Graph Diffusion Policy Optimization

Yijing Liu[*1], **Chao Du**[*†2], **Tianyu Pang**[2], **Chongxuan Li**[3], **Min Lin**[2], **Wei Chen**[†1]
[1]State Key Lab of CAD&CG, Zhejiang University
[2]Sea AI Lab, Singapore
[3]Renmin University of China
{liuyj86,chenvis}@zju.edu.cn;
{duchao,tianyupang,linmin}@sea.com; chongxuanli@ruc.edu.cn

Abstract

Recent research has made significant progress in optimizing diffusion models for downstream objectives, which is an important pursuit in fields such as graph generation for drug design. However, directly applying these models to graph presents challenges, resulting in suboptimal performance. This paper introduces *graph diffusion policy optimization* (GDPO), a novel approach to optimize graph diffusion models for arbitrary (e.g., non-differentiable) objectives using reinforcement learning. GDPO is based on an *eager policy gradient* tailored for graph diffusion models, developed through meticulous analysis and promising improved performance. Experimental results show that GDPO achieves state-of-the-art performance in various graph generation tasks with complex and diverse objectives. Code is available at https://github.com/sail-sg/GDPO.

1 Introduction

Graph generation, a key facet of graph learning, has applications in a variety of domains, including drug and material design [54], code completion [8], social network analysis [20], and neural architecture search [64]. Numerous studies have shown significant progress in graph generation with deep generative models [34, 62, 69, 21]. one of The most notable advances in the field is the introduction of graph diffusion probabilistic models (DPMs) [61, 31]. These methods can learn the underlying distribution from graph data samples and produce high-quality novel graph structures.

In many use cases of graph generation, the primary focus is on achieving specific objectives, such as high drug efficacy [60] or creating novel graphs with special discrete properties [22]. These objectives are often expressed as specific reward signals, such as binding affinity [10] and synthetic accessibility [7], rather than a set of training graph samples. Therefore, a more pertinent goal in such scenarios is to train graph generative models to meet these predefined objectives directly, rather than learning to match a distribution over training data [72].

A major challenge in this context is that most signals are non-differentiable w.r.t. graph representations, making it difficult to apply many optimization algorithms. To address this, methods based on property predictors [29, 37] learn parametric models to predict the reward signals, providing gradient guidance for graph generation. However, since reward signals can be highly complex (e.g., results from physical simulations), these predictors often struggle to provide accurate guidance [44]. An alternative direction is to learn graph generative models as policies through reinforcement learning (RL) [72], which enables the integration of exact reward signals into the optimization. However, existing work primarily explores earlier graph generative models and has yet to leverage the superior performance of graph DPMs [9, 68]. On the other hand, several pioneer works have seen significant

[*]Equal contribution. Work done during Yijing Liu's internship at Sea AI Lab.
[†]Correspondence to Wei Chen and Chao Du.

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

https://doi.org/10.52202/079017-0305

progress in optimizing continuous-variable (e.g., images) DPMs for downstream objectives [6, 16]. The central idea is to formulate the sampling process as a policy, with the objective serving as a reward, and then learn the model using policy gradient methods. However, when these approaches are directly extended to (discrete-variable) graph DPMs, we empirically observe a substantial failure, which we will illustrate and discuss in Sec. 4.

To close this gap, we present *graph diffusion policy optimization* (GDPO), a policy gradient method designed to optimize graph DPMs for arbitrary reward signals. Using an RL formulation similar to that introduced by Black et al. [6] and Fan et al. [16] for continuous-variable DPMs, we first adapt the discrete diffusion process of graph DPMs to a Markov decision process (MDP) and formulate the learning problem as policy optimization. Then, to address the observed empirical failure, we introduce a slight modification to the standard policy gradient method REINFORCE [58], dubbed the *eager policy gradient* and specifically tailored for graph DPMs. Experimental evaluation shows that GDPO proves effective across various scenarios and achieves high sample efficiency. Remarkably, our method achieves a $\mathbf{41.64}\%$ to $\mathbf{81.97}\%$ average reduction in generation-test distance and a 1.03% to $\mathbf{19.31}\%$ improvement in the rate of generating effective drugs, while only querying a small number of samples (1/25 of the training samples).

2 Related Works

Graph Generative Models. Early work in graph generation employs nonparametric random graph models [15, 26]. To learn complex distributions from graph-structured data, recent research has shifted towards leveraging deep generative models. This includes approaches based on auto-regressive generative models [69, 39], variational autoencoders (VAEs) [34, 41, 23], generative adversarial networks (GANs) [62, 9, 43], and normalizing flows [53, 40, 42].

Recently, diffusion probabilistic models (DPMs) [25, 56] have significantly advanced graph generation [70]. Models like EDP-GNN [46] GDSS [31] and DruM [30] construct graph DPMs using continuous diffusion processes [57]. While effective, the use of continuous representations and Gaussian noise can hurt the sparsity of generated graphs. DiGress [61] employs categorical distributions as the Markov transitions in discrete diffusion [2], performing well on complex graph generation tasks. While these works focus on learning graph DPMs from a given dataset, our primary focus in this paper is on learning from arbitrary reward signals.

Controllable Generation for Graphs. Recent progress in controllable generation has also enabled graph generation to achieve specific objectives or properties. Previous work leverages mature conditional generation techniques from GANs and VAEs [66, 52, 36, 28, 14]. This paradigm has been extended with the introduction of guidance-based conditional generation in DPMs [12]. DiGress [61] and GDSS [31] provide solutions that sample desired graphs with guidance from additional property predictors. MOOD [37] improves these methods by incorporating out-of-distribution control. However, as predicting the properties (e.g., drug efficacy) can be extremely difficult [33, 44], the predictors often struggle to provide accurate guidance. Our work directly performs property optimization on graph DPMs, thus bypassing this challenge.

Graph Generation using RL. RL techniques find wide application in graph generation to meet downstream objectives. REINVENT [47] and GCPN [68] are representative works, which define graph environments and optimize policy networks with policy gradient methods [59]. For data-free generation modelling, MolDQN [71] replaces the data-related environment with a human-defined graph environmentand and utilizes Q-Learning [24] for policy optimi zation. To generate more realistic molecules, DGAPN [63] and FREED [67] investigate the fragment-based chemical environment, which reduce the search space significantly. Despite the great successes, existing methods exhibit high time complexity and limited policy model capabilities. Our work, based on graph DPMs with enhanced policy optimization, achieves new state-of-the-art performance.

Aligning DPMs. Several works focus on optimizing generative models to align with human preferences [45, 3]. DPOK [16] and DDPO [6] are representative works that align text-to-image DPMs with black-box reward signals. They formulate the denoising process of DPMs as an MDP and optimize the model using policy gradient methods. For differentiable rewards, such as human preference models [35], AlignProp [50] and DRaFT [11] propose effective approaches to optimize DPMs with direct backpropagation, providing a more accurate gradient estimation than DDPO and DPOK. However,

these works are conducted on images. To the best of our knowledge, our work is the first effective method for aligning graph DPMs, filling a notable gap in the literature.

3 Preliminaries

In this section, we briefly introduce the background of graph DPMs and policy gradient methods.

Following Vignac et al. [61], we consider graphs with categorical node and edge attributes, allowing representation of diverse structured data like molecules. Let $\mathcal{X}$ and $\mathcal{E}$ be the space of categories for nodes and edges, respectively, with cardinalities $a = |\mathcal{X}|$ and $b = |\mathcal{E}|$. For a graph with n nodes, we denote the attribute of node i by a one-hot encoding vector $\boldsymbol{x}^{(i)} \in \mathbb{R}^a$. Similarly, the attribute of the edge[1] from node i to node j is represented as $\boldsymbol{e}^{(ij)} \in \mathbb{R}^b$. By grouping these one-hot vectors, the graph can then be represented as a tuple $\boldsymbol{G} \triangleq (\boldsymbol{X}, \boldsymbol{E})$, where $\boldsymbol{X} \in \mathbb{R}^{n \times a}$ and $\boldsymbol{E} \in \mathbb{R}^{n \times n \times b}$.

