Unifying Generative Models with GFlowNets

Dinghuai Zhang [1] **Ricky T. Q. Chen** [2] **Nikolay Malkin** [1] **Yoshua Bengio** [1]

Abstract

There are many frameworks for deep generative modeling, each often presented with their own specific training algorithms and inference methods. We present a short note on the connections between existing deep generative models and the GFlowNet framework (Bengio et al., 2021b), shedding light on their overlapping traits and providing a unifying viewpoint through the lens of learning with Markovian trajectories. This provides a means for unifying training and inference algorithms, and provides a route to construct an agglomeration of generative models.

1. Preliminaries about GFlowNets

From a probabilistic modeling viewpoint, generative flow network (GFlowNet) (Bengio et al., 2021a) is one kind of generative model that aims to sample $\mathbf{x}$ in proportion to a given reward function $R(\mathbf{x})$. Concretely, a GFlowNet would sample a Markovian trajectory $\tau = (s_0, s_1, \ldots, s_n)$ with length n. If not specially specified, we use the notation $\mathbf{x} = s_n$ for the final state of the trajectory. This process has a natural connection to reinforcement learning (Sutton & Barto, 2005). All the states s construct a directed acyclic graph (DAG) in the latent space. Each trajectory starts from the same (abstract) initial state s_0 and runs to a different end point s_n. The flow function $F(\tau) \in \mathbb{R}_+$ defined in Bengio et al. (2021b) can be understood by an analogy of water amount in a water flow. Ideally, we want the amount of flow leading to $\mathbf{x}$ equals the given reward: $\sum_{\tau=(s_0,\cdots,\mathbf{x})} F(\tau) = R(\mathbf{x})$.

GFlowNet has several training criterion. We start from the flow matching condition (Eq. 1). Define $F(s,s') \triangleq \sum_{(s\to s')\in\tau} F(\tau)$ as the edge flow function and $F(s) \triangleq \sum_{s\in\tau} F(\tau)$ as the state flow function. It's easy to see that

$$\sum_{s} F(s,s') = \sum_{s''} F(s',s''), \tag{1}$$

as both side of the term equals $F(s')$. If one parametrizes the GFlowNet with the edge flow function $F_\theta(\cdot,\cdot)$, then Eq. 1 would be used as training loss (*i.e.*, $L(\theta,s') = \left(\sum_s F_\theta(s,s') - \sum_{s''} F_\theta(s',s'')\right)^2$).

The detailed balance condition of GFlowNet writes

$$F(s)P_F(s' \mid s) = F(s')P_B(s \mid s'), \tag{2}$$

where $P_F(s'|s)$ and $P_B(s|s')$ are referred to as the forward and backward policy for the transition between different states. We can separately parametrize three models for $F_\theta(\cdot), P_F(\cdot|\cdot;\theta), P_B(\cdot|\cdot;\theta)$. The detailed balance condition is closely related to the flow matching condition, in the sense that $P_F(s'|s) = F(s,s')/F(s)$ and $P_B(s|s') = F(s,s')/F(s')$. It suffices to determine a GFlowNet by determining its forward policy in theory.

What's more, the general trajectory balance condition of GFlowNet (Malkin et al., 2022) wants to match GFlowNet's forward trajectory $P_F(\tau)$ and the backward trajectory $P_B(\tau)$, where

$$P_F(\tau) = \prod_{t=0}^{n-1} P_F(s_{t+1}|s_t) \tag{3}$$

$$P_B(\tau) = \frac{R(\mathbf{x})}{Z}\prod_{t=0}^{n-1} P_B(s_t|s_{t+1}). \tag{4}$$

Here $Z = \sum_{\mathbf{x}} R(\mathbf{x})$ is the normalizing factor. We also write $P_B(\tau|\mathbf{x}) = \prod_{t=0}^{n-1} P_B(s_t \mid s_{t+1})$.

2. Hierarchical Variational Autoencoders

The evidence lower bound for bottom-up hierarchical VAEs (HVAEs) (Ranganath et al., 2016) reads

$$\log p(\mathbf{x}) \geq \mathrm{ELBO}(p,q) \tag{5}$$

$$\triangleq \mathbb{E}_{q(\mathbf{z}_{1:L}|\mathbf{x})}\left[\log p(\mathbf{x},\mathbf{z}_{1:L}) - \log q(\mathbf{z}_{1:L}|\mathbf{x})\right] \tag{6}$$

