

1 Introduction

For my thesis¹ at Imperial I explored the interaction between discrete diffusion models, learned samplers and reinforcement learning. This paper summarizes ideas for future extensions of this work. The core question that this paper attempts to answer is: Can we frame discrete diffusion as a the continuous limit of a DAG, and thereby transform the method to a learned sampler trainable with GFlowNet objective functions?

2 Setting

In discrete flow matching, the task consists of transforming a source distribution $p_0(\mathbf{x}_0)$ to a target distribution $p_1(\mathbf{x}_1)$ [2]. Assume that the sample space is discrete and D -dimensional, i.e. $\mathbf{x}_t \in \mathcal{X} = \{1, \dots, S\}^D$, where S indicates the number of states that each dimension can assume, and naturally $t \in [0, 1]$. We model the transformation from p_0 to p_1 via a probability path p_t as a continuous time Markov chain (CTMC). We assume that generation from the CTMC is via sampling each dimension independently from a generating probability velocity $u_t^i(z_t^i, \mathbf{x}_t)$ starting at $\mathbf{x}_0 \sim p_0(\mathbf{x}_0)$ according to the following Euler discretisation

$$x_{t+h}^i \sim \delta_{x_t^i}(\cdot) + hu_t^i(\cdot, \mathbf{x}_t) \quad (1)$$

for some step size h . We generally require that Equation 1 define a valid probability mass function, which amounts to $\sum_{z^i} u_t^i(z^i, \mathbf{x}_t) = 0$ and $u_t^i(z^i, \mathbf{x}_t) \geq 0 \forall z^i \neq x^i$. The core objective of my proposed project is to learn a generating probability velocity $u_t^i(z_t^i, \mathbf{x}_t)$ to generate samples from p_1 , which is a distribution implicitly defined by a target *energy* function $\mathcal{E}(\mathbf{x}_1)$ as $p_1 = \frac{\exp(-\mathcal{E})}{Z}$, by sampling according to Equation 1 starting from p_0 . Importantly, assume no access to training data from p_1 .

3 Discrete Flows as Continuous DAG

To make discrete flows compatible with the trajectory balance objective used to train GFlowNets we need to ensure that for any $h \in (0, 1)$ Equation 1 defines a valid Markovian flow on some directed acyclic graph (DAG) [1]. To be compatible with the notion of a Markovian flow [1], we require $p_0 = \delta_M$, wherein M is a fully masked state. Thus, we can view M as our root node. From here on, to avoid cycles, we can ‘index’ the mask augmented state space by time t by taking its Cartesian product with the interval $[0, 1]$. Then $\mathbf{x}_t \in \mathcal{X}_{M,t} = \{1, \dots, S, M\}^D \times [0, 1]$ and we obtain an ‘infinite’ node set $\mathcal{S} = \bigcup_t \mathcal{X}_{M,t}$. In the h -discretization then (denoting the now finite node set as \mathcal{S}_h), connect any pair of nodes between the j -th layer (corresponding to the slice $t = j \cdot h$) and $(j+1)$ -th layer (slice $t = (j+1) \cdot h$) to obtain an edge set \mathcal{A}_h , and hence a DAG $G = (\mathcal{S}_h, \mathcal{A}_h)$.

Note that even if $\mathbf{x}_t^i = \mathbf{x}_{t'}^i \forall i$, as long as $t \neq t'$, we consider these nodes as distinct. By construction, the only state with no incoming edges is then $x_0 = M \in \mathcal{A}$. The sinks, i.e. the states with no outgoing edges, are $\mathbf{x}_1 \in \mathcal{X}_{M,t}$ with the restriction $x^i \neq M \forall i$.

