16: 285–286, 2005.

Tai, K. S., Socher, R., and Manning, C. D. Improved semantic representations from tree-structured long short-term memory networks. *Association for Computational Linguistics (ACL)*, 2015.

Tieleman, T. Training restricted Boltzmann machines using approximations to the likelihood gradient. *International Conference on Machine Learning (ICML)*, 2008.

Tieleman, T. and Hinton, G. E. Using fast weights to improve persistent contrastive divergence. *International Conference on Machine Learning (ICML)*, 2009.

Titsias, M. K. and Yau, C. The hamming ball sampler. *Journal of the American Statistical Association*, 112:1598 – 1611, 2017.

Uria, B., Côté, M.-A., Gregor, K., Murray, I., and Larochelle, H. Neural autoregressive distribution estimation. *Journal of Machine Learning Research*, 17: 205:1–205:37, 2016.

van den Oord, A., Kalchbrenner, N., Espeholt, L., Kavukcuoglu, K., Vinyals, O., and Graves, A. Conditional image generation with pixelcnn decoders. *Neural Information Processing Systems (NIPS)*, 2016a.

van den Oord, A., Kalchbrenner, N., and Kavukcuoglu, K. Pixel recurrent neural networks. *International Conference on Machine Learning (ICML)*, 2016b.

Wang, J.-S. and Swendsen, R. H. Cluster Monte Carlo algorithms. *Physica A: Statistical Mechanics and its Applications*, 167(3):565–579, 1990.

Welling, M. and Teh, Y. W. Bayesian learning via stochastic gradient Langevin dynamics. *International Conference on Machine Learning (ICML)*, 2011.

Xie, J., Lu, Y., Zhu, S.-C., and Wu, Y. N. A theory of generative convnet. *ArXiv*, abs/1602.03264, 2016.

Yu, L., Song, Y., Song, J., and Ermon, S. Training deep energy-based models with f-divergence minimization. *International Conference on Machine Learning (ICML)*, 2020.

Zanella, G. Informed proposals for local mcmc in discrete spaces. *Journal of the American Statistical Association*, 115:852 – 865, 2019.

Zhang, R., Liu, X., and Liu, Q. A langevin-like sampler for discrete distributions. In *International Conference on Machine Learning*. PMLR, 2022.

A. Summary of GFlowNet notation

Symbol	Description
D	data dimension
$\mathcal{S}$	state space $\{0, 1, \oslash\}^D$
$\mathcal{X}$	data space $\{0, 1\}^D$, identified with the set of terminal states in $\mathcal{S}$
$\mathcal{A}$	action / transition space (edges $\mathbf{s} \to \mathbf{s}'$)
$\mathcal{T}$	set of complete trajectories
G	directed acyclic graph $(\mathcal{S}, \mathcal{A})$
$\mathbf{s}$	state in $\mathcal{S}$
$\mathbf{s}_0$	initial state $(\oslash, \ldots, \oslash) \in \mathcal{S}$
$\mathbf{x}$	terminal state in $\mathcal{X}$
$F : \mathcal{T} \to \mathbb{R}$	Markovian flow
$F : \mathcal{S} \to \mathbb{R}$	state flow, used in the proof of Proposition 1
$F : \mathcal{A} \to \mathbb{R}$	edge flow, used in the proof of Proposition 1
P_F	forward policy (distribution over children)
P_B	backward policy (distribution over parents)
Z	scalar, equal to $\sum_{\tau \in \mathcal{T}} F(\tau)$ for a Markovian flow

B. Proofs of propositions

Recall Proposition 1:

Proposition. *Suppose G is the DAG defined in §3.1. Let R be a nonnegative reward function on $\mathcal{X}$ and let P_B° be the uniform backward policy on G. Let F° be the Markovian flow uniquely determined by P_B° and R subject to the reward matching constraint (2). Then F° has maximal entropy among all Markovian flows satisfying (2).*

Proof. We use the following definitions from Bengio et al. (2021b):

For a trajectory flow F and for any state $\mathbf{s}$, define the state flow $F(\mathbf{s}) = \sum_{\mathbf{s} \in \tau} F(\tau)$, and, for any edge $\mathbf{s} \to \mathbf{s}'$, the edge flow

$$F(\mathbf{s} \to \mathbf{s}') = \sum_{\tau = (\ldots \to \mathbf{s} \to \mathbf{s}' \to \ldots)} F(\tau).$$

Notice that $Z = F(s_0)$ immediately from (1).