3.1 Graph Diffusion Probabilistic Models

Graph diffusion probabilistic models (DPMs) [61] involve a forward diffusion process $q(\boldsymbol{G}_{1:T}|\boldsymbol{G}_0) = \prod_{t=1}^{T} q(\boldsymbol{G}_t|\boldsymbol{G}_{t-1})$, which gradually corrupts a data distribution $q(\boldsymbol{G}_0)$ into a simple noise distribution $q(\boldsymbol{G}_T)$ over a specified number of diffusion steps, denoted as T. The transition distribution $q(\boldsymbol{G}_t|\boldsymbol{G}_{t-1})$ can be factorized into a product of categorical distributions for individual nodes and edges, i.e., $q(\boldsymbol{x}_t^{(i)}|\boldsymbol{x}_{t-1}^{(i)})$ and $q(\boldsymbol{e}_t^{(ij)}|\boldsymbol{e}_{t-1}^{(ij)})$. For simplicity, superscripts are omitted when no ambiguity is caused in the following. The transition distribution for each node is defined as $q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) = \mathrm{Cat}(\boldsymbol{x}_t; \boldsymbol{x}_{t-1}\boldsymbol{Q}_t)$, where the transition matrix is chosen as $\boldsymbol{Q}_t \triangleq \alpha_t \boldsymbol{I} + (1-\alpha_t)(\boldsymbol{1}_a \boldsymbol{1}_a^\top)/a$, with α_t transitioning from 1 to 0 as t increases [2]. It then follows that $q(\boldsymbol{x}_t|\boldsymbol{x}_0) = \mathrm{Cat}(\boldsymbol{x}_t; \boldsymbol{x}_0\bar{\boldsymbol{Q}}_t)$ and $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0) = \mathrm{Cat}(\boldsymbol{x}_{t-1}; \frac{\boldsymbol{x}_t \boldsymbol{Q}_t^\top \odot \boldsymbol{x}_0 \bar{\boldsymbol{Q}}_{t-1}}{\boldsymbol{x}_0 \bar{\boldsymbol{Q}}_t \boldsymbol{x}_t^\top})$, where $\bar{\boldsymbol{Q}}_t \triangleq \boldsymbol{Q}_1 \boldsymbol{Q}_2 \cdots \boldsymbol{Q}_t$ and $\odot$ denotes element-wise product. The design choice of $\boldsymbol{Q}_t$ ensures that $q(\boldsymbol{x}_T|\boldsymbol{x}_0) \approx \mathrm{Cat}(\boldsymbol{x}_T; \boldsymbol{1}_a/a)$, i.e., a uniform distribution over $\mathcal{X}$. The transition distribution for edges is defined similarly, and we omit it for brevity.

Given the forward diffusion process, a parametric reverse denoising process $p_\theta(\boldsymbol{G}_{0:T}) = p(\boldsymbol{G}_T) \prod_{t=1}^{T} p_\theta(\boldsymbol{G}_{t-1}|\boldsymbol{G}_t)$ is then learned to recover the data distribution from $p(\boldsymbol{G}_T) \approx q(\boldsymbol{G}_T)$ (an approximate uniform distribution). The reverse transition $p_\theta(\boldsymbol{G}_{t-1}|\boldsymbol{G}_t)$ is a product of categorical distributions over nodes and edges, denoted as $p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{G}_t)$ and $p_\theta(\boldsymbol{e}_{t-1}|\boldsymbol{G}_t)$. Notably, in line with the $\boldsymbol{x}_0$-parameterization used in continuous DPMs [25, 32], $p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{G}_t)$ is modeled as:

$$p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{G}_t) \triangleq \sum_{\widetilde{\boldsymbol{x}}_0 \in \mathcal{X}} q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \widetilde{\boldsymbol{x}}_0) p_\theta(\widetilde{\boldsymbol{x}}_0|\boldsymbol{G}_t), \tag{1}$$

where $p_\theta(\widetilde{\boldsymbol{x}}_0|\boldsymbol{G}_t)$ is a neural network predicting the posterior probability of $\boldsymbol{x}_0$ given a noisy graph $\boldsymbol{G}_t$. For edges, each definition is analogous and thus omitted.

The model is learned with a graph dataset $\mathcal{D}$ by maximizing the following objective [61]:

$$\mathcal{J}_{\mathrm{GDPM}}(\theta) = \mathbb{E}_{\boldsymbol{G}_0, t} \mathbb{E}_{q(\boldsymbol{G}_t|\boldsymbol{G}_0)} \left[\log p_\theta(\boldsymbol{G}_0|\boldsymbol{G}_t)\right], \tag{2}$$

where $\boldsymbol{G}_0$ and t follow uniform distributions over $\mathcal{D}$ and $[\![1, T]\!]$, respectively. After learning, graph samples can then be generated by first sampling $\boldsymbol{G}_T$ from $p(\boldsymbol{G}_T)$ and subsequently sampling $\boldsymbol{G}_t$ from $p_\theta(\boldsymbol{G}_{t-1}|\boldsymbol{G}_t)$, resulting in a generation trajectory $(\boldsymbol{G}_T, \boldsymbol{G}_{T-1}, \ldots, \boldsymbol{G}_0)$.

3.2 Markov Decision Process and Policy Gradient

Markov decision processes (MDPs) are commonly used to model sequential decision-making problems [17]. An MDP is formally defined by a quintuple $(\mathcal{S}, \mathcal{A}, P, r, \rho_0)$, where $\mathcal{S}$ is the state space containing all possible environment states, $\mathcal{A}$ is the action space comprising all available potential actions, P is the transition function determining the probabilities of state transitions, r is the reward signal, and ρ_0 gives the distribution of the initial state.

In the context of an MDP, an agent engages with the environment across multiple steps. At each step t, the agent observes a state $\boldsymbol{s}_t \in \mathcal{S}$ and selects an action $\boldsymbol{a}_t \in \mathcal{A}$ based on its policy distribution

[1] For convenience, "no edge" is treated as a special type of edge.

https://doi.org/10.52202/079017-0305

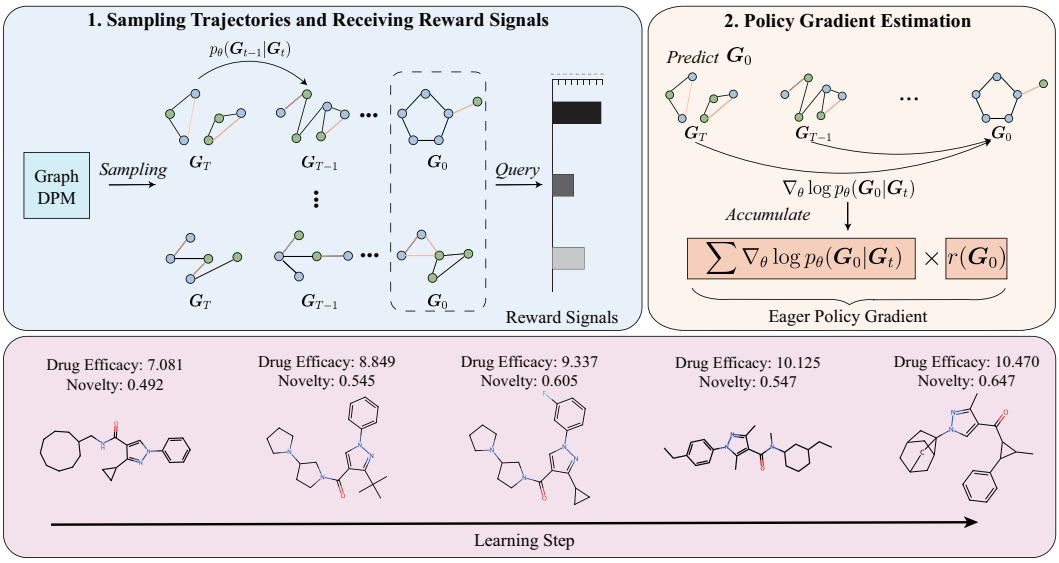


Figure 1: Overview of GDPO. (1) In each optimization step, GDPO samples multiple generation trajectories from the current Graph DPM and queries the reward function with different $\boldsymbol{G}_0$. (2) For each trajectory, GDPO accumulates the gradient $\nabla_\theta \log p_\theta(\boldsymbol{G}_0|\boldsymbol{G}_t)$ of each $(\boldsymbol{G}_0, \boldsymbol{G}_t)$ pair and assigns a weight to the aggregated gradient based on the corresponding reward signal. Finally, GDPO estimates the *eager policy gradient* by averaging the aggregated gradient from all trajectories.