We now define a few terms and results, analogously to the definitions in [3]. First, define complete trajectories as $\tau = (\mathbf{x}_0 \rightarrow \mathbf{x}_h \rightarrow \dots \rightarrow \mathbf{x}_1)$, and the set \mathcal{T} of all such trajectories. The trajectory flow

¹<https://github.com/moritzhauschulz/samplingEBMs>

is then $F : \mathcal{T} \rightarrow \mathbb{R}_{\geq 0}$, and for each node \mathbf{x} we can define the node flow $F(s) = \sum_{s \in \tau} F(\tau)$ and for any edge in \mathcal{A} the edge flow

$$F(\mathbf{x}_t \rightarrow \mathbf{x}_{t+h}) = \sum_{\tau = (\dots \rightarrow \mathbf{x}_t \rightarrow \mathbf{x}_{t+h} \rightarrow \dots)} F(\tau). \quad (2)$$

This construction then satisfies detailed balance

$$F(\mathbf{x}_t) = \sum_{(\mathbf{x}_{t-h} \rightarrow \mathbf{x}) \in \mathcal{A}} F(\mathbf{x}_{t-h} \rightarrow \mathbf{x}_t) = \sum_{(\mathbf{x}_t \rightarrow \mathbf{x}_{t+h}) \in \mathcal{A}} F(\mathbf{x}_t \rightarrow \mathbf{x}_{t+h}). \quad (3)$$

By applying appropriate normalization, we obtain a distribution over trajectories:

$$P(\tau) = \frac{1}{Z}, \quad Z = F(\mathbf{M}) = \sum_{\tau \in \mathcal{T}} F(\tau) \quad (4)$$

A trajectory flow is said to be Markovian if there exist action distributions $P_F(\cdot | \mathbf{x}_t)$ over the children of each nonterminal state \mathbf{x}_t such that the distribution P has a factorisation

$$P(\tau = (\mathbf{M} \rightarrow \dots \rightarrow \mathbf{x}_1)) = \prod_{j=1}^{1/h} P_F(\mathbf{x}_{j \cdot h} | \mathbf{x}_{(j-1) \cdot h}). \quad (5)$$

Further, [1] show that the existence of distributions $P_B(\cdot | \mathbf{x}_t)$ over the parents of each non-initial state \mathbf{x}_t with

$$P(\tau = (\mathbf{M} \rightarrow \dots \rightarrow \mathbf{x}_1) | \mathbf{x}_1) = \prod_{j=1}^{1/h} P_B(\mathbf{x}_{(j-1) \cdot h} | \mathbf{x}_{j \cdot h}). \quad (6)$$

(i.e. the existence of a ‘backward’ flow) is an equivalent criterion. It is easy to see that in this way, taking the limit $h \rightarrow 0$, we may view the CTMC as the continuous limit of our DAG over which $u_t^i(\cdot, \mathbf{x}_t)$ defines a marginal flow at the j -th layer with $t = j/h$. Hence, we may think about formulating objectives to learn P_F^θ , or equivalently u_t^θ , as a neural network with parameters θ . We make this rigorous in the next section, where the usefulness of the backward formulation will become clearer.

3.1 Training Discrete Flows with Trajectory Balance

A GFlowNet according to [1], but adapted to our context, can be specified generally as follows:

- A model capable of providing the initial state flow $Z = F(\mathbf{x}_0)$ as well as the forward action distributions $P_F(\mathbf{x}_t | \mathbf{x}_{t-1})$ for any nonterminal state \mathbf{x}_t (and therefore, by the above, uniquely but possibly in an implicit way determining a Markovian flow F);
- An objective function, such that if the model is capable of expressing any action distribution and the objective function is globally minimized, then the constraint $F(\mathbf{x}_1) = \exp(-\mathcal{E}(\mathbf{x}_1)) \quad \forall \mathbf{x}_1 \in \mathcal{X}$ is satisfied for the corresponding Markovian flow F .