If F is a Markovian flow, then P_F and P_B can be computed in terms of state and edge flows:

$$P_F(\mathbf{s}'|\mathbf{s}) = \frac{F(\mathbf{s} \to \mathbf{s}')}{F(\mathbf{s})}, \quad P_B(\mathbf{s}|\mathbf{s}') = \frac{F(\mathbf{s} \to \mathbf{s}')}{F(\mathbf{s}')}, \tag{12}$$

and we have

$$F(\mathbf{s}) = \sum_{(\mathbf{s}'' \to \mathbf{s}) \in \mathcal{A}} F(\mathbf{s}'' \to \mathbf{s}) = \sum_{(\mathbf{s} \to \mathbf{s}') \in \mathcal{A}} F(\mathbf{s} \to \mathbf{s}'). \tag{13}$$

The following computation shows that the entropy of the forward policy, defined by (8), equals a similar expression for the

backward policy:

$$
\begin{aligned}
\mathcal{H}[F] &= \mathbb{E}_{(\mathbf{s}_0\to\ldots\to\mathbf{s}_n)\sim P_F}\left[\sum_{t=0}^{n-1}\mathcal{H}[P_F(\cdot|\mathbf{s}_t)]\right] && \text{(definition (8))}\\
&= \sum_{\mathbf{s}\in\mathcal{S}\text{ nonterminal}} \mathbb{P}[\tau=(\ldots\to\mathbf{s}\to\ldots)]\mathcal{H}[P_F(\cdot|\mathbf{s})] && \text{(linearity of expectation)}\\
&= -\sum_{\mathbf{s}\in\mathcal{S}\text{ nonterminal}} \frac{F(\mathbf{s})}{Z}\sum_{\mathbf{s}':(\mathbf{s}\to\mathbf{s}')\in\mathcal{A}} P_F(\mathbf{s}'|\mathbf{s})\log P_F(\mathbf{s}'|\mathbf{s}) && \text{(by definition of state flow)}\\
&= -\sum_{(\mathbf{s},\mathbf{s}')\in\mathcal{A}} \frac{F(\mathbf{s})}{Z}\frac{F(\mathbf{s}\to\mathbf{s}')}{F(\mathbf{s})}\log\frac{F(\mathbf{s}\to\mathbf{s}')}{F(\mathbf{s})} && \text{(grouping terms and (12))}\\
&= \frac{-1}{Z}\left(\sum_{(\mathbf{s}\to\mathbf{s}')\in\mathcal{A}} F(\mathbf{s}\to\mathbf{s}')\log F(\mathbf{s}\to\mathbf{s}') - \sum_{\mathbf{s}\in\mathcal{S}\text{ nonterminal}} F(\mathbf{s})\log F(\mathbf{s})\right) && \text{(rearrangement and (13))}\\
&= -\sum_{(\mathbf{s},\mathbf{s}')\in\mathcal{A}} \frac{F(\mathbf{s}')}{Z}\frac{F(\mathbf{s}\to\mathbf{s}')}{F(\mathbf{s}')}\log\frac{F(\mathbf{s}\to\mathbf{s}')}{F(\mathbf{s}')} + \underbrace{\frac{1}{Z}\left(Z\log Z - \sum_{x\text{ terminal}} F(x)\log F(x)\right)}_{\Delta} && \text{(rearrangement and (13))}\\
&= -\sum_{\mathbf{s}'\in\mathcal{S}\text{ noninitial}} \frac{F(\mathbf{s}')}{Z}\sum_{\mathbf{s}:(\mathbf{s}\to\mathbf{s}')\in\mathcal{A}} P_B(\mathbf{s}|\mathbf{s}')\log P_B(\mathbf{s}|\mathbf{s}') + \Delta && \text{(grouping terms and (12))}\\
&= \sum_{\mathbf{s}'\in\mathcal{S}\text{ noninitial}} \mathbb{P}[\tau=(\ldots\to\mathbf{s}'\to\ldots)]\mathcal{H}[P_B(\cdot|\mathbf{s}')] + \Delta && \text{(by definition of state flow)}\\
&= \mathbb{E}_{(\mathbf{s}_0\to\ldots\to\mathbf{s}_n)\sim P_F}\left[\sum_{t=1}^{n}\mathcal{H}[P_B(\cdot|\mathbf{s}_t)]\right] + \Delta. && \text{(linearity of expectation)}
\end{aligned}
$$