$\pi_\theta(\boldsymbol{a}_t|\boldsymbol{s}_t)$. Subsequently, the agent receives a reward $r(\boldsymbol{s}_t, \boldsymbol{a}_t)$ and transitions to a new state $\boldsymbol{s}_{t+1}$ following the transition function $P(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t, \boldsymbol{a}_t)$. As the agent interacts in the MDP (starting from an initial state $\boldsymbol{s}_0 \sim \rho_0$), it generates a trajectory (i.e., a sequence of states and actions) denoted as $\boldsymbol{\tau} = (\boldsymbol{s}_0, \boldsymbol{a}_0, \boldsymbol{s}_1, \boldsymbol{a}_1, \ldots, \boldsymbol{s}_T, \boldsymbol{a}_T)$. The cumulative reward over a trajectory $\boldsymbol{\tau}$ is given by $R(\boldsymbol{\tau}) = \sum_{t=0}^{T} r(\boldsymbol{s}_t, \boldsymbol{a}_t)$. In most scenarios, the objective is to maximize the following expectation:

$$\mathcal{J}_{\mathrm{RL}}(\theta) = \mathbb{E}_{\boldsymbol{\tau} \sim p(\boldsymbol{\tau}|\pi_\theta)} \left[R(\boldsymbol{\tau}) \right]. \tag{3}$$

Policy gradient methods aim to estimate $\nabla_\theta \mathcal{J}_{\mathrm{RL}}(\theta)$ and thus solve the problem by gradient descent. An important result is the policy gradient theorem [19], which estimates $\nabla_\theta \mathcal{J}_{\mathrm{RL}}(\theta)$ as follows:

$$\nabla_\theta \mathcal{J}_{\mathrm{RL}}(\theta) = \mathbb{E}_{\boldsymbol{\tau} \sim p(\boldsymbol{\tau}|\pi_\theta)} \left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(\boldsymbol{a}_t|\boldsymbol{s}_t) R(\boldsymbol{\tau}) \right]. \tag{4}$$

The REINFORCE algorithm [58] provides a simple method for estimating the above policy gradient using Monte-Carlo simulation, which will be adopted and discussed in the following section.

4 Method

In this section, we study the problem of learning graph DPMs from arbitrary reward signals. We first present an MDP formulation of the problem and conduct an analysis on the failure of a direct application of REINFORCE. Based on the analysis, we introduce a substitute termed *eager policy gradient*, which forms the core of our method *Graph Diffusion Policy Optimization* (GDPO).

4.1 A Markov Decision Process Formulation

A graph DPM defines a sample distribution $p_\theta(\boldsymbol{G}_0)$ through its reverse denoising process $p_\theta(\boldsymbol{G}_{0:T})$. Given a reward signal $r(\cdot)$ for $\boldsymbol{G}_0$, we aim to maximize the expected reward (ER) over $p_\theta(\boldsymbol{G}_0)$:

$$\mathcal{J}_{\mathrm{ER}}(\theta) = \mathbb{E}_{\boldsymbol{G}_0 \sim p_\theta(\boldsymbol{G}_0)} \left[r(\boldsymbol{G}_0) \right]. \tag{5}$$

However, directly optimizing $\mathcal{J}_{\mathrm{ER}}(\theta)$ is challenging since the likelihood $p_\theta(\boldsymbol{G}_0)$ is unavailable [25] and $r(\cdot)$ is black-box, hindering the use of typical RL algorithms [6]. Following Fan et al. [16], we

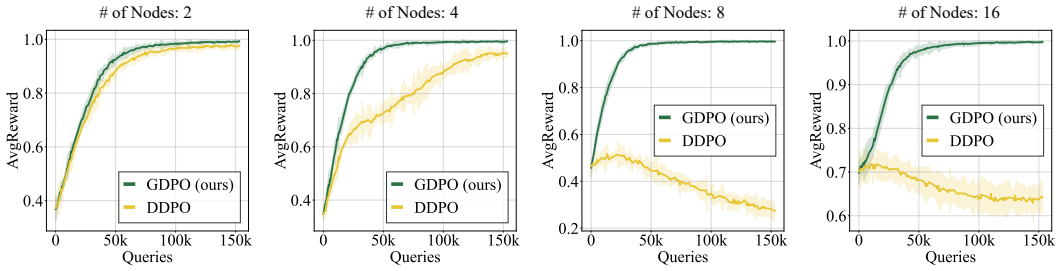


Figure 2: Toy experiment comparing DDPO and GDPO. We generate connected graphs with increasing number of nodes. Node categories are disregarded, and the edge categories are binary, indicating whether two nodes are linked. The graph DPM is initialized randomly as a one-layer graph transformer from DiGress [61]. The diffusion step T is set to 50, and the reward signal $r(\boldsymbol{G}_0)$ is defined as 1 if $\boldsymbol{G}_0$ is connected and 0 otherwise. We use 256 trajectories for gradient estimation in each update. The learning curve illustrates the diminishing performance of DDPO as the number of nodes increases, while GDPO consistently performs well.

formulate the denoising process as a T-step MDP and obtain an equivalent objective. Using notations in Sec. 3, we define the MDP of graph DPMs as follows:

$$\begin{gathered}\boldsymbol{s}_t \triangleq (\boldsymbol{G}_{T-t}, T-t), \quad \boldsymbol{a}_t \triangleq \boldsymbol{G}_{T-t-1}, \quad \pi_\theta(\boldsymbol{a}_t|\boldsymbol{s}_t) \triangleq p_\theta(\boldsymbol{G}_{T-t-1}|\boldsymbol{G}_{T-t}), \\ P(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t, \boldsymbol{a}_t) \triangleq (\delta_{\boldsymbol{G}_{T-t-1}}, \delta_{T-t-1}), \quad r(\boldsymbol{s}_t, \boldsymbol{a}_t) \triangleq r(\boldsymbol{G}_0) \text{ if } t = T, \; r(\boldsymbol{s}_t, \boldsymbol{a}_t) \triangleq 0 \text{ if } t < T,\end{gathered} \tag{6}$$

where the initial state $\boldsymbol{s}_0$ corresponds to the initial noisy graph $\boldsymbol{G}_T$ and the policy corresponds to the reverse transition distribution. As a result, the graph generation trajectory $(\boldsymbol{G}_T, \boldsymbol{G}_{T-1}, \ldots, \boldsymbol{G}_0)$ can be considered as a state-action trajectory $\boldsymbol{\tau}$ produced by an agent acting in the MDP. It then follows that $p(\boldsymbol{\tau}|\pi_\theta) = p_\theta(\boldsymbol{G}_{0:T})$.[2] Moreover, we have $R(\boldsymbol{\tau}) = \sum_{t=0}^{T} r(\boldsymbol{s}_t, \boldsymbol{a}_t) = r(\boldsymbol{G}_0)$. Therefore, the expected cumulative reward of the agent $\mathcal{J}_{\mathrm{RL}}(\theta) = \mathbb{E}_{p(\boldsymbol{\tau}|\pi_\theta)}[R(\boldsymbol{\tau})] = \mathbb{E}_{p_\theta(\boldsymbol{G}_{0:T})}[r(\boldsymbol{G}_0)]$ is equivalent to $\mathcal{J}_{\mathrm{ER}}(\theta)$, and thus $\mathcal{J}_{\mathrm{ER}}(\theta)$ can also be optimized with the policy gradient $\nabla_\theta \mathcal{J}_{\mathrm{RL}}(\theta)$:

$$\nabla_\theta \mathcal{J}_{\mathrm{RL}}(\theta) = \mathbb{E}_{\boldsymbol{\tau}}\left[r(\boldsymbol{G}_0)\sum_{t=1}^{T} \nabla_\theta \log p_\theta(\boldsymbol{G}_{t-1}|\boldsymbol{G}_t)\right], \tag{7}$$

where the generation trajectory $\boldsymbol{\tau}$ follows the parametric reverse process $p_\theta(\boldsymbol{G}_{0:T})$.