$$= \mathop{\mathbb{E}}_{q(\mathbf{z}_L|z_{L+1})\cdots q(\mathbf{z}_1|\mathbf{z}_2)}\left[\log p(\mathbf{z}_1) + \sum_{i=1}^{L}\log\frac{p(\mathbf{z}_{i+1}|\mathbf{z}_i)}{q(\mathbf{z}_i|\mathbf{z}_{i+1})}\right], \tag{7}$$

where we denote $\mathbf{x} := \mathbf{z}_{L+1}$. It is well known that this hierarchical ELBO can also be represented as $\mathcal{D}_{\mathrm{KL}}\left(q(\mathbf{z}_{1:L}|\mathbf{x})\|p(\mathbf{z}_{1:L}|\mathbf{x})\right)$, where $p(\mathbf{z}_{1:L}|\mathbf{x}) \propto$

[1]Mila - Quebec AI Institute [2]Facebook AI Research. Correspondence to: Dinghuai Zhang <dinghuai.zhang@mila.quebec>.

ICML 2022 Beyond Bayes workshop. Copyright 2022 by the author(s).

$p(\mathbf{x}|\mathbf{z}_{1:L})p(\mathbf{z}_{1:L})$ and $\mathbb{D}_{\mathrm{KL}}$ denotes the KL divergence. In GFlowNets, we also aim to match the forward trajectory policy which ends with data $\mathbf{x}$ with the corresponding backward trajectory policy, *i.e.*, $P_F(\tau) \approx P_B(\tau)$, conditioning on the event $\{\mathbf{x} \in \tau\}$, *i.e.*, $\mathbf{x}$ is the last state of τ. Note that we have $P(\mathbf{x}|\tau) = \mathbb{1}\{\mathbf{x} \in \tau\}$. A slight difference between HVAEs and GFlowNets is that $\mathbf{z}_{1:L}$, the latent variables of the hierarchical VAE, does not include $\mathbf{x}$.

Observation 1. *HVAE is a special kind of GFlowNets in the following sense: each trajectory is in the form of* $\tau = (\mathbf{z}_1, \mathbf{z}_2, \cdots, \mathbf{z}_L, \mathbf{z}_{L+1} = \mathbf{x})$. *The decoder of VAE, which samples* $\mathbf{z}_1 \to \cdots \to \mathbf{z}_L \to \mathbf{z}_{L+1}$, *corresponds to the forward policy; the encoder samples* $\mathbf{x} \to \mathbf{z}_L \to \cdots \to \mathbf{z}_1$, *and corresponds to the backward policy.*

With $\mathbf{x} = \mathbf{z}_{L+1}$, we could then write

$$P_F(\tau) = p(\mathbf{z}_1) \prod_{i=1}^{L} p(\mathbf{z}_{i+1}|\mathbf{z}_i), \tag{8}$$

$$P_B(\tau) = p_d(\mathbf{x}) \prod_{i=1}^{L} q(\mathbf{z}_i|\mathbf{z}_{i+1}), \tag{9}$$

where $p_d(\mathbf{x}) = \frac{R(\mathbf{x})}{Z}$ is the true target density. We can see the i-th state in a GFlowNet trajectory s_i corresponds to $\mathbf{z}_i$. A subtle difference is that in Bengio et al. (2021b) we additionally have an abstract initial state s_0 which has no concrete meaning for some simplicity reasons. We ignore this difference here.

ZDH: forward traj and backward traj don't necessarily need to have same length – which means the HVAE's encoder and decoder don't need to have same number of layers

The following proposition reveals an equivalence between the two perspectives in Observation 1.

Proposition 2. *Training hierarchical latent variable models with the KL-trajectory balance is equivalent to training HVAEs by maximizing its ELBO.*

We remark that there is also top-down modeling for hierarchical VAE (Sønderby et al., 2016; Child, 2021), where the inference model takes the factorization form of $q(\mathbf{z}_1|\mathbf{x}) \prod_{i=1}^{L-1} q(\mathbf{z}_{i+1}|\mathbf{z}_i)$. However, we do not discuss it here as it does not have much connection to GFlowNets, although it has more application power in practice.

