DGFN: Double Generative Flow Networks

Elaine Lau[1,2], Nikhil Vemgal[1,2], Doina Precup[1,2,3], Emmanuel Bengio[4]
[1]Mila, [2]McGill University,
[3]Google DeepMind, [4] Valence Labs
{tsoi.lau, nikhil.vemgal}@mail.mcgill.ca
dprecup@cs.mcgill.ca, bengioe@gmail.com

Abstract

Deep learning is emerging as an effective tool in drug discovery, with potential applications in both predictive and generative models. Generative Flow Networks (GFlowNets/GFNs) are a recently introduced method recognized for the ability to generate diverse candidates, in particular in small molecule generation tasks. In this work, we introduce double GFlowNets (DGFNs). Drawing inspiration from reinforcement learning and Double Deep Q-Learning, we introduce a target network used to sample trajectories, while updating the main network with these sampled trajectories. Empirical results confirm that DGFNs effectively enhance exploration in sparse reward domains and high-dimensional state spaces, both challenging aspects of de-novo design in drug discovery.

1 Introduction

One of the greatest challenges in modern medicine currently lies in the discovery and development of novel therapeutics for disease treatment. This challenge is most evident in the field of infectious diseases, where the creation of new antibiotics has been challenging due to substantial research costs, lengthy timelines, and limited returns. In recent years, a promising alternative approach has emerged in the form of Generative Flow Networks (GFlowNets) [2, 3]. GFlowNets tackle the sampling problem by learning to sample trajectories approximately proportionally to their quality, as captured by a reward function. This approach encourages the discovery of a diverse set of high-reward samples, offering the potential to significantly accelerate the drug discovery and development process.

Despite existing research on credit assignment for GFlowNets, a central challenge continues to be the improvement of exploration and exploitation within the GFlowNets framework [8, 7, 11, 4]. In environments like small molecule generation, where the rewards are sparse [2], GFlowNets may encounter difficulties in breaking away from the current best mode, thus reducing the chances for the agent to discover new modes in the environment [10, 3]. Therefore, it is important to find new ways to enhance exploration efficiency in sparse-reward domains.

In this work, we take inspiration from the double deep Q-learning (DDQN) algorithm from reinforcement learning [14], and introduce double GFlowNets (DGFNs). This approach simply involves employing a target network, which acts as the delayed version of the online network and from which we generate trajectories. Intuitively, this prevents the data distribution on which the online model is trained to become too peaked, too “opinionated", too quickly. We apply DGFNs to two standard GFN tasks: hypergrid (where the complexity and sparsity can be controlled well) and small molecule generation (which is more illustrative of real applications) [2]. Our empirical findings demonstrate that DGFNs finds all modes faster in hypegrid, and uncover a greater number of high-reward modes in the fragment-based molecular design task. These observations provide strong evidence that our proposed strategy indeed encourages diverse exploration, promoting better coverage of the state space and the discovery of diverse candidate solutions.

Accepted to NeurIPS 2023 Workshop on New Frontiers of AI for Drug Discovery and Development (AI4D3 2023).

2 Preliminaries

We begin by introducing GFlowNets, following previous work [2, 8]. Consider a directed acyclic graph $G = (S; A)$, where each vertex $s \in S$ represents a state and $s \to s' \in A$ a state transition. Notably, s_0 is the initial state, with no incoming edges, while s_f is the sink state, with no outgoing edges. A state s_n is considered terminal if $s_n \to s_f \in A$. Each state is assumed to represent some object $s_n = x \in X$. We sample such objects by sampling trajectories starting from s_0, and following "actions" a drawn from A. This yields a sequence $\tau = (s_0 \to s_1 \to \cdots s_n \to s_f)$, referred to as a *complete trajectory*. Let T be the set of all possible trajectories. Assuming a probability distribution over the edges from each node, let $F(\tau)$ denote the flow of τ, representing its unnormalized probability. The *edge flow* is defined as $F(s \to s') = \sum_{\tau \in T : (s \to s') \in \tau} F(\tau)$. The *state flow* is defined as $F(s) = \sum_{\tau \in T : s \in \tau} F(\tau)$. Using these concepts, we can define forward and backward policies, P_F and P_B, as follows::

$$P_F(s \to s') = \frac{F(s \to s')}{F(s)}; \quad P_B(s' \to s) = \frac{F(s \to s')}{F(s')} \tag{1}$$