4.2 Learning Graph DPMs with Policy Gradient

The policy gradient $\nabla_\theta \mathcal{J}_{\mathrm{RL}}(\theta)$ in Eq. (7) is generally intractable and an efficient estimation is necessary. In a related setting centered on continuous-variable DPMs for image generation, DDPO [6] estimates the policy gradient $\nabla_\theta \mathcal{J}_{\mathrm{RL}}(\theta)$ with REINFORCE and achieves great results. This motivates us to also try REINFORCE on graph DPMs, i.e., to approximate Eq. (7) with a Monte Carlo estimation:

$$\nabla_\theta \mathcal{J}_{\mathrm{RL}} \approx \frac{1}{K}\sum_{k=1}^{K} \frac{T}{|\mathcal{T}_k|} \sum_{t \in \mathcal{T}_k} r(\boldsymbol{G}_0^{(k)}) \nabla_\theta \log p_\theta(\boldsymbol{G}_{t-1}^{(k)}|\boldsymbol{G}_t^{(k)}), \tag{8}$$

where $\{\boldsymbol{G}_{0:T}^{(k)}\}_{k=1}^{K}$ are K trajectories sampled from $p_\theta(\boldsymbol{G}_{0:T})$ and $\{\mathcal{T}_k \subset [\![1, T]\!]\}_{k=1}^{K}$ are uniformly random subsets of timesteps (which avoid summing over all timesteps and accelerate the estimation).

However, we empirically observe that it rarely converges on graph DPMs. To investigate this, we design a toy experiment, where the reward signal is whether $\boldsymbol{G}_0$ is connected. The graph DPMs are randomly initialized and optimized using Eq. (8). We refer to this setting as DDPO. Fig. 2 depicts the learning curves, where the horizontal axis represents the number of queries to the reward signal and the vertical axis represents the average reward. The results demonstrate that DDPO fails to converge to a high reward signal area when generating graphs with more than 4 nodes. Furthermore, as the

[2] With a slight abuse of notation we will use $\boldsymbol{\tau} = \boldsymbol{G}_{0:T}$ and $\boldsymbol{\tau} = (\boldsymbol{s}_0, \boldsymbol{a}_0, \boldsymbol{s}_1, \boldsymbol{a}_1, \ldots, \boldsymbol{s}_T, \boldsymbol{a}_T)$ interchangeably, which should not confuse as the MDP relates them with a bijection.

https://doi.org/10.52202/079017-0305

number of nodes increases, the fluctuation of the learning curves grows significantly. This implies that DDPO is essentially unable to optimize properly on randomly initialized models. We conjecture that the failure is due to the vast space constituted by discrete graph trajectories and the well-known high variance issue of REINFORCE [58]. A straightforward method to reduce variance is to sample more trajectories. However, this is typically expensive in DPMs, as each trajectory requires multiple rounds of model inference. Moreover, evaluating the reward signals of additional trajectories also incurs high computational costs, such as drug simulation [48].

This prompts us to delve deeper at a micro level. Since the policy gradient estimation in Eq. (8) is a weighted summation of gradients, we first inspect each summand term $\nabla_\theta \log p_\theta(\boldsymbol{G}_{t-1}|\boldsymbol{G}_t)$. With the parameterization Eq. (1) described in Sec. 3.1, it has the following form:

$$\nabla_\theta \log p_\theta(\boldsymbol{G}_{t-1}|\boldsymbol{G}_t) = \frac{1}{p_\theta(\boldsymbol{G}_{t-1}|\boldsymbol{G}_t)} \sum_{\widetilde{\boldsymbol{G}}_0} \underbrace{q(\boldsymbol{G}_{t-1}|\boldsymbol{G}_t, \widetilde{\boldsymbol{G}}_0)}_{\text{weight}} \underbrace{\nabla_\theta p_\theta(\widetilde{\boldsymbol{G}}_0|\boldsymbol{G}_t)}_{\text{gradient}}, \tag{9}$$

where we can view the "weight" term as a weight assigned to the gradient $\nabla_\theta p_\theta(\widetilde{\boldsymbol{G}}_0|\boldsymbol{G}_t)$, and thus $\nabla_\theta \log p_\theta(\boldsymbol{G}_{t-1}|\boldsymbol{G}_t)$ as a weighted sum of such gradients, with $\widetilde{\boldsymbol{G}}_0$ taken over all possible graphs. Intuitively, the gradient $\nabla_\theta p_\theta(\widetilde{\boldsymbol{G}}_0|\boldsymbol{G}_t)$ promotes the probability of predicting $\widetilde{\boldsymbol{G}}_0$ from $\boldsymbol{G}_t$. Note, however, that the weight $q(\boldsymbol{G}_{t-1}|\boldsymbol{G}_t, \widetilde{\boldsymbol{G}}_0)$ is completely independent of $r(\widetilde{\boldsymbol{G}}_0)$ and could assign large weight for $\widetilde{\boldsymbol{G}}_0$ that has low reward. Since the weighted sum in Eq. (9) can be dominated by gradient terms with large $q(\boldsymbol{G}_{t-1}|\boldsymbol{G}_t, \widetilde{\boldsymbol{G}}_0)$, given a particular sampled trajectory, it is fairly possible that $\nabla_\theta \log p_\theta(\boldsymbol{G}_{t-1}|\boldsymbol{G}_t)$ increases the probabilities of predicting undesired $\widetilde{\boldsymbol{G}}_0$ with low rewards from $\boldsymbol{G}_t$. This explains why Eq. (8) tends to produce fluctuating and unreliable policy gradient estimates when the number of Monte Carlo samples (i.e., K and $|\mathcal{T}_k|$) is limited. To further analyze why DDPO does not yield satisfactory results, we present additional findings in Appendix A.5. Besides, we discuss the impact of importance sampling techniques in the same section.

4.3 Graph Diffusion Policy Optimization

To address the above issues, we suggest a slight modification to Eq. (8) and obtain a new policy gradient denoted as $\boldsymbol{g}(\theta)$:

$$\boldsymbol{g}(\theta) \triangleq \frac{1}{K} \sum_{k=1}^{K} \frac{T}{|\mathcal{T}_k|} \sum_{t \in \mathcal{T}_k} r(\boldsymbol{G}_0^{(k)}) \nabla_\theta \log p_\theta(\boldsymbol{G}_0^{(k)}|\boldsymbol{G}_t^{(k)}), \tag{10}$$

which we refer to as the *eager policy gradient*. Intuitively, although the number of possible graph trajectories is tremendous, if we partition them into different equivalence classes according to $\boldsymbol{G}_0$, where trajectories with the same $\boldsymbol{G}_0$ are considered equivalent, then the number of these equivalence classes will be much smaller than the number of graph trajectories. The optimization over these equivalence classes will be much easier than optimizing in the entire trajectory space.

Technically, by replacing the summand gradient term $\nabla_\theta \log p_\theta(\boldsymbol{G}_{t-1}|\boldsymbol{G}_t)$ with $\nabla_\theta \log p_\theta(\boldsymbol{G}_0|\boldsymbol{G}_t)$ in Eq. (8), we skip the weighted sum in Eq. (9) and directly promotes the probability of predicting $\boldsymbol{G}_0$ which has higher reward from $\boldsymbol{G}_t$ at all timestep t. As a result, our estimation does not focus on how $\boldsymbol{G}_t$ changes to $\boldsymbol{G}_{t-1}$ within the trajectory; instead, it aims to force the model's generated results to be close to the desired $\boldsymbol{G}_0$, which can be seen as optimizing in equivalence classes. While being a biased estimator of the policy gradient $\nabla_\theta \mathcal{J}_{\mathrm{RL}}(\theta)$, the eager policy gradient consistently leads to more stable learning and better performance than DDPO, as demonstrated in Fig. 2. We present the resulting method in Fig. 1 and Algorithm 1, naming it *Graph Diffusion Policy Optimization* (GDPO).

5 Reward Functions for Graph Generation

In this work, we study both general graph and molecule reward signals that are crucial in real-world tasks. Below, we elaborate on how we formulate diverse reward signals as numerical functions.