3. Diffusion Models & SDEs

Behavior of deep latent variable model in its infinite depth regime is studied by Tzen & Raginsky (2019). In short, if the encoder and decoder take the following form

$$p(\mathbf{z}_{i+1}|\mathbf{z}_i) = \mathcal{N}\left(\mathbf{z}_{i+1}; \mathbf{z}_i + h\mathbf{f}_i(\mathbf{z}_i), hg_i^2)\right), \tag{10}$$

$$\begin{aligned} q(\mathbf{z}_i|\mathbf{z}_{i+1}) = \mathcal{N}(\mathbf{z}_i; \mathbf{z}_{i+1} + h(-\mathbf{f}_{i+1}(\mathbf{z}_{i+1}) \quad & (11)\\ + g_{i+1}^2 \nabla \log p_{i+1}(\mathbf{z}_{i+1})), hg_{i+1}^2), \quad & (12) \end{aligned}$$

where we assume all $\mathbf{z}_i$ have the same number of dimensionality as $\mathbf{x}$ and $h = 1/L$, then the hierarchical model is equivalent to a stochastic process in its diffusion limit ($h \to 0$). Huang et al. (2021); Kingma et al. (2021); Song et al. (2021) also study connections between hierarchical variational inference on latent variable models and diffusion.

Consider a stochastic differential equation (SDE) (Øksendal, 1985) and its reverse time SDE (Anderson, 1982)

$$\mathrm{d}\mathbf{x} = \mathbf{f}(\mathbf{x})\,\mathrm{d}t + g(t)\,\mathrm{d}\mathbf{w}_t, \tag{13}$$

$$\mathrm{d}\bar{\mathbf{x}} = \left[\mathbf{f}(\bar{\mathbf{x}}) - g^2(t)\nabla_{\bar{\mathbf{x}}} \log p_t(\bar{\mathbf{x}})\right] \mathrm{d}\bar{t} + g(t)\,\mathrm{d}\bar{\mathbf{w}}_t, \tag{14}$$

where $\mathbf{f}(\cdot)$ and $g(\cdot)$ are given and $\mathbf{x}, \mathbf{w} \in \mathbb{R}^D$, and $\bar{\mathbf{x}}, \bar{t}, \bar{\mathbf{w}}_t$ denote the reverse time version of $\mathbf{x}, t, \mathbf{w}_t$. We define $\mathbb{P}_h(\mathbf{x}_{t+h}|\mathbf{x}_t)$ to be the transition kernel induced by the SDE in Eq. 13: $\mathbf{x}_{t+h} = \mathbf{x}_t + \int_t^{t+h} f(\mathbf{x}_\tau)\,\mathrm{d}\tau + \int_t^{t+h} g(\tau)\,\mathrm{d}\mathbf{w}_\tau$, where h denotes an infinite small time step. We then make the following observation.

Observation 3. *SDE is a special case of GFlowNets, in the sense that GFlowNet states take the time-augmented form* $s = (\mathbf{x}_t, t)$ *for some* t*, the SDE models the forward policy of GFlowNets (*i.e.*, how states should move forward) while the reverse time SDE models the backward policy of GFlowNets (*i.e.*, how states should move backward). In this case, a trajectory is in the form of* $\{\mathbf{x}_t\}_{t=0}^1$.

Note that we cannot directly treat any $\mathbf{x}_t$ to be GFlowNet state, because theory of GFlowNets requires the graph of all latent states is a DAG (*i.e.*, cannot possess circles). This state augmenting operation also induces that in SDE case, the latent state graph of GFlowNets could not be arbitrary DAG, since any $(\mathbf{x}, t) \to (\mathbf{x}', t')$ with $t \geq t'$ is forbidden. Without loss of generality, we assume $\mathbf{x}_t$ state already contain time stamp in itself in the following context[1].

We next point out an analogy between stochastic processes property and GFlowNet property:

Observation 4. *Such property of stochastic process*

$$\begin{aligned} &\int p(\mathbf{x}_{t-h}, t-h)\mathbb{P}_h(\mathbf{x}_t|\mathbf{x}_{t-h})\,\mathrm{d}\mathbf{x}_{t-h} \\ &\quad = \int p(\mathbf{x}_t, t)\mathbb{P}_h(\mathbf{x}_{t+h}|\mathbf{x}_t)\,\mathrm{d}\mathbf{x}_{t+h} = p(\mathbf{x}_t, t), \end{aligned} \tag{15}$$

could be interpreted as GFlowNets flow matching condition:

$$\sum_s F(s, s') = \sum_{s''} F(s', s'') \triangleq F(s'), \tag{16}$$

for any s'*, where we also have* $F(s, s') = F(s)P_F(s'|s)$.

We point out that Eq. 15 is a standard starting point for deriving the Fokker-Planck equation:

[1] This is equivalent to define $\tilde{\mathbf{x}}_t = (\mathbf{x}_t, t)$ and conduct discussion with $\tilde{\mathbf{x}}_t$ instead.