Given a non-negative reward function $R : X \to \mathbb{R}_0$, let the terminal edge flows be $F(s_n \to s_f) = R(x = s_n)$. The primary objective of GFlowNets is to train a generative policy such that the likelihood of sampling $x \in X$ is proportional to $R(x)$, where

$$p(x) = \sum_{\tau = (s_0 \to \cdots \to s_n = x)} P_F(\tau) = \sum_{\tau} \prod_{t=1}^{n} P_F(s_t \mid s_{t-1}) \tag{2}$$

This is achieved by *balancing* flows such that the total quantity of flow is preserved. [2] In particular, this can be expressed through the *trajectory balance* (TB) [8] condition, where for all trajectories τ:

$$Z \prod_{t=1}^{n} P_F(s_t \mid s_{t-1}) = R(x) \prod_{t=1}^{n} P_B(s_{t-1} \mid s_t) \tag{3}$$

Here, Z represents the total flow, i.e. $Z = \sum_{x \in X} R(x)$. This equality can be turned into an objective and used to learn parameterized P_F, P_B, and Z [8, 7]. Satisfying this constraint across all complete trajectories ensures that $P_F(x) \propto R(x)$. Throughout our experiments, we adopt trajectory balance as the primary training objective.

2.1 Related Work

Double Deep Q-Learning (DDQN) In reinforcement learning, it is well-known that the Q-learning algorithm for learning optimal policies suffers from overestimation bias [15, 5]. This overestimation leads to collapsing the exploration too quickly, slowing down the learning process. Double Deep Q-Networks (DDQN) were introduced to mitigate this issue, by decoupling action selection and value estimation [15]. DDQN uses two separate Q-networks: a target network and an online network. The target network is used to compute the Q-learning target, which is used to update the weights of the online value network. The latter is employed for action selection. The target network is updated periodically by copying the weights of the online network. DDQN has been shown empirically to provide more stable Q-value estimates, thereby enhancing performance.

Improving GFlowNets A number of works have delved into enhancing the training process by manipulating the sampling distribution. For example, Thompson Sampling GFlowNets [11], improve exploration by maintaining uncertainty through ensembling and using it within Thompson sampling. Replay buffers coupled with reward-prioritized replay sampling [12, 16] have been shown to enhance GFlowNet training dynamics. However, existing work only considers using single networks to generate trajectories.

3 Double Generative Flow Networks (DGFN)

In environments characterized by large state spaces or sparse reward signals, the standard approach to GFlowNet training can induce training instability and/or lead to mode collapse [12]. To address this limitation and promote exploration, we draw inspiration from the DDQN idea. We introduce a

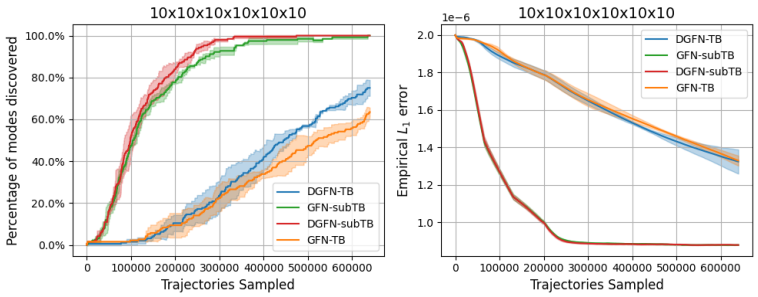

Figure 1: Left: Percentage of modes discovered over the trajectories sampled. Right: L_1 distance between empirical and target distribution over the trajectories sampled.

target network, whose primary role is to sample trajectories, on which the the online network's loss is computed. In our experiments, this approach safeguards against over-optimization of the GFlowNet, particularly in environments characterized by sparse rewards, where the agent may be pushed to exploit known modes if it is able to learn about them quickly, thereby exploring too little. The target network dampens this problem. The target network is periodically copied from the online network, but this process is a bit different from standard DDQN. Because the initial sampled trajectories tend to be relatively random and low reward, in the initial stages of training, we update the target network more frequently. Subsequently, we transition to periodic updates employing a Polyak averaging technique. A full description of the approach is given in Algorithm 1.

Algorithm 1: Double GFlowNets (DGFNs)

Input: Initial phase length T^I, update period T^U
Initialize online flow network F , target flow network F_θ, < 1
for *each training step* $t = 1$ *to* T **do**
  Sample a batch of M trajectories $= fs_0 \; ! \quad ! \; s_n g$ from F_θ
  Compute loss of the online network using sampled trajectories
  if $t < T^I$ or $t \bmod T^U \quad 0$ **then**
    ^{0} + (1) ^{0}
end

4 Experiments

We study the performance of the proposed DGFNs in comparison to conventional GFlowNets with different objective functions on two benchmark tasks: hypergrid and molecule generation.