For an optimally trained GFlowNet, the forward policy given by $P_F(\cdot | \mathbf{x}_t)$ can then be used to sample from $p_1 = \frac{\exp(-\mathcal{E})}{Z}$. The authors of [3] then observe that by combining Equation 3, Equation 6 and

Equation 5 we obtain for a valid Markovian flow F the following ‘*trajectory balance constraint*’ for a given trajectory $\tau = (\mathbf{x}_0 \rightarrow \mathbf{x}_h \rightarrow \dots \rightarrow \mathbf{x}_1)$:

$$Z \prod_{j=1}^{1/h} P_F(\mathbf{x}_{j \cdot h} | \mathbf{x}_{(j-1) \cdot h}) = F(\mathbf{x}_1) \prod_{j=1}^{1/h} P_B(\mathbf{x}_{(j-1) \cdot h} | \mathbf{x}_{j \cdot h}), \quad (7)$$

Where in our case $F(\mathbf{x}_1) = \exp(-\mathcal{E}(\mathbf{x}_1))$. This can be readily transformed into a training objective, by parameterizing P_F and P_B as neural networks. Recall that our goal is to train a CTMC, i.e. we are interested in extracting probability velocities $u_{F,t}^i(z^i, \mathbf{x}_t; \theta)$ and $u_{B,t}^i(z^i, \mathbf{x}_t; \theta)$ so that

$$P_F(\cdot | \mathbf{x}_t; \theta) = \prod_{i=1}^D \delta_{x_t^i}(\cdot) + hu_{F,t}^i(\cdot, \mathbf{x}_t; \theta) \quad (8)$$

$$P_B(\cdot | \mathbf{x}_t; \theta) = \prod_{i=1}^D \delta_{x_t^i}(\cdot) + hu_{B,t}^i(\cdot, \mathbf{x}_t; \theta) \quad (9)$$

This parameterises the forwards and backwards policies implicitly, and independently of h . Using this implicit parameterization, I propose to train the (forward and backward) generating velocities $u_{F,t}^i(z^i, \mathbf{x}_t; \theta)$ and $u_{B,t}^i(z^i, \mathbf{x}_t; \theta)$ by optimizing the trajectory balance loss due to [1]. For trajectory τ , this is given by:

$$\mathcal{L}_{TB}(\tau) = \left(\log \frac{Z_\theta \prod_{j=1}^{1/h} P_F(\mathbf{x}_{j \cdot h} | \mathbf{x}_{(j-1) \cdot h; \theta})}{\exp(-\mathcal{E}(\mathbf{x}_1)) \prod_{j=1}^{1/h} P_B(\mathbf{x}_{(j-1) \cdot h} | \mathbf{x}_{j \cdot h; \theta})} \right)^2. \quad (10)$$

When this is minimized, we may expect to sample from p_1 by integrating the learned forward CTMC (or discrete flow) forward in time.

4 Open Questions

1. Is the training of discrete flows with a trajectory balance objective tractable, given the vast extend of the DAG?
2. If so, should training be based on on- or off-policy trajectories, with the latter being cheaper to generate and the former less likely to suffer from the curse of dimensionality?
3. If not, can we make it tractable by reducing the size of the DAG, for instance by only allowing each state to be unmasked once, without compromising the scalability of our approach?
4. How does this approach compare to recent developments such as discrete denoising posterior prediction [4], which can be viewed as a special case of learned sampling?

References

- [1] Bengio, Y., S. Lahlou, T. Deleu, E. J. Hu, M. Tiwari, and E. Bengio (2023). Gflownet foundations.
- [2] Gat, I., T. Remez, N. Shaul, F. Kreuk, R. T. Q. Chen, G. Synnaeve, Y. Adi, and Y. Lipman (2024). Discrete flow matching.
- [3] Malkin, N., M. Jain, E. Bengio, C. Sun, and Y. Bengio (2023). Trajectory balance: Improved credit assignment in gflownets.
- [4] Rector-Brooks, J., M. Hasan, Z. Peng, Z. Quinn, C. Liu, S. Mittal, N. Dziri, M. Bronstein, Y. Bengio, P. Chatterjee, A. Tong, and A. J. Bose (2024). Steering masked discrete diffusion models via discrete denoising posterior prediction.