Because we have assumed $F(x) = R(x)$ for all x terminal (condition (2)), and we have $Z = \sum_{\mathbf{x}\in\mathcal{X}} F(\mathbf{x})$ clearly from the definitions, the quantity Δ is independent of the choice of Markovian flow. Therefore, maximizing $\mathcal{H}[F]$ is equivalent to maximizing the expected entropy of P_B.

Finally, notice that every complete trajectory $s_0\to\ldots\to s_n$ passes through exactly one state $\mathbf{s}_d$ with $\mathcal{H}[P_B^\circ(\cdot|\mathbf{s}_d)] = \log d$ for each $d = 1,\ldots,D$, and that

$$\mathcal{H}[P_B(\cdot|\mathbf{s}_d)] \le \mathcal{H}[P_B^\circ(\cdot|\mathbf{s}_d)]$$

with equality if P_B is uniform over the parents of $\mathbf{s}_d$. Thus $\mathcal{H}[F]$ is maximized when $P_B(\cdot|\mathbf{s}) = P_B^\circ(\cdot|\mathbf{s})$ for all $\mathbf{s}$. □

Then we prove the Proposition 2:

Proposition. *If a GFlowNet fits the reward perfectly,* i.e.*, satisfies (2), then $A_{\tau,\tau'}(\mathbf{x}\to\mathbf{x}') = 1$, so the MH rejection step will always accept the proposal.*

Proof. Recall that the acceptance probability for a move from $\mathbf{x}$ to $\mathbf{x}'$ along a reverse trajectory τ and a forward trajectory τ' is given by

$$A_{\tau,\tau'}(\mathbf{x}\to\mathbf{x}') \triangleq \min\left(1, e^{\mathcal{E}_\phi(\mathbf{x})-\mathcal{E}_\phi(\mathbf{x}')}\frac{P_B(\tau|\mathbf{x})P_F(\tau')}{P_B(\tau'|\mathbf{x}')P_F(\tau)}\right).$$

According to Eq. (21) of Malkin et al. (2022), for a GFlowNet satisfying the reward matching constraint (2) with respect to a reward function R, we have $R(\mathbf{x})P_B(\tau|\mathbf{x})P_F(\tau') = R(\mathbf{x}')P_B(\tau'|\mathbf{x}')P_F(\tau)$. Elementary algebraic manipulation, and substituting $R(\mathbf{x}) = e^{-\mathcal{E}(\mathbf{x})}$, yields that $A_{\tau,\tau'}(\mathbf{x}\to\mathbf{x}') = 1$. □

C. More about experiments

C.1. Ising models

For the architecture of the GFlowNet, we use a four-layer MLP with 256 units in each hidden layer. We used Adam optimizer and batch size 256 to train EB-GFN and the baselines. For baselines, we use 100 steps of MCMC computation, *i.e.*, PCD-100.

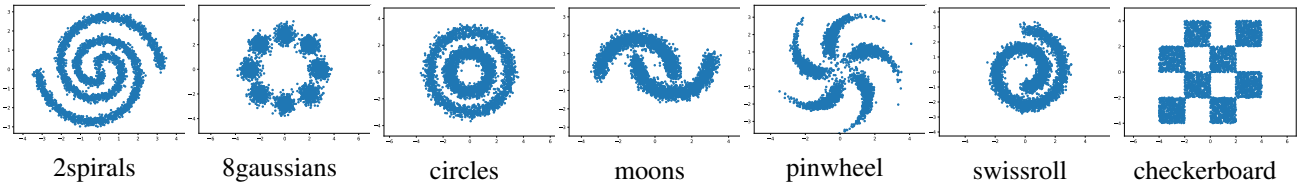

Figure C.1. Visualization of samples for synthetic problems from ground truth.