4.1 Hypergrid Environment: High Dimensional, Sparse Rewards

In the synthetic hypergrid environment introduced by Bengio et al. [2], the goal is to sample trajectories in a D-dimensional grid-world with side length H. The initial state is $(0; :::; 0)$, and actions increment a coordinate by 1 within the bounds of D and H. The agent can terminate at any state.

We use a 6-dimensional grid with side lengths $H = 8; 10; 12$. The reward function is defined as in [2, 8]: $R(x) = R_0 + R_1 \; _i \, \mathbb{I}(0{:}25 < jx_i{=}H \quad 0{:}5j) + R_2 \; _i \, \mathbb{I}(0{:}3 < jx_i{=}H \quad 0{:}5j < 0{:}4)$ with $0 < R_0 \quad R_1 < R_2, R_1 = 1{=}2; R_2 = 2$. We opt for a more challenging environment by setting $R_0 = 10^{-3}$, making exploration less rewarding for the agent. We measure the L_1 error between the true reward distribution and the empirical distribution over the sampled terminal states. Additionally, we track the number of modes discovered over the sampled terminal states.

The results are shown in Fig. 1 and appendix A.1, with means and standard errors computed over 5 independent runs on $DGFN_{TB}$, $DGFN_{SubTB}$, and baseline models: GFN_{TB} and GFN_{SubTB}. "TB" refers to the objective mentioned in equation 3, while "SubTB" is a recent objective proposed by Madan et al. [7] to learn from partial trajectories rather than complete trajectories. As hypothesized, the DGFNs find modes across the hypergrid faster than conventional GFlowNet methods. In particular, the gap becomes wider in more complex environments.

Table 1: Results on the molecule synthesis task. Mean and standard error over 3 runs.

Algorithm	Diverse Top-100		Diverse Top-1000		Top-100 Reward		Top-1000 Reward	
DGFN-TB	**1.035**	**0.002**	**1.017**	**0.002**	**1.036**	**0.002**	**1.020**	**0.002**
GFN-TB	0.972	0.028	0.836	0.097	0.982	0.024	0.931	0.044
GFN-SubTB	1.017	0.001	0.992	0.002	1.017	0.001	0.996	0.002

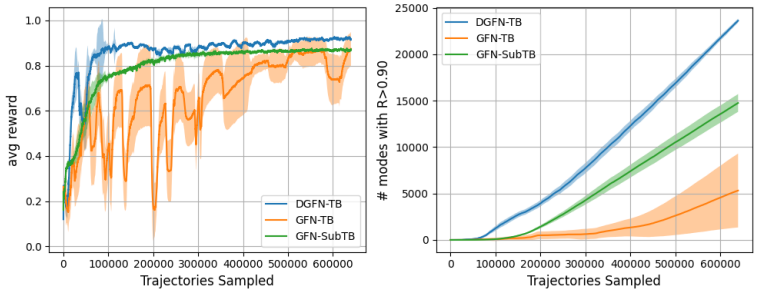

Figure 2: Left: Average reward as a function of trajectories sampled. Right: Number of modes with rewards $R > 0.90$ as a function of trajectories sampled.

4.2 Small molecule synthesis

To assess the capabilities of DGFNs for drug discovery, we conduct experiments involving the generation of small molecules. Specifically, this task involves creating molecules with low binding energy to the soluble epoxide hydrolase (sEH) protein, employing a docking prediction approach originally introduced by Trott and Olson [13]. The agent is tasked with a sequential decision-making process, determining attachment points for molecular building blocks, while adhering to the constraints of chemical validity. This task is particularly challenging, primarily due to the vast state space, estimated at around 10^{16} distinct states, and number of available actions at each state, which can range from 100 to 2000. The reward function R relies on a pre-trained proxy model developed by Bengio et al. [2]. For implementation details, please refer to Appendix A.1.

We train three models: $DGFN_{TB}$ and two baseline models, GFN_{TB} and GFN_{SubTB}. To ensure the reliability of our findings, we report reward means and standard error over 5 independent random seeds. We calculate the number of modes with rewards greater than a 0.9 threshold. Fig. 2 shows that GFN_{TB} exhibits more pronounced fluctuations during the training process, whereas $DGFN_{TB}$ has lower variance. Additionally, $DGFN_{TB}$ surpasses GFN_{SubTB} in the discovery of modes with rewards exceeding 0.9.