Proposition 5 (Øksendal (1985)). *Taking the limit as $h \to 0$, Eq. 15 implies*

$$\partial_t p(\mathbf{x}, t) = -\nabla_{\mathbf{x}} \left(p(\mathbf{x}, t)\mathbf{f}(\mathbf{x}, t)\right) + \frac{1}{2}\nabla^2_{\mathbf{x}} \left(p(\mathbf{x}, t) g^2(\mathbf{x}, t)\right). \tag{17}$$

Equivalence between detailed balance and score matching. At the end of this section, we investigate such a setting where we want to model our own reverse process:

$$\mathrm{d}\mathbf{x} = \left[\mathbf{f}(\mathbf{x}) - g^2(t)\mathbf{s}(\mathbf{x}, t)\right] \mathrm{d}\bar{t} + g(t)\,\mathrm{d}\bar{\mathbf{w}}_t, \tag{18}$$

where $\mathbf{s}(\cdot,\cdot) : \mathbb{R}^D \times \mathbb{R} \to \mathbb{R}^D$ is a neural network, and $\mathbf{f}(\cdot)$ and $g(\cdot)$ are given[2]. We propose to use the detailed balance criterion of GFlowNet to learn this neural network. From the above discussion, we can see there is an analogy about $F(s_t) \approx p_t(\mathbf{x})$. We show the validity of such strategy in the following proposition.

Proposition 6. *GFlowNets' detailed balance condition*

$$\lim_{h\to 0} \frac{1}{\sqrt{h}} (\log p_t(\mathbf{x}_t) + \log P_F(\mathbf{x}_{t+h}|\mathbf{x}_t) - \log p_{t+h}(\mathbf{x}_{t+h}) + \log P_B(\mathbf{x}_t|\mathbf{x}_{t+h})) = 0, \tag{19}$$

$(\forall \mathbf{x}_t \in \mathbb{R}^D, \forall t \in (0, 1))$ *is equivalent to*

$$\boldsymbol{\epsilon}^\top \left(\mathbf{s}(\mathbf{x}_t, t) - \nabla_{\mathbf{x}} \log p_t(\mathbf{x})\right) = 0, \forall \boldsymbol{\epsilon}, \mathbf{x}_t \in \mathbb{R}^D, \forall t \in [0, 1],$$

which is the optimal solution to score matching:

$$\min_{\mathbf{s}} \mathbb{E}_{\mathbf{x}\sim p_t} \mathbb{E}_{\boldsymbol{\epsilon}} \left[\boldsymbol{\epsilon}^\top \nabla_{\mathbf{x}} \mathbf{s}(\mathbf{x}) \boldsymbol{\epsilon} + \frac{1}{2}\left(\boldsymbol{\epsilon}^\top \mathbf{s}(\mathbf{x}, t)\right)^2\right], \forall t.$$

4. Exact Likelihood Models

Autoregressive (AR) models sample $p(\mathbf{x}_{1:i+1}|\mathbf{x}_{1:i})$ sequentially for every dimension i to generate a vector data $\mathbf{x}$. In Zhang et al. (2022), the authors use an AR-like (with a learnable ordering) model to parametrize the GFlowNet. Indeed, if we define every (forward) action of GFlowNets is to add one pixel to the current state, and the backward policy is to turn one pixel into so-called "void" state, then AR models could be seen as GFlowNets. As a formal example, we analyze AR models with natural ordering.

Observation 7. *Autoregressive model is a special kind of GFlowNets when*

- $s_t := \mathbf{x}_{1:t}$ *is the GFlowNet state;*
- $P_F(s_{t+1}|s_t) = p(\mathbf{x}_{1:i+1}|\mathbf{x}_{1:i})$ *is the forward policy;*
- $P_B(s_t|s_{t+1}) = \delta\{s_t$ *is the first t dimensions of* $s_{t+1}\}$,

where $\delta\{\cdot\}$ is the Dirac Delta distribution for continuous variables, and is the indicator function for discrete cases.

This modeling makes the latent graph of the GFlowNet to be a tree; alternatively, if we allow learnable ordering as in Zhang et al. (2022), the latent space is a general DAG. ZDH: cite some arbitrary ordering AR papers

Normalizing flow (Dinh et al., 2015) (NF) is another way to sequentially construct desired data. It first samples $\mathbf{z}_1$ from a base distribution (usually the standard Gaussian), and then doing a series of invertible transformations $\mathbf{z}_{i+1} = \mathbf{f}(\mathbf{z}_i)$ until we finally get $\mathbf{x} := \mathbf{z}_{L+1}$, where L denotes the number of transformation layers.