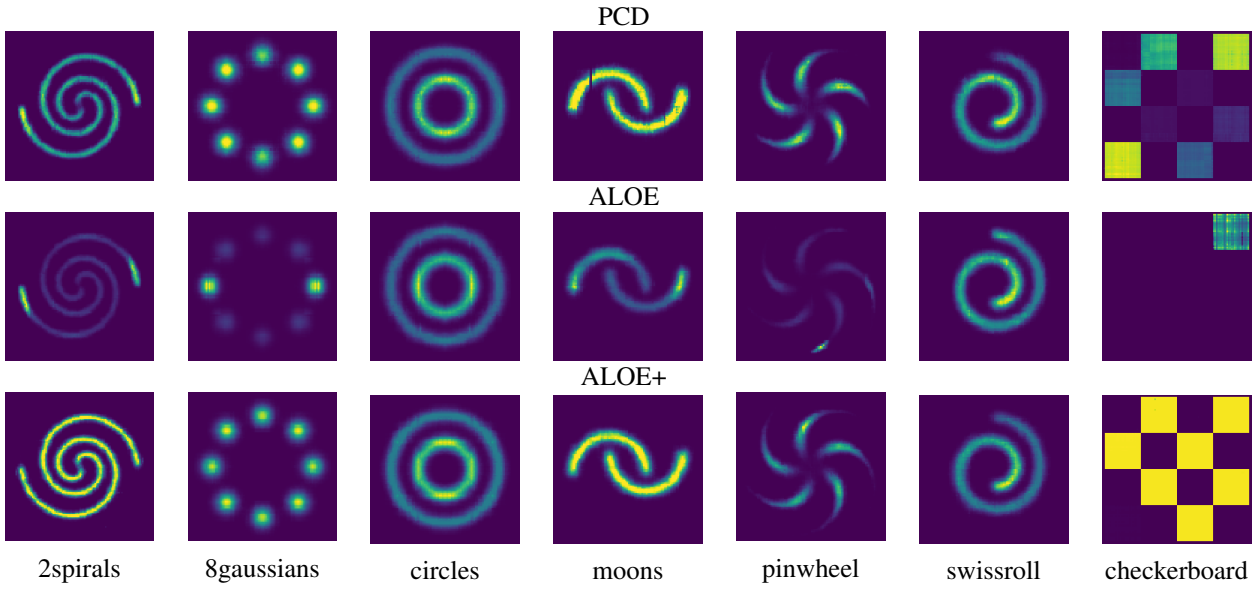

Figure C.2. Visualization of the learned energy function from different baseline methods.

For all methods, we stopped the training when RMSE between ground truth and learnt J reached its minimal value. We report the best result for each setting with a same hyperparameter searching protocol for all three methods. We search the learning rate of energy function in $\{1 \times 10^{-4}, 5 \times 10^{-4}\}$, the learning rate of GFlowNet in $\{1 \times 10^{-3}, 1 \times 10^{-2}\}$, the coefficient ℓ_1 regularization of J in $\{0.01, 0.02, 0.05, 0.1\}$. We keep other hyperparameters to be consistent with Grathwohl et al. (2021b). Notice that the benefit of EB-GFN is most clear when $\sigma < 0$. This matches our hypothesis that GFlowNet has a good inductive bias for modeling multimode distributions (when $\sigma > 0$, the lattice Ising model only has two modes, namely when the configuration is all $+1$ or all -1).

C.2. Synthetic tasks

This synthetic task is also adopted by previous works such as Grathwohl et al. (2019); Dai et al. (2020). We keep a consistent setting with Dai et al. (2020) unless specified. The data is first generated by an infinite data oracle as 2D floating-points values. They are then turned into 16-bit Gray code. This problem is challenging itself even without the existence of Gray code transformation, which is highly nonlinear. We use a 4 layer MLP with 256 hidden dimension and ELU activation (Clevert et al., 2016) as the energy function. The training of this energy function lasts 10^5 steps. An Adam optimizer with 1×10^{-3} learning rate is used to update the EBM. The batch size is 128. For PCD baseline, we use 10 steps Gibbs sampling to generate negative samples, and choose the best results from three different replay buffer re-initialization rate: $\{0.05, 0.1, 1\}$. For ALOE and ALOE+, we keep the same configuration as in Dai et al. (2020), where the former uses a naive 32-dim multinomial initial distribution proposal, and the latter use an autoregressive model which contains 32 MLPs that each has three layers with 512 hidden dimension. Both methods have an editor network and stop policy network, either of which is three layer MLP with 512 hidden feature. For the EB-GFN algorithm, the policy network is a similar three layer MLP (the forward policy and backward policy share the same first layers and differ in the last layer), and the output dimension is $3 \times 32 = 96$. The GFlowNet is optimized with an Adam optimizer, where the learning rate is 1×10^{-3}. We use an equal mix of $P_F(\tau)$ and $P_B(\tau|\mathbf{x})$ to generate training trajectories for trajectory balance objective (*i.e.*, set $\alpha = 0.5$ in Algorithm 1). For the back-and-forth proposal, we set K to linearly increase from 1 to D through the training process.