5.1 Reward Functions for General Graph Generation

Validity. For graph generation, a common objective is to generate a specific type of graph. For instance, one might be interested in graphs that can be drawn without edges crossing each other [43].

Table 1: General graph generation on SBM and Planar datasets.

Method	Planar Graphs				SBM Graphs			
	Deg ↓	*Clus* ↓	*Orb* ↓	*V.U.N (%)* ↑	*Deg* ↓	*Clus* ↓	*Orb* ↓	*V.U.N (%)* ↑
GraphRNN	24.51 ± 3.22	9.03 ± 0.78	2508.30 ± 30.81	0	6.92 ± 1.13	1.72 ± 0.05	3.15 ± 0.23	4.92 ± 0.35
SPECTRE	2.55 ± 0.34	2.52 ± 0.26	2.42 ± 0.37	25.46 ± 1.33	1.92 ± 1.21	1.64 ± 0.06	1.67 ± 0.14	53.76 ± 3.62
GDSS	10.81 ± 0.86	12.99 ± 0.22	38.71 ± 0.83	0.78 ± 0.72	15.53 ± 1.30	3.50 ± 0.81	15.98 ± 2.30	0
MOOD	5.73 ± 0.82	11.87 ± 0.34	30.62 ± 0.67	1.21 ± 0.83	12.87 ± 1.20	3.06 ± 0.37	2.81 ± 0.35	0
DiGress	1.43 ± 0.90	1.22 ± 0.32	1.72 ± 0.44	70.02 ± 2.17	1.63 ± 1.51	1.50 ± 0.04	1.70 ± 0.16	60.94 ± 4.98
DDPO	109.59 ± 36.69	31.47 ± 4.96	504.19 ± 17.61	2.34 ± 1.10	250.06 ± 7.44	2.93 ± 0.32	6.65 ± 0.45	31.25 ± 5.22
GDPO (ours)	**0.03** ± 0.04	**0.62** ± 0.11	**0.02** ± 0.01	**73.83** ± 2.49	**0.15** ± 0.13	**1.50** ± 0.01	**1.12** ± 0.14	**80.08** ± 2.07

For such objectives, the reward function $r_{\text{val}}(\cdot)$ is then formulated as binary, with $r_{\text{val}}(\boldsymbol{G}_0) \triangleq 1$ indicating that the generated graph $\boldsymbol{G}_0$ conforms to the specified type; otherwise, $r_{\text{val}}(\boldsymbol{G}_0) \triangleq 0$.

Similarity. In certain scenarios, the objective is to generate graphs that resemble a known set of graphs $\mathcal{D}$ at the distribution level, based on a pre-defined distance metric $d(\cdot, \cdot)$ between sets of graphs. As an example, the $\textit{Deg}(\mathcal{G}, \mathcal{D})$ [38] measures the maximum mean discrepancy (MMD) [18] between the degree distributions of a set $\mathcal{G}$ of generated graphs and the given graphs $\mathcal{D}$. Since our method requires a reward for each single generated graph $\boldsymbol{G}_0$, we simply adopt $\textit{Deg}(\{\boldsymbol{G}_0\}, \mathcal{D})$ as the signal. As the magnitude of reward is critical for policy gradients [58], we define $r_{\text{deg}}(\boldsymbol{G}_0) \triangleq \exp\big(-\textit{Deg}(\{\boldsymbol{G}_0\}, \mathcal{D})^2/\sigma^2\big)$, where the σ controls the reward distribution, ensuring that the reward lies within the range of 0 to 1. The other two similar distance metrics are $\textit{Clus}(\mathcal{G}, \mathcal{D})$ and $\textit{Orb}(\mathcal{G}, \mathcal{D})$, which respectively measure the distances between two sets of graphs in terms of the distribution of clustering coefficients [55] and the distribution of substructures [1]. Based on the two metrics, we define two reward signals analogous to r_{deg}, namely, r_{clus} and r_{orb}.

5.2 Reward Functions for Molecular Graph Generation

Novelty. A primary objective of molecular graph generation is to discover novel drugs with desired therapeutic potentials. Due to drug patent restrictions, the novelty of generated molecules has paramount importance. The Tanimoto similarity [4], denoted as $J(\cdot, \cdot)$, measures the chemical similarity between two molecules, defined by the Jaccard index of molecule fingerprint bits. Specifically, $J \in [0, 1]$, and $J(\boldsymbol{G}_0, \boldsymbol{G}_0') = 1$ indicates that two molecules $\boldsymbol{G}_0$ and $\boldsymbol{G}_0'$ have identical fingerprints. Following Xie et al. [65], we define the novelty of a generated graph $\boldsymbol{G}_0$ as $\textit{NOV}(\boldsymbol{G}_0) \triangleq 1 - \max_{\boldsymbol{G}_0' \in \mathcal{D}} J(\boldsymbol{G}_0, \boldsymbol{G}_0')$, i.e., the similarity gap between $\boldsymbol{G}_0$ and its nearest neighbor in the training dataset $\mathcal{D}$, and further define $r_{\text{NOV}}(\boldsymbol{G}_0) \triangleq \textit{NOV}(\boldsymbol{G}_0)$.

Drug-Likeness. Regarding the efficacy of molecular graph generation in drug design, a critical indicator is the binding affinity between the generated drug candidate and a target protein. The docking score [10], denoted as $\textit{DS}(\cdot)$, estimates the binding energy (in kcal/mol) between the ligand and the target protein through physical simulations in 3D space. Following Lee et al. [37], we clip the docking score in the range $[-20, 0]$ and define the reward function as $r_{\text{DS}}(\boldsymbol{G}_0) \triangleq -\textit{DS}(\boldsymbol{G}_0)/20$.

Another metric is the quantitative estimate of drug-likeness $\textit{QED}(\cdot)$, which measures the chemical properties to gauge the likelihood of a molecule being a successful drug [5]. As it takes values in the range $[0, 1]$, we adopt $r_{\text{QED}}(\boldsymbol{G}_0) \triangleq \mathbb{I}[\textit{QED}(\boldsymbol{G}_0) > 0.5]$.

Synthetic Accessibility. The synthetic accessibility [7] $\textit{SA}(\cdot)$ evaluates the inherent difficulty in synthesizing a chemical compound, with values in the range from 1 to 10. We follow Lee et al. [37] and use a normalized version as the reward function: $r_{\text{SA}}(\boldsymbol{G}_0) \triangleq (10 - \textit{SA}(\boldsymbol{G}_0))/9$.

6 Experiments

In this section, we first examine the performance of GDPO on both general graph generation tasks and molecular graph generation tasks. Then, we conduct several ablation studies to investigate the effectiveness of GDPO's design. Our code can be found in the supplementary material.

https://doi.org/10.52202/079017-0305

Table 2: Molecule property optimization results on ZINC250k.