Observation 8. *NF is a special kind of GFlowNets where deterministic forward and backward policies (except the first transition step), and $s_t = \mathbf{z}_t$ are GFlowNet states.*

In this scenario, NF is a GFlowNet whose forward policy is to sample from the base distribution at the first step (hence a stochastic step), and conduct deterministic invertible transformations afterwards. We next discuss maximum likelihood estimation (MLE) of such models.

About MLE training. AR models and NFs are usually trained with MLE. Albeit the likelihood of general GFlowNets is intractable, we can estimate its lower bound:

$$\log p_T(\mathbf{x}) = \int_{\mathbf{x}\in\tau} P_F(\tau)\,\mathrm{d}\tau \tag{20}$$

$$= \log \mathbb{E}_{P_B(\tau|\mathbf{x})} \left[\frac{P_F(\tau)}{P_B(\tau|\mathbf{x})}\right] \tag{21}$$

$$\geq \mathbb{E}_{P_B(\tau|\mathbf{x})} \left[\log \frac{P_F(\tau)}{P_B(\tau|\mathbf{x})}\right] \tag{22}$$

$$= -\mathcal{D}_{\mathrm{KL}} \left(P_B(\tau|\mathbf{x}) \| P_F(\tau)\right), \tag{23}$$

which could be again seen as one kind of trajectory balance as in Proposition 2. An IWAE-type bound (Burda et al., 2016) is also applicable. If the backward policy is a fixed one (*i.e.*, deterministic and without learnable parameters), we can directly use $\log P_B(\tau|\mathbf{x}) - \log P_F(\tau)$ as a sample based tractable training loss for $\mathbf{x} \in \mathcal{D}$, with $\tau \sim P_B(\tau|\mathbf{x})$. Furthermore, if we use AR natural ordering, then $P_B(\tau|\mathbf{x})$ only contains one trajectory, therefore $\log P_B(\tau|\mathbf{x}) - \log P_F(\tau)$ is the same thing as the MLE loss of AR models. On the other hand, if the backward policy is learnable, one may need REINFORCE (Williams, 2004) or reparametrization trick (Kingma & Welling, 2014) to optimize with regard to P_B, making it a variational distribution over trajectories.

We summarize that both AR models and NFs are GFlowNets with tree-structured latent state graphs, making every terminating state could only be reached by one trajectory.

[2] $\mathbf{f}(\mathbf{x}), \mathbf{s}(\mathbf{x})$ could also be written as $\mathbf{f}(\mathbf{x}_t, t), \mathbf{s}(\mathbf{x}_t, t)$ in a more strict/general way.

5. Learning a Reward Function from Data

Energy-based model (EBM) can be used as the (negative log) reward function for GFlowNets training. We could use any GFlowNet modeling including those discussed in previous sections, and jointly train it together with an EBM. For instance, in the EB-GFN algorithm (Zhang et al., 2022) a GFlowNet is used to amortize the computational MCMC process of the EBM contrastive divergence training. The two models (EBM and GFlowNet) are trained alternately.

Generative adversarial network (GAN) is closely related to EBM (Che et al., 2020), while its algorithm is more computationally efficient. However, though it may look reasonable at first glance, we cannot directly use the discriminator $D(\mathbf{x})$ as the reward for GFlowNets training. If so, at the end of a perfect training, we would get an optimal discriminator $D^*(\mathbf{x}) = \frac{p_d(\mathbf{x})}{p_d(\mathbf{x})+p_g(\mathbf{x})}$ and optimal GFlowNet generator distribution $p_g(\mathbf{x}) \propto D^*(\mathbf{x})$. This cannot induce $p_g(\mathbf{x}) = p_d(\mathbf{x})$. In fact, if $p_g(\mathbf{x}) = p_d(\mathbf{x})$, we will have $D^*(\mathbf{x}) \equiv 1/2$ and $p_g(\mathbf{x}) = p_d(\mathbf{x}) \equiv$ constant, which is impossible for general data with unbounded support. To fill this gap, we instead design the following algorithm that makes much more sense.

Proposition 9. *An alternate algorithm which trains the discriminator to distinguish between generated data and true data, and trains the GFlowNet with negative energy as*

$$\log \frac{D(\mathbf{x})}{1-D(\mathbf{x})} + \log p_g(\mathbf{x}) \tag{24}$$

would result in a valid generative model.

Nonetheless, we unfortunately do not have access to the value of $p_g(\mathbf{x})$ if the generator is a general GFlowNet, which makes this algorithm an intractable one.

6. Conclusion

We have interpreted existing generative models as GFlowNets with different policies over sample trajectories. This provides some insight into the overlaps between existing generative modeling frameworks, and their connection to general-purpose algorithms for training them. Furthermore, this unification implies a method of constructing an agglomeration of varying types of generative modeling approaches, where GFlowNet acts as a general-purpose glue for tractable inference and training.