Table C.1. Experiment results with seven 2D synthetic problems. We display the MMD with linear kernel (in units of 1×10^{-4}). Notice that ALOE+ uses a thirty times larger parametrization than ALOE and EB-GFN.

Metric	Method	2spirals	8gaussians	circles	moons	pinwheel	swissroll	checkerboard
MMD↓	PCD	48.22	19.28	6.029	18.72	10.48	29.45	64.05
	ALOE	464.8	2240	10.18	557.1	810.4	**5.001**	1376
	ALOE+	**3.802**	**1.647**	12.54	9.181	32.72	15.01	264.1
	EB-GFN	9.705	10.22	**4.056**	**1.899**	**1.298**	10.23	**27.18**

Table C.2. Experiment results of both the EBM and GFlowNet in EB-GFN framework with seven 2D synthetic problems. We display the NLL and MMD with exponential kernel (in units of 1×10^{-4}).

Metric	Method	2spirals	8gaussians	circles	moons	pinwheel	swissroll	checkerboard
NLL↓	EBM	20.061	19.984	20.568	19.736	19.574	20.161	20.683
	GFlowNet	20.050	19.982	20.546	19.732	19.554	20.146	20.696
MMD↓	EBM	0.585	0.068	0.160	0.186	1.298	0.459	9.138
	GFlowNet	0.583	0.531	0.305	0.121	0.492	0.274	1.206

The NLL computation of GFlowNet follows the method described in §3.2. We set the value of M to be 100, which is large enough to converge (as a reference, for checkerboard dataset, the NLL is 20.695967 when $M = 10$, 20.695692 when $M = 50$, 20.695967 when $M = 100$, 20.695583 when $M = 500$, 20.695490 when $M = 1000$). The number of samples is set to 10^5, which is also enough for convergence in a similar sense.

To help better understanding the oracle of this task, we visualize the ground truth samples in Fig. C.1. We can see that EB-GFN could generate samples very close to these true data. We also plot the visualization of the baselines' energy function in Figure C.2. It demonstrates that ALOE actually has a hard time modeling multimode distribution without the help of a large initial proposal model, as ALOE+ does.

In Dai et al. (2020), the authors mentioned using Hamming kernel MMD, while in their public code linear MMD is adopted. Further, in their public implementation, the MMD result is calculated within a fixed group of 4000 samples. The variance of such a calculation results in many results in their experimental table being negative (note that MMD is a non-negative metric in theory). Based on these considerations, we choose to a more commonly adopted exponential Hamming kernel with 0.1 bandwidth in Table 2. Besides, we report the average of 10 repeat results, each with 4000 samples. To make a fair comparison, we also report the results given by linear kernel MMD in Table C.1 which is also used in ALOE public code. We can see that our algorithm keeps being state-of-the-art, and surpasses both PCD and the basic ALOE method on all datasets except *swissroll*.

One interesting property of the proposed EB-GFN framework, is that we can get samples either from the resulting GFlowNet (by sampling with the forward policy) or the learned EBM (by sampling with MCMC). Theoretically, GFlowNet would benefit more from its inductive bias as we discussed in §1, but we ideally want both models to achieve good performance. To this end, we also track the performance of the learned EBM. We find the learned EBM shares similar performance with the GFlowNet. The comparison is shown in Table C.2. On average, the EBM expresses slightly worse NLL and MMD than the GFlowNet, but is still very competitive if compared with other baseline methods. This can also demonstrate the benefit of the GFlowNet prior. As a result, we hypothesize that *the EB-GFN algorithm can achieve a good GFlowNet even with a not-so-good reward function*. This is because GFlowNet only needs the reward function to be *relatively* accurate with respect the true target distribution. We point out an interesting analogy to this phenomenon in reinforcement learning: a policy can have good performance even when the agent has learned a not-so-good Q function (Sutton & Barto, 2005; Bengio et al., 2020).