Method	Metric	Target Protein				
		parp1	*fa7*	*5ht1b*	*braf*	*jak2*
GCPN	*Hit Ratio*	0	0	1.455 ± 1.173	0	0
	DS (top 5%)	-8.102± 0.105	-6.688±0.186	-8.544± 0.505	-8.713± 0.155	-8.073±0.093
REINVENT	*Hit Ratio*	0.480 ± 0.344	0.213 ± 0.081	2.453 ± 0.561	0.127 ± 0.088	0.613 ± 0.167
	DS (top 5%)	-8.702 ± 0.523	-7.205 ± 0.264	-8.770 ± 0.316	-8.392 ± 0.400	-8.165 ± 0.277
FREED	*Hit Ratio*	4.627 ± 0.727	1.332 ± 0.113	16.767 ± 0.897	2.940 ± 0.359	5.800 ± 0.295
	DS (top 5%)	-10.579 ± 0.104	-8.378 ± 0.044	-10.714 ± 0.183	-10.561 ± 0.080	-9.735 ± 0.022
MOOD	*Hit Ratio*	7.017 ± 0.428	0.733 ± 0.141	18.673 ± 0.423	5.240 ± 0.285	9.200 ± 0.524
	DS (top 5%)	-10.865 ± 0.113	-8.160 ± 0.071	-11.145 ± 0.042	-11.063 ± 0.034	-10.147 ± 0.060
DiGress	*Hit Ratio*	0.366 ± 0.146	0.182 ± 0.232	4.236 ± 0.887	0.122 ± 0.141	0.861 ± 0.332
	DS (top 5%)	-9.219 ± 0.078	-7.736 ± 0.156	-9.280 ± 0.198	-9.052 ± 0.044	-8.706 ± 0.222
DiGress-guidance	*Hit Ratio*	1.172±0.672	0.321±0.370	2.821± 1.140	0.152±0.303	0.311±0.621
	DS (top 5%)	-9.463± 0.524	-7.318±0.213	-8.971± 0.395	-8.825± 0.459	-8.360±0.217
DDPO	*Hit Ratio*	0.419 ± 0.280	0.342 ± 0.685	5.488 ± 1.989	0.445 ± 0.297	1.717 ± 0.684
	DS (top 5%)	-9.247 ± 0.242	-7.739 ± 0.244	-9.488 ± 0.287	-9.470 ± 0.373	-8.990 ± 0.221
GDPO (ours)	*Hit Ratio*	**9.814** ± 1.352	**3.449** ± 0.188	**34.359** ± 2.734	**9.039** ± 1.473	**13.405** ± 1.515
	DS (top 5%)	**-10.938** ± 0.042	**-8.691** ± 0.074	**-11.304** ± 0.093	**-11.197** ± 0.132	**-10.183** ± 0.124

6.1 General Graph Generation

Datasets and Baselines. Following DiGress [61], we evaluate GDPO on two benchmark datasets: SBM (200 nodes) and Planar (64 nodes), each consisting of 200 graphs. We compare GDPO with GraphRNN [69], SPECTRE [43], GDSS [31], MOOD [37] and DiGress. The first two models are based on RNN and GAN, respectively. The remaining methods are graph DPMs, and MOOD employs an additional property predictor. We also test DDPO [6], i.e., graph DPMs optimized with Eq. (8).

Implementation. We set $T = 1000$, $|\mathcal{T}| = 200$, and $N = 100$. The number of trajectory samples K is 64 for SBM and 256 for Planar. We use a DiGress model with 10 layers. More implementation details can be found in Appendix A.1.

Metrics and Reward Functions. We consider four metrics: *Deg*$(\mathcal{G}, \mathcal{D}_{test})$, *Clus*$(\mathcal{G}, \mathcal{D}_{test})$, *Orb*$(\mathcal{G}, \mathcal{D}_{test})$, and the *V.U.N* metrics. *V.U.N* measures the proportion of generated graphs that are valid, unique, and novel. The reward function is defined as follows:

$$r_{\text{general}} = 0.1 \times (r_{\text{deg}} + r_{\text{clus}} + r_{\text{orb}}) + 0.7 \times r_{\text{val}}, \tag{11}$$

where we do not explicitly incorporate uniqueness and novelty. All rewards are calculated on the training dataset if a reference graph set is required. All evaluation metrics are calculated on the test dataset. More details about baselines, reward signals, and metrics are in Appendix A.3.

Results. Table 1 summarizes GDPO's superior performance in general graph generation, showing notable improvements in *Deg* and *V.U.N* across both SBM and Planar datasets. On the Planar dataset, GDPO significantly reduces distribution distance, achieving an $\mathbf{81.97}\%$ average decrease in metrics of *Deg*, *Clus*, and *Orb* compared to DiGress (the best baseline method). For the SBM dataset, GDPO has a $\mathbf{41.64}\%$ average improvement. The low distributional distances to the test dataset suggests that GDPO accurately captures the data distribution with well-designed rewards. Moreover, we observe that our method outperforms DDPO by a large margin, primarily because the graphs in Planar and SBM contain too many nodes, which aligns with the observation in Fig. 2.

6.2 Molecule Property Optimization

Datasets and Baselines. Molecule property optimization aims to generate molecules with desired properties. We evaluate our method on two large molecule datasets: ZINC250k [27] and MOSES [49]. The ZINC250k dataset comprises 249,456 molecules, each containing 9 types of atoms, with a maximum node count of 38; the MOSES dataset consists of 1,584,663 molecules, with 8 types of atoms and a maximum node count of 30. We compare GDPO with several leading methods:

Table 3: Molecule property optimization results on MOSES.

Method	Metric	Target Protein				
		parp1	*fa7*	*5ht1b*	*braf*	*jak2*
FREED	*Hit Ratio*	0.532 ± 0.614	0	4.255 ± 0.869	0.263 ± 0.532	0.798 ± 0.532
	DS (top 5%)	-9.313 ± 0.357	-7.825 ± 0.167	-9.506 ± 0.236	-9.306 ± 0.327	-8.594 ± 0.240
MOOD	*Hit Ratio*	5.402 ± 0.042	0.365 ± 0.200	26.143 ± 1.647	3.932 ± 1.290	11.301 ± 1.154
	DS (top 5%)	-9.814 ± 1.352	-7.974 ± 0.029	10.734 ± 0.049	-10.722 ± 0.135	-10.158 ± 0.185
DiGress	*Hit Ratio*	0.231 ± 0.463	0.113 ± 0.131	3.852 ± 5.013	0	0.228 ± 0.457
	DS (top 5%)	-9.223 ± 0.083	-6.644 ± 0.533	-8.640 ± 0.907	8.522 ± 1.017	-7.424 ± 0.994
DDPO	*Hit Ratio*	3.037 ± 2.107	0.504 ± 0.667	7.855 ± 1.745	0	3.943 ± 2.204
	DS (top 5%)	-9.727 ± 0.529	-8.025 ± 0.253	-9.631 ± 0.123	-9.407 ± 0.125	-9.404 ± 0.319
GDPO (ours)	*Hit Ratio*	**24.711** ± 1.775	**1.393** ± 0.982	17.646 ± 2.484	**19.968** ± 2.309	**26.688** ± 2.401
	DS (top 5%)	**-11.002** ± 0.056	**-8.468** ± 0.058	**-10.990** ± 0.334	**-11.337** ± 0.137	**-10.290** ± 0.069

GCPN [68], REINVENT [47], FREED [67] and MOOD [37]. GCPN, REINVENT and FREED are RL methods that search in the chemical environment. MOOD, based on graph DPMs, employs a property predictor for guided sampling. Similar to general graph generation, we also compare our method with DiGress and DDPO. Besides, we show the performance of DiGress with property predictors, termed as DiGress-guidance.

Implementation. We set $T = 500$, $|\mathcal{T}| = 100$, $N = 100$, and $K = 256$ for both datasets. We use the same model structure with DiGress. See more details in Appendix A.1.

Metrics and Reward Functions. Following MOOD, we consider two metrics essential for real-world novel drug discovery: **Novel hit ratio (%)** and **Novel top** 5% **docking score**, denoted as *Hit Ratio* and *DS (top* 5%*)*, respectively. Using the notations from Sec. 5.2, the *Hit Ratio* is the proportion of unique generated molecules that satisfy: *DS* $<$ median *DS* of the known effective molecules, *NOV* $>$ 0.6, *QED* > 0.5, and *SA* < 5. The *DS (top* 5%*)* is the average *DS* of the top 5% molecules (ranked by *DS*) that satisfy: *NOV* > 0.6, *QED* > 0.5, and *SA* < 5. Since calculating *DS* requires specifying a target protein, we set five different protein targets to fully test GDPO: *parp1*, *fa7*, *5ht1b*, *braf*, and *jak2*. The reward function for molecule property optimization is defined as follows:

$$r_{\text{molecule}} = 0.1 \times (r_{\text{QED}} + r_{\text{SA}}) + 0.3 \times r_{\text{NOV}} + 0.5 \times r_{\text{DS}}. \tag{12}$$

We do not directly use *Hit Ratio* and *DS (top* 5%*)* as rewards in consideration of method generality. The reward weights are determined through several rounds of search, and we find that assigning a high weight to r_{NOV} leads to training instability, which is discussed in Sec. 6.3. More details about the experiment settings are discussed in Appendix A.4.