References

Anderson, B. D. O. Reverse-time diffusion equation models. *Stochastic Processes and their Applications*, 12:313–326, 1982.

Bengio, E., Jain, M., Korablyov, M., Precup, D., and Bengio, Y. Flow network based generative models for non-iterative diverse candidate generation. In *NeurIPS*, 2021a.

Bengio, Y., Deleu, T., Hu, E. J., Lahlou, S., Tiwari, M., and Bengio, E. Gflownet foundations. *ArXiv*, abs/2111.09266, 2021b.

Burda, Y., Grosse, R. B., and Salakhutdinov, R. Importance weighted autoencoders. *CoRR*, abs/1509.00519, 2016.

Che, T., Zhang, R., Sohl-Dickstein, J., Larochelle, H., Paull, L., Cao, Y., and Bengio, Y. Your gan is secretly an energy-based model and you should use discriminator driven latent sampling. *ArXiv*, abs/2003.06060, 2020.

Child, R. Very deep vaes generalize autoregressive models and can outperform them on images. *ArXiv*, abs/2011.10650, 2021.

Dinh, L., Krueger, D., and Bengio, Y. Nice: Non-linear independent components estimation. *CoRR*, abs/1410.8516, 2015.

Huang, C.-W., Lim, J. H., and Courville, A. C. A variational perspective on diffusion-based generative models and score matching. *ArXiv*, abs/2106.02808, 2021.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. *CoRR*, abs/1312.6114, 2014.

Kingma, D. P., Salimans, T., Poole, B., and Ho, J. Variational diffusion models. *ArXiv*, abs/2107.00630, 2021.

Malkin, N., Jain, M., Bengio, E., Sun, C., and Bengio, Y. Trajectory balance: Improved credit assignment in gflownets. *ArXiv*, abs/2201.13259, 2022.

Øksendal, B. Stochastic differential equations. *The Mathematical Gazette*, 77:65–84, 1985.

Ranganath, R., Tran, D., and Blei, D. M. Hierarchical variational models. *ArXiv*, abs/1511.02386, 2016.

Sønderby, C. K., Raiko, T., Maaløe, L., Sønderby, S. K., and Winther, O. Ladder variational autoencoders. In *NIPS*, 2016.

Song, Y., Garg, S., Shi, J., and Ermon, S. Sliced score matching: A scalable approach to density and score estimation. In *UAI*, 2019.

Song, Y., Durkan, C., Murray, I., and Ermon, S. Maximum likelihood training of score-based diffusion models. 2021.

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

Tzen, B. and Raginsky, M. Neural stochastic differential equations: Deep latent gaussian models in the diffusion limit. *ArXiv*, abs/1905.09883, 2019.

Williams, R. J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256, 2004.

Zhang, D., Malkin, N., Liu, Z., Volokhova, A., Courville, A. C., and Bengio, Y. Generative flow networks for discrete probabilistic modeling. *ArXiv*, abs/2202.01361, 2022.

A. Proofs

A.1. Proposition 2

Proof. We have

$$\begin{aligned}\mathcal{D}_{\mathrm{KL}}\left(P_B(\tau)\|P_F(\tau)\right) &= \mathbb{E}_{p_d(\mathbf{x})q(\mathbf{z}_{1:L}|\mathbf{x})}\left[\log \frac{p_d(\mathbf{x})q(\mathbf{z}_{1:L}|\mathbf{x})}{p(\mathbf{z}_{1:L})p(\mathbf{x}|\mathbf{z}_{1:L})}\right] \\ &= -\mathbb{E}_{p_d(\mathbf{x})}\mathbb{E}_{q(\mathbf{z}_{1:L}|\mathbf{x})}\left[\log \frac{p(\mathbf{x},\mathbf{z}_{1:L})}{q(\mathbf{z}_{1:L}|\mathbf{x})}\right] + \mathbb{E}_{p_d(\mathbf{x})}\left[\log p_d(\mathbf{x})\right] \\ &= -\mathbb{E}_{p_d(\mathbf{x})}\left[\mathrm{ELBO}(p,q)\right] - \mathcal{H}[p_d(\cdot)].\end{aligned}$$

Here $\mathcal{H}[p_d(\cdot)]$ denotes the entropy of the target distribution, which is a constant w.r.t. GFlowNet parameters. □