We also compute the diverse Top-K (i.e. the set of top molecules such that their pairwise Tanimoto similarity is at most 0.7, c.f. [2]) and Top-K Reward for the generated molecules, showed in Table 1. Here, $DGFN_{TB}$ matches the baselines. This shows that $DGFN_{TB}$ is able to find a greater diversity of solutions without sacrificing reward. Appendix A.2 provides examples of the top 12 molecules generated by our model.

5 Discussion and Conclusion

In this work, we introduced the concept of Double Generative Flow Networks (DGFNs), in order to improve the training stability and the ability of GFlowNets to explore well in large state spaces with sparse rewards. This issue is especially important in drug discovery, where sampling molecules demands a more robust exploration strategy. Our empirical results in hypergrid and molecule synthesis tasks demonstrate the effectiveness of DGFN in promoting diversity in sample generation and enhancing stability. More work remains: more extensive testing of DGFNs across different tasks, the development of a more comprehensive theoretical framework for this approach, and ultimately, the exploration of more techniques inspired by RL and generative modeling to improve the stability of GFlowNets.

Acknowledgements

We genuinely appreciate the funding support from Fonds Recherche Quebec through the FACS-Acquity grant and the National Research Council of Canada. Mila has been instrumental in providing the computational resources for this project. We also want to acknowledge Jarrid Rector Brooks and Moksh Jain for their valuable discussions on related work. This work is partially done at Valence Labs.

References

[1] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A Next-generation Hyperparameter Optimization Framework. *arXiv e-prints*, page arXiv:1907.10902, July 2019.

[2] Emmanuel Bengio, Moksh Jain, Maksym Korablyov, Doina Precup, and Yoshua Bengio. Flow Network based Generative Models for Non-Iterative Diverse Candidate Generation. *arXiv e-prints*, page arXiv:2106.04399, June 2021.

[3] Yoshua Bengio, Salem Lahlou, Tristan Deleu, Edward J. Hu, Mo Tiwari, and Emmanuel Bengio. GFlowNet Foundations. *arXiv e-prints*, page arXiv:2111.09266, November 2021.

[4] Tristan Deleu, António Góis, Chris Emezue, Mansi Rankawat, Simon Lacoste-Julien, Stefan Bauer, and Yoshua Bengio. Bayesian Structure Learning with Generative Flow Networks. *arXiv e-prints*, page arXiv:2202.13903, February 2022.

[5] Hado Hasselt. Double q-learning. In J. Lafferty, C. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems*, volume 23. Curran Associates, Inc., 2010.

[6] Greg Landrum. Rdkit: Open-source cheminformatics. 2006. *Google Scholar*, 2006.

[7] Kanika Madan, Jarrid Rector-Brooks, Maksym Korablyov, Emmanuel Bengio, Moksh Jain, Andrei Nica, Tom Bosc, Yoshua Bengio, and Nikolay Malkin. Learning GFlowNets from partial episodes for improved convergence and stability. *arXiv e-prints*, page arXiv:2209.12782, September 2022.

[8] Nikolay Malkin, Moksh Jain, Emmanuel Bengio, Chen Sun, and Yoshua Bengio. Trajectory balance: Improved credit assignment in GFlowNets. *arXiv e-prints*, page arXiv:2201.13259, January 2022.

[9] Philipp Moritz, Robert Nishihara, Stephanie Wang, Alexey Tumanov, Richard Liaw, Eric Liang, Melih Elibol, Zongheng Yang, William Paul, Michael I. Jordan, and Ion Stoica. Ray: A Distributed Framework for Emerging AI Applications. *arXiv e-prints*, page arXiv:1712.05889, December 2017.

[10] Ling Pan, Dinghuai Zhang, Aaron Courville, Longbo Huang, and Yoshua Bengio. Generative Augmented Flow Networks. *arXiv e-prints*, page arXiv:2210.03308, October 2022.

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

[12] Max W. Shen, Emmanuel Bengio, Ehsan Hajiramezanali, Andreas Loukas, Kyunghyun Cho, and Tommaso Biancalani. Towards Understanding and Improving GFlowNet Training. *arXiv e-prints*, page arXiv:2305.07170, May 2023.