Results. In Table 2, GDPO shows significant improvement on ZINC250k, especially in the *Hit Ratio*. A higher *Hit Ratio* means the model is more likely to generate valuable new drugs, and GDPO averagely improves the *Hit Ratio* by 5.72% in comparison with other SOTA methods. For *DS (top* 5%*)*, GDPO also has a 1.48% improvement on average. Discovering new drugs on MOSES is much more challenging than on ZINC250k due to its vast training dataset. In Table 3, GDPO also shows promising results on MOSES. Despite a less favorable *Hit Ratio* on *5ht1b*, GDPO achieves an average improvement of 12.94% on the other four target proteins. For *DS (top* 5%*)*, GDPO records an average improvement of 5.54% compared to MOOD, showing a big improvement in drug efficacy.

6.3 Generalizability, Sample Efficiency, and A Failure Case

To validate whether GDPO correctly optimizes the model, we test the performance of GDPO on metrics not used in the reward signal. In Table 4, we evaluate the performance on Spectral MMD [43], where the GDPO is optimized by Eq. (11). The results demonstrate that GDPO does

Table 4: Generalizability of GDPO on Spectral MMD.

Dataset	Methods		
	DiGress	*DDPO*	*GDPO (ours)*
PLANAR	1.0353± 0.4474	20.1431± 3.5810	**0.8047**± 0.2030
SBM	1.2024± 0.2874	13.2773± 1.4233	**1.0861**± 0.2551

https://doi.org/10.52202/079017-0305

not show overfitting; instead, it finds a more powerful model. The results presented in Appendix A.5 further support that GDPO can attain high sample novelty and diversity.

We then investigate two crucial factors for GDPO: 1) the number of trajectories; 2) the selection of the reward signals. We test our method on ZINC250k and set the target proteins as *5ht1b*. In Fig. 3 (a), the results indicate that GDPO exhibits good sampling efficiency, as it achieves a significant improvement in average reward by querying only 10k molecule reward signals, which is much less than the number of molecules contained in ZINC250k. Moreover, the sample efficiency can be further improved by reducing the number of trajectories, but this may lead to training instability. To achieve consistent results, we use 256 trajectories. In Fig. 3 (b), we illustrate a failure case of GDPO when assigning a high weight to r_{NOV}. Generating novel samples is challenging. MOOD [37] addresses this challenge by controlling noise in the sampling process, whereas we achieve it by novelty optimization. However, assigning a large weight to r_{NOV} can lead the model to rapidly degenerate. One potential solution is to gradually increase the weight and conduct multi-stage optimization.

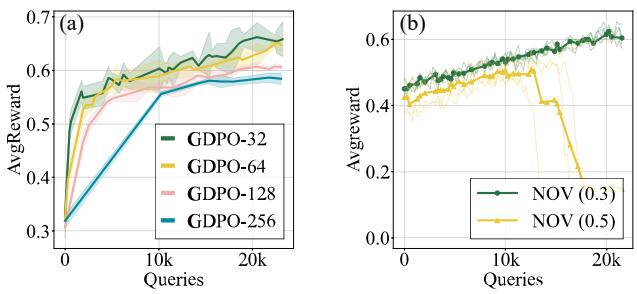


Figure 3: We investigate two key factors of GDPO on ZINC250k, with the target protein being *5ht1b*. Similarly, the vertical axis represents the total queries, while the horizontal axis represents the average reward.(a) We vary the number of trajectories for gradient estimation. (b) We fix the weight of r_{QED} and r_{SA}, and change the weight of r_{NOV} while ensuring the total weight is 1.

7 Conclusion

We introduce GDPO, a novel policy gradient method for learning graph DPMs that effectively addresses the problem of graph generation under given objectives. Evaluation results on both general and molecular graphs indicate that GDPO is compatible with complex multi-objective optimization and achieves state-of-the-art performance on a series of representative graph generation tasks. We discuss some limitations of our work in Appendix A.2. Our future work will investigate the theoretical gap between GDPO and DDPO in order to obtain effective unbiased estimators.

Acknowledgment

This work is supported by the Zhejiang Provincial Natural Science Foundation of China (LD24F020011) and "Pioneer and Leading Goose" R&D Program of Zhejiang (2024C01167). Chongxuan Li was supported by Beijing Natural Science Foundation (L247030); Beijing Nova Program (No. 20230484416).

References

[1] Nesreen K Ahmed, Jennifer Neville, Ryan A Rossi, and Nick Duffield. Efficient graphlet counting for large networks. In *2015 IEEE international conference on data mining*, pages 1–10. IEEE, 2015.

[2] Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. Structured denoising diffusion models in discrete state-spaces. *ArXiv*, abs/2107.03006, 2021.

[3] Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, T. J. Henighan, Nicholas Joseph, Saurav Kadavath, John Kernion, Tom Conerly, Sheer El-Showk, Nelson Elhage, Zac Hatfield-Dodds, Danny Hernandez, Tristan Hume, Scott Johnston, Shauna Kravec, Liane Lovitt, Neel Nanda, Catherine Olsson, Dario Amodei, Tom B. Brown, Jack Clark, Sam McCandlish, Christopher Olah, Benjamin

Mann, and Jared Kaplan. Training a helpful and harmless assistant with reinforcement learning from human feedback. *ArXiv*, abs/2204.05862, 2022.

[4] Dávid Bajusz, Anita Rácz, and Károly Héberger. Why is tanimoto index an appropriate choice for fingerprint-based similarity calculations? *Journal of Cheminformatics*, 7, 2015.

[5] G. Richard J. Bickerton, Gaia V. Paolini, Jérémy Besnard, Sorel Muresan, and Andrew L. Hopkins. Quantifying the chemical beauty of drugs. *Nature chemistry*, 4 2:90–8, 2012.

[6] Kevin Black, Michael Janner, Yilun Du, Ilya Kostrikov, and Sergey Levine. Training diffusion models with reinforcement learning. *ArXiv*, abs/2305.13301, 2023.

[7] Krisztina Boda, Thomas Seidel, and Johann Gasteiger. Structure and reaction based evaluation of synthetic accessibility. *Journal of Computer-Aided Molecular Design*, 21:311–325, 2007.

[8] Marc Brockschmidt, Miltiadis Allamanis, Alexander L. Gaunt, and Oleksandr Polozov. Generative code modeling with graphs. *ArXiv*, abs/1805.08490, 2018.

[9] Nicola De Cao and Thomas Kipf. Molgan: An implicit generative model for small molecular graphs. *ArXiv*, abs/1805.11973, 2018.

[10] Tobiasz Ciepliński, Tomasz Danel, Sabina Podlewska, and Stanislaw Jastrzebski. Generative models should at least be able to design molecules that dock well: A new benchmark. *Journal of Chemical Information and Modeling*, 63:3238 – 3247, 2020.

[11] Kevin Clark, Paul Vicol, Kevin Swersky, and David J. Fleet. Directly fine-tuning diffusion models on differentiable rewards. *ArXiv*, abs/2309.17400, 2023.

[12] Prafulla Dhariwal and Alex Nichol. Diffusion models beat gans on image synthesis. *ArXiv*, abs/2105.05233, 2021.

[13] Jerome Eberhardt, Diogo Santos-Martins, Andreas F Tillack, and Stefano Forli. Autodock vina 1.2. 0: New docking methods, expanded force field, and python bindings. *Journal of chemical information and modeling*, 61(8):3891–3898, 2021.

[14] Peter Eckmann, Kunyang Sun, Bo Zhao, Mudong Feng, Michael K. Gilson, and Rose Yu. Limo: Latent inceptionism for targeted molecule generation. *Proceedings of machine learning research*, 162:5777–5792, 2022.

[15] Paul L. Erdos and Alfréd Rényi. On the evolution of random graphs. *Transactions of the American Mathematical Society*, 286:257–257, 1984.

[16] Ying Fan, Olivia Watkins, Yuqing Du, Hao Liu, Moonkyung Ryu, Craig Boutilier, P. Abbeel, Mohammad Ghavamzadeh, Kangwook Lee, and Kimin Lee. Dpok: Reinforcement learning for fine-tuning text-to-image diffusion models. *ArXiv*, abs/2305.16381, 2023.

[17] Eugene A Feinberg and Adam Shwartz. *Handbook of Markov decision processes: methods and applications*, volume 40. Springer Science & Business Media, 2012.