A.2. Proposition 5

Proof. First notice

$$\partial_t p(\mathbf{x},t) \triangleq \lim_{h\to 0}\frac{1}{h}\left(p(\mathbf{x},t+h)-p(\mathbf{x},t)\right) = \lim_{h\to 0}\frac{1}{h}\left(\int p(\mathbf{x}',t)\mathbb{P}_h(\mathbf{x}|\mathbf{x}')\,\mathrm{d}\mathbf{x}' - p(\mathbf{x},t)\right).$$

Then for any function $w(\mathbf{x})$, we have

$$\begin{aligned}&\int w(\mathbf{x})\partial_t p(\mathbf{x},t)\,\mathrm{d}\mathbf{x} \\ =&\int w(\mathbf{x})\lim_{h\to 0}\frac{1}{h}\left(\int p(\mathbf{x}',t)\mathbb{P}_h(\mathbf{x}|\mathbf{x}')\,\mathrm{d}\mathbf{x}' - p(\mathbf{x},t)\right)\mathrm{d}\mathbf{x} \\ =&\lim_{h\to 0}\frac{1}{h}(\int w(\mathbf{x})\int p(\mathbf{x}',t)\mathbb{P}_h(\mathbf{x}|\mathbf{x}')\,\mathrm{d}\mathbf{x}'\,\mathrm{d}\mathbf{x} - \int w(\mathbf{x}')p(\mathbf{x}',t)\int \mathbb{P}_h(\mathbf{x}|\mathbf{x}')\,\mathrm{d}\mathbf{x}\,\mathrm{d}\mathbf{x}') \\ =&\lim_{h\to 0}\frac{1}{h}\int p(\mathbf{x}',t)\mathbb{P}_h(\mathbf{x}|\mathbf{x}')\left(w(\mathbf{x})-w(\mathbf{x}')\right)\mathrm{d}\mathbf{x}\,\mathrm{d}\mathbf{x}' \\ \triangleq&\int p(\mathbf{x}',t)\sum_{n=1} w^{(n)}(\mathbf{x}')D_n(\mathbf{x}')\mathbf{x}' \\ =&\int w(\mathbf{x}')\sum_{n=1}\left(-\frac{\partial}{\partial \mathbf{x}'}\right)^n\left(p(\mathbf{x}',t)D_n(\mathbf{x}')\right)\mathrm{d}\mathbf{x}',\end{aligned}$$

where $D_n(\mathbf{x}') = \lim_{h\to 0}\frac{1}{hn!}\int \mathbb{P}_h(\mathbf{x}|\mathbf{x}')(\mathbf{x}-\mathbf{x}')^n\,\mathrm{d}\mathbf{x}$ and the last step uses integral by parts. This tells us

$$\partial_t p(\mathbf{x},t) = \sum_{n=1}\left(-\frac{\partial}{\partial \mathbf{x}}\right)^n\left(p(\mathbf{x},t)D_n(\mathbf{x})\right) = -\nabla_{\mathbf{x}}\left(p(\mathbf{x},t)\mathbf{f}(\mathbf{x},t)\right) + \frac{1}{2}\nabla^2_{\mathbf{x}}\left(p(\mathbf{x},t)g^2(\mathbf{x},t)\right),$$

which is essentially the Fokker-Planck equation.

□

A.3. Proposition 6

From the above SDEs, we know that the forward and backward policy is

$$\begin{aligned}\mathbf{x}_{t+h} &= \mathbf{x}_t + \mathbf{f}(\mathbf{x}_t)h + \sqrt{h}g(t)\cdot\boldsymbol{\delta}_F, \\ \mathbf{x}_t &= \mathbf{x}_{t+h} + \left[g^2(t+h)\mathbf{s}(\mathbf{x}_{t+h}) - \mathbf{f}(\mathbf{x}_{t+h})\right]h + \sqrt{h}g(t+h)\boldsymbol{\delta}_B,\end{aligned}$$

where $\boldsymbol{\delta}_F, \boldsymbol{\delta}_B \sim \mathcal{N}(0, \mathbf{I}_D)$. Since we know $h \to 0$, the left and right side of Eq. 19 become

$$\begin{aligned}\log P_F(\mathbf{x}_{t+h}|\mathbf{x}_t) - \log P_B(\mathbf{x}_t|\mathbf{x}_{t+h}) &= -\frac{D}{2}\log(2\pi g_t^2 h) - \frac{1}{2g_t^2 h}\|\Delta\mathbf{x} - h\mathbf{f}_t\|^2 \\ &\quad + \frac{D}{2}\log(2\pi g_{t+h}^2 h) + \frac{1}{2g_{t+h}^2 h}\|\Delta\mathbf{x} - h\mathbf{f}_{t+h} + hg_{t+h}^2\mathbf{s}(\mathbf{x}_{t+h})\|^2, \\ \log p_{t+h}(\mathbf{x}_{t+h}) - \log p_t(\mathbf{x}_t) &= \Delta\mathbf{x}^\top \nabla_\mathbf{x} \log p_t(\mathbf{x}) + O(h),\end{aligned}$$