Ablation study on synthetic tasks. For completeness, we conduct ablation study to understand the importance of two features in EB-GFN algorithms: (1) backward trajectory sampling in GFlowNets training distribution mentioned in §3.2, and (2) the back-and-forth proposal proposed in §3.3. We do experiments on *checkerboard* and *moons* tasks. We first remove the usage of backward training samples and only use $\tau \sim P_F(\tau)$ to train the GFlowNet. The GFlowNet NLL on *moons* becomes 19.746 from 19.732, and Hamming exponential MMD becomes 0.342 from 0.121 (in units of 1×10^{-4}). For

checkerboard, the NLL becomes 20.709 from 20.696 and the MMD becomes 2.648 from 1.206 (in units of 1×10^{-4}). This shows removing the backward trajectory feature would only do little harm to the performance. For the second part, once we remove the back-and-forth proposal and always use $K = D$, the training loss of EB-GFN quickly diverges on both tasks. This indicates the suggested proposal trick is crucial to a reasonable optimization landscape.

Understanding the learned backward policy. In Fig. C.3, we give a visualization to show that the learned erasure policy $P_B(\cdot|\cdot;\boldsymbol{\theta})$ is meaningful. The GFlowNet-damaged samples have a clear visual structure that, interestingly, indicates that the several highest-magnitude bits in the Gray code are the first to be deleted by P_B and, correspondingly, are the last to be generated by a forward policy that minimizes the trajectory balance loss jointly with this P_B.

C.3. Discrete image modeling

We explain the details of our discrete image modeling task here. Discrete image modeling with EBM is a hard problem, and pure PCD training would diverge if the number of MCMC steps is not large enough or if there is no replay buffer trick to help training. This is not the case with continuous circumstances (Nijkamp et al., 2020). In this part, we follow the settings of Grathwohl et al. (2021b), which are stated below. We use Adam optimizer with 1×10^{-4} learning rate to update the energy function. The batch size is set to be 100 and the training lasts for 5×10^4 steps. The energy function is an MLP with 256 hidden units and three hidden layers[5]. We do not use exponential moving average for simplicity. The GFlowNet is optimized with an Adam optimizer, where the learning rate is 1×10^{-3}. The validation of EBM likelihood is achieved with 300000 step annealed importance sampling (AIS). GFlowNet is modelled as an MLP with three hidden layers and 512 hidden units. In this part we take the canonical design of backward policy, as we find that it could stabilize the training process. We choose the checkpoint which has the best validation result, and report the corresponding test set performance. The same GFlowNet training techniques are utilized as in synthetic tasks: we use an equal mix of $P_F(\tau)$ and $P_B(\tau|\mathbf{x})$ to generate training trajectories for trajectory balance objective (*i.e.*, set $\alpha = 0.5$ in Algorithm 1). For the back-and-forth proposal, we set K to linearly increase from 1 to D through the training process. We also do ablation study on these features in this task, and we get different results from §C.2. To be precise, in the static mnist experiment, we find that these two techniques are both important: EB-GFN would diverge without either trick. This partially reflects the difficulty of this task.

We also find that introducing LayerNorm (Ba et al., 2016) into the forward policy network architecture is of great benefit to the generative modeling performance. We add a LayerNorm after each linear layer except the last one. The negative likelihood of this ablation is presented in the following table.

Method / Dataset	Omniglot	Silhouettes	Static MNIST	Dynamic MNIST
w/o LayerNorm	112.59	185.57	102.43	105.75
w/ LayerNorm	104.88	174.48	89.48	88.76

[5]We do not use the ResNet-18 backbone for EBM, because we find that it takes 2 weeks for the training of Gibbs-With-Gradients to finish with the original code of GWG public repo.