[13] Oleg Trott and Olson Arthur. AutoDock Vina: improving the speed and accuracy of docking with a new scoring function, efficient optimization and multithreading. *Inc. J Comput Chem 2010*, January 2010.

[14] Hado van Hasselt, Arthur Guez, and David Silver. Deep Reinforcement Learning with Double Q-learning. *arXiv e-prints*, page arXiv:1509.06461, September 2015.

[15] Hado van Hasselt, Arthur Guez, and David Silver. Deep Reinforcement Learning with Double Q-learning. *arXiv e-prints*, page arXiv:1509.06461, September 2015.

[16] Nikhil Vemgal, Elaine Lau, and Doina Precup. An Empirical Study of the Effectiveness of Using a Replay Buffer on Mode Discovery in GFlowNets. *arXiv e-prints*, page arXiv:2307.07674, July 2023.

A Appendix

A.1 Experiment details: Hypergrid

The model architecture for both the forward and backward policies remains consistent with the original GFlowNets models [8, 7], using Adam as the optimizer. All models were trained using a batch size of 64 for a total of 640,000 trajectories. Hyperparameter tuning was conducted via Optuna [1], which automates hyperparameter optimization with Ray Tune [9]. For dimension 6, side length 10, the optimal hyperparameters for $DGFN_{TB}$ were found to be an initial phase length of $T^I = 698$ and an update period of $T^U = 137$. For $DGFN_{SubTB}$, the optimal settings were an initial phase length of $T^I = 794$ and an update period of $T^U = 149$.

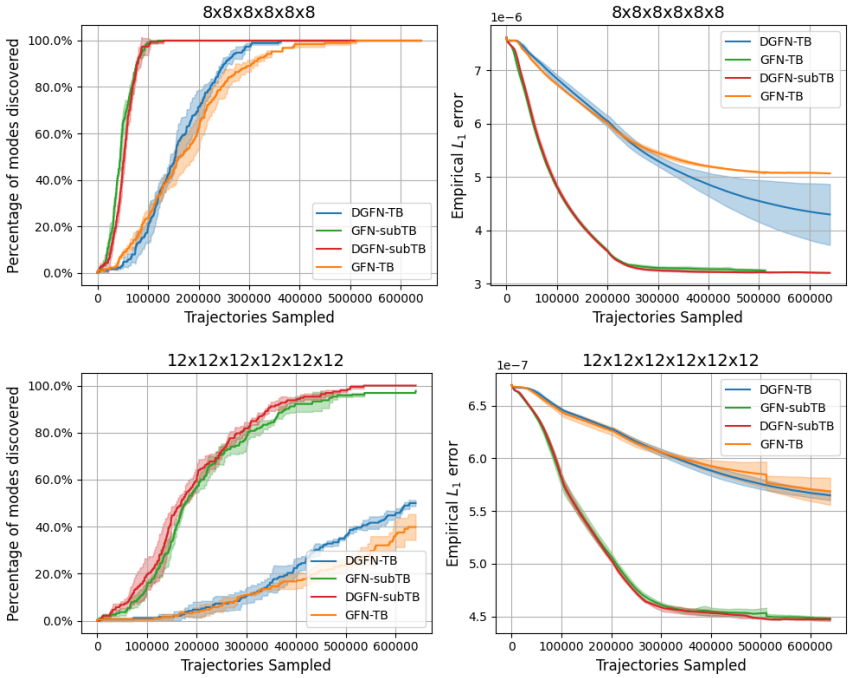

Figure 3: Left: Average reward as a function of trajectories sampled. Right: Number of modes with rewards R > 0.90 as a function of trajectories sampled.

A.2 Experiment details: Small Molecule Synthesis

In our experiments, we used the dataset and proxy model provided by Bengio et al. [2]. The model architecture has the same implementation as the GFlowNet's trajectory balance [8]. Additionally, we incorporated the AutoDock Vina library [13] for binding energy estimation and relied on the RDKit library [6] for chemistry routines.

For the experimental setup, we trained the models for a total of 10,000 iterations using a batch size of 64. The temperature coefficient for the reward function, denoted as [2], was set to 96. To determine the optimal initial phase length T^I and update period T^U, we conducted a grid search over the following values: $T^I \in \{500, 1000, 1500, 2000, 2500, 3000\}$ and $T^U \in \{50, 100, 150, 200, 250, 300, 350\}$. Our experiments revealed that T^I of 2000 and T^U of 200 produced the best results.