where $g_t = g(t), \mathbf{f}_t = \mathbf{f}(\mathbf{x}_t), \Delta\mathbf{x} = \mathbf{x}_{t+h} - \mathbf{x}_t = \sqrt{h}g_t\boldsymbol{\epsilon}, \boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I}_D)$. Therefore,

$$\begin{aligned}&\lim_{h\to 0}\frac{1}{\sqrt{h}g_t}(\log p_{t+h}(\mathbf{x}_{t+h})x - \log p_t(\mathbf{x}_t)) = \boldsymbol{\epsilon}^\top \nabla_\mathbf{x}\log p_t(\mathbf{x}), \\ &\lim_{h\to 0}\frac{1}{\sqrt{h}g_t}(\log P_F(\mathbf{x}_{t+h}|\mathbf{x}_t) - \log P_B(\mathbf{x}_t|\mathbf{x}_{t+h})) \\ &\qquad = \lim_{h\to 0}\frac{D}{\sqrt{h}g_t}(\log g_{t+h} - \log g_t) - \frac{1}{2g_t^3 h^{3/2}}\left(hg_t^2\|\boldsymbol{\epsilon}\|^2 + h^2\|\mathbf{f}_t\|^2 - 2h^{3/2}g_t\boldsymbol{\epsilon}^\top\mathbf{f}_t\right) \\ &\qquad\qquad + \frac{1}{2g_t g_{t+h}^2 h^{3/2}}\left(hg_t^2\|\boldsymbol{\epsilon}\|^2 - 2h^{3/2}g_t\boldsymbol{\epsilon}^\top\mathbf{f}_{t+h} + 2g_t g_{t+h}^2 h^{3/2}\boldsymbol{\epsilon}^\top\mathbf{s}(\mathbf{x}_{t+h})\right) \\ &\qquad = \lim_{h\to 0}\boldsymbol{\epsilon}^\top\left(\frac{\mathbf{f}_t}{g_t^2} - \frac{\mathbf{f}_{t+h}}{g_{t+h}^2}\right) + \boldsymbol{\epsilon}^T\mathbf{s}(\mathbf{x}_{t+h}) = \boldsymbol{\epsilon}^T\mathbf{s}(\mathbf{x}_t),\end{aligned}$$

with some smoothness assumptions on $g(t)$ and $\mathbf{f}(\mathbf{x}_t)/g(t)$. This tells us that in order to satisfy the detailed balance criterion, we need to satisfy

$$\boldsymbol{\epsilon}^\top(\mathbf{s}(\mathbf{x}_t, t) - \nabla_\mathbf{x}\log p_t(\mathbf{x})) = 0, \forall\boldsymbol{\epsilon}, \mathbf{x}_t \in \mathbb{R}^D, \forall t \in [0, 1].$$

Since $\boldsymbol{\epsilon}$ could take any value, this is equivalent that the model $\mathbf{s}$ should match with score function $\nabla_\mathbf{x}\log p(\mathbf{x})$. Also, note that this is exactly sliced score matching (Song et al., 2019), which has a more practical formulation

$$\mathbb{E}_{\boldsymbol{\epsilon}}\mathbb{E}_{\mathbf{x}\sim p}\left[\boldsymbol{\epsilon}^\top\nabla_\mathbf{x}\mathbf{s}(\mathbf{x})\boldsymbol{\epsilon} + \frac{1}{2}\left(\boldsymbol{\epsilon}^\top\mathbf{s}(\mathbf{x})\right)^2\right].$$

A.4. Proposition 9

When both models (the GFlowNet and the discriminator) are trained perfectly, we have

$$D^*(\mathbf{x}) = \frac{p_d(\mathbf{x})}{p_d(\mathbf{x}) + p_g(\mathbf{x})},$$

and thus

$$p_g(\mathbf{x}) \propto \exp\left(\log\frac{D^*(\mathbf{x})}{1 - D^*(\mathbf{x})} + \log p_g(\mathbf{x})\right) = \exp\left(\log\frac{p_d(\mathbf{x})}{p_g(\mathbf{x})} + \log p_g(\mathbf{x})\right) = p_d(\mathbf{x}).$$

Hence it is a valid generative model algorithm.