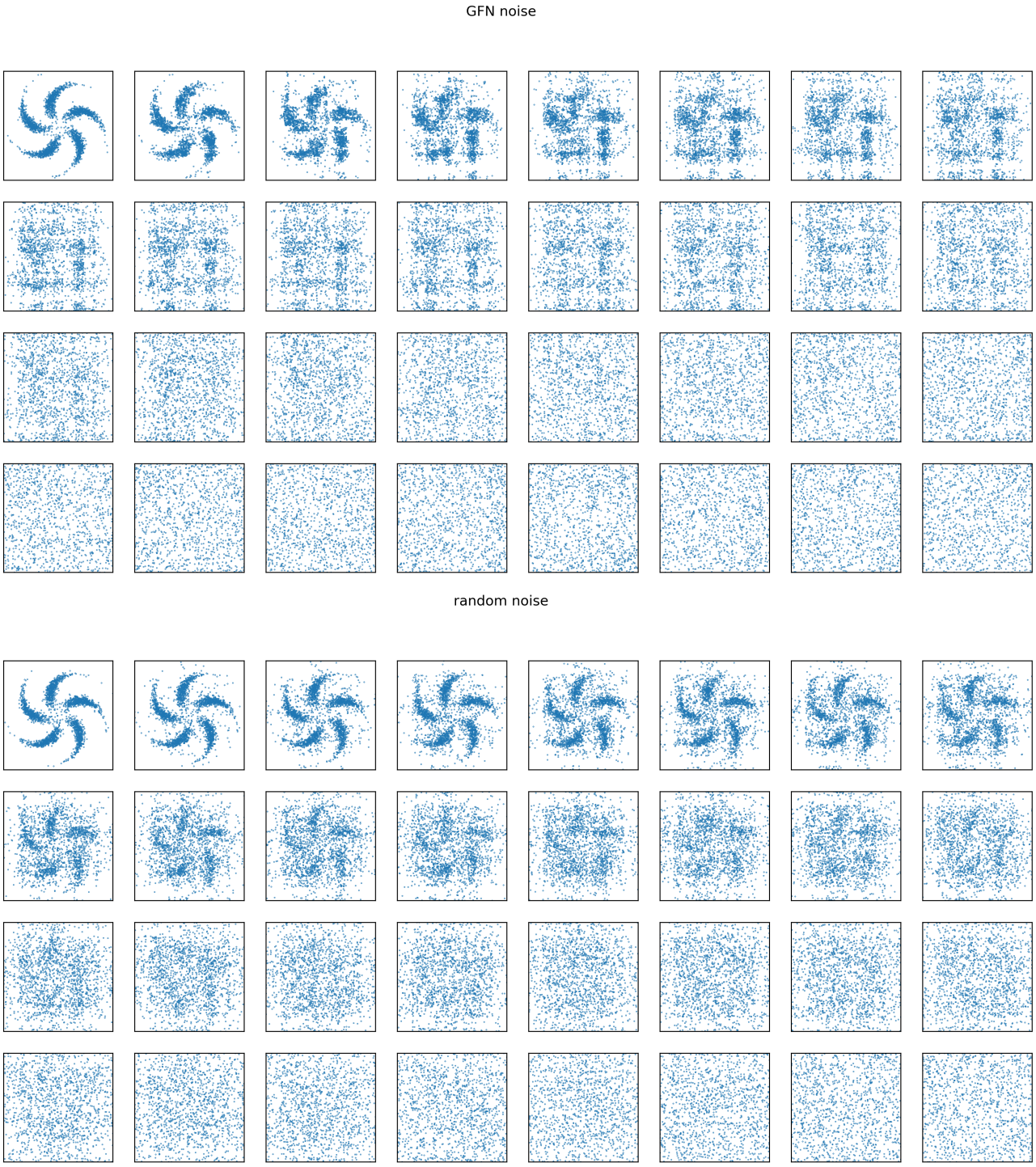

Figure C.3. We begin with real data from the *pinwheel* dataset (top left) and take increasing numbers of steps (up to 31) from either the learned backward policy $P_B(\cdot|\cdot;\boldsymbol{\theta})$ of a GFlowNet *(top)* or a uniform backward policy P_B° *(bottom)*, then randomly (i.i.d. Bernoulli) pick new 0/1 values for the voided bits and visualize the resulting damaged samples.