[18] Arthur Gretton, Karsten M. Borgwardt, Malte J. Rasch, Bernhard Scholkopf, and Alex Smola. A kernel two-sample test. *J. Mach. Learn. Res.*, 13:723–773, 2012.

[19] Ivo Grondman, Lucian Busoniu, Gabriel AD Lopes, and Robert Babuska. A survey of actor-critic reinforcement learning: Standard and natural policy gradients. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(6):1291–1307, 2012.

[20] Aditya Grover, Aaron Zweig, and Stefano Ermon. Graphite: Iterative generative modeling of graphs. In *International Conference on Machine Learning*, 2018.

[21] Xiaojie Guo and Liang Zhao. A systematic survey on deep generative models for graph generation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45:5370–5390, 2020.

[22] Frank Harary and C. St. J. A. Nash-Williams. On eulerian and hamiltonian graphs and line graphs. *Canadian Mathematical Bulletin*, 8:701 – 709, 1965.

 https://doi.org/10.52202/079017-0305

[23] Arman Hasanzadeh, Ehsan Hajiramezanali, Nick G. Duffield, Krishna R. Narayanan, Mingyuan Zhou, and Xiaoning Qian. Semi-implicit graph variational auto-encoders. *ArXiv*, abs/1908.07078, 2019.

[24] Hado Hasselt. Double q-learning. *Advances in neural information processing systems*, 23, 2010.

[25] Jonathan Ho, Ajay Jain, and P. Abbeel. Denoising diffusion probabilistic models. *ArXiv*, abs/2006.11239, 2020.

[26] Paul Holland, Kathryn B. Laskey, and Samuel Leinhardt. Stochastic blockmodels: First steps. *Social Networks*, 5:109–137, 1983.

[27] John J. Irwin, T. Sterling, Michael M. Mysinger, Erin S. Bolstad, and Ryan G. Coleman. Zinc: A free tool to discover chemistry for biology. *Journal of Chemical Information and Modeling*, 52:1757 – 1768, 2012.

[28] Wengong Jin, Regina Barzilay, and T. Jaakkola. Hierarchical generation of molecular graphs using structural motifs. In *International Conference on Machine Learning*, 2020.

[29] Wengong Jin, Regina Barzilay, and T. Jaakkola. Multi-objective molecule generation using interpretable substructures. In *International Conference on Machine Learning*, 2020.

[30] Jaehyeong Jo, Dongki Kim, and Sung Ju Hwang. Graph generation with diffusion mixture. *arXiv preprint arXiv:2302.03596*, 2023.

[31] Jaehyeong Jo, Seul Lee, and Sung Ju Hwang. Score-based generative modeling of graphs via the system of stochastic differential equations. In *International Conference on Machine Learning*, 2022.

[32] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. *ArXiv*, abs/2206.00364, 2022.

[33] Sarah L. Kinnings, Nina Liu, Peter J. Tonge, Richard M. Jackson, Lei Xie, and Philip E. Bourne. A machine learning-based method to improve docking scoring functions and its application to drug repurposing. *Journal of chemical information and modeling*, 51 2:408–19, 2011.

[34] Thomas Kipf and Max Welling. Variational graph auto-encoders. *ArXiv*, abs/1611.07308, 2016.

[35] Kimin Lee, Hao Liu, Moonkyung Ryu, Olivia Watkins, Yuqing Du, Craig Boutilier, P. Abbeel, Mohammad Ghavamzadeh, and Shixiang Shane Gu. Aligning text-to-image models using human feedback. *ArXiv*, abs/2302.12192, 2023.

[36] Myeong-Sung Lee and Kyoungmin Min. Mgcvae: Multi-objective inverse design via molecular graph conditional variational autoencoder. *Journal of chemical information and modeling*, 2022.

[37] Seul Lee, Jaehyeong Jo, and Sung Ju Hwang. Exploring chemical space with score-based out-of-distribution generation. *ArXiv*, abs/2206.07632, 2022.

[38] Renjie Liao, Yujia Li, Yang Song, Shenlong Wang, Will Hamilton, David K Duvenaud, Raquel Urtasun, and Richard Zemel. Efficient graph generation with graph recurrent attention networks. *Advances in neural information processing systems*, 32, 2019.

[39] Renjie Liao, Yujia Li, Yang Song, Shenlong Wang, Charlie Nash, William L. Hamilton, David Kristjanson Duvenaud, Raquel Urtasun, and Richard S. Zemel. Efficient graph generation with graph recurrent attention networks. In *Neural Information Processing Systems*, 2019.

[40] Jenny Liu, Aviral Kumar, Jimmy Ba, Jamie Ryan Kiros, and Kevin Swersky. Graph normalizing flows. *ArXiv*, abs/1905.13177, 2019.

[41] Qi Liu, Miltiadis Allamanis, Marc Brockschmidt, and Alexander L. Gaunt. Constrained graph variational autoencoders for molecule design. In *Neural Information Processing Systems*, 2018.

[42] Youzhi Luo, Keqiang Yan, and Shuiwang Ji. Graphdf: A discrete flow model for molecular graph generation. In *International Conference on Machine Learning*, 2021.

[43] Karolis Martinkus, Andreas Loukas, Nathanael Perraudin, and Roger Wattenhofer. Spectre : Spectral conditioning helps to overcome the expressivity limits of one-shot graph generators. In *International Conference on Machine Learning*, 2022.

[44] Duc Duy Nguyen and Guowei Wei. Agl-score: Algebraic graph learning score for protein-ligand binding scoring, ranking, docking, and screening. *Journal of chemical information and modeling*, 2019.

[45] Khanh Nguyen, Hal Daumé, and Jordan L. Boyd-Graber. Reinforcement learning for bandit neural machine translation with simulated human feedback. *ArXiv*, abs/1707.07402, 2017.

[46] Chenhao Niu, Yang Song, Jiaming Song, Shengjia Zhao, Aditya Grover, and Stefano Ermon. Permutation invariant graph generation via score-based generative modeling. In *International Conference on Artificial Intelligence and Statistics*, 2020.

[47] Marcus Olivecrona, Thomas Blaschke, Ola Engkvist, and Hongming Chen. Molecular de-novo design through deep reinforcement learning. *Journal of Cheminformatics*, 9, 2017.

[48] Nataraj Sekhar Pagadala, Khajamohiddin Syed, and Jack Adam Tuszynski. Software for molecular docking: a review. *Biophysical Reviews*, 9:91 – 102, 2017.

[49] Daniil Polykovskiy, Alexander Zhebrak, Benjamín Sánchez-Lengeling, Sergey Golovanov, Oktai Tatanov, Stanislav Belyaev, Rauf Kurbanov, Aleksey Anatolievich Artamonov, Vladimir Aladinskiy, Mark Veselov, Artur Kadurin, Sergey I. Nikolenko, Alán Aspuru-Guzik, and Alex Zhavoronkov. Molecular sets (moses): A benchmarking platform for molecular generation models. *Frontiers in Pharmacology*, 11, 2018.

[50] Mihir Prabhudesai, Anirudh Goyal, Deepak Pathak, and Katerina Fragkiadaki. Aligning text-to-image diffusion models with reward backpropagation. *ArXiv*, abs/2310.03739, 2023.

[51] Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36, 2024.

[52] Davide Rigoni, Nicoló Navarin, and Alessandro Sperduti. Conditional constrained graph variational autoencoders for molecule design. *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 729–736, 2020.

[53] Chence Shi, Minkai Xu, Zhaocheng Zhu, Weinan Zhang, Ming Zhang, and Jian Tang. Graphaf: a flow-based autoregressive model for molecular graph generation. *ArXiv*, abs/2001.09382, 2020.

[54] Martin Simonovsky and Nikos Komodakis. Graphvae: Towards generation of small graphs using variational autoencoders. In *International Conference on Artificial Neural Networks*, 2018.

[55] Sara Nadiv Soffer and Alexei Vazquez. Network clustering coefficient without degree-correlation biases. *Physical Review E*, 71(5):057101, 2005.

[56] Jascha Narain Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. *ArXiv*, abs/1503.03585, 2015.

[57] Yang Song, Jascha Narain Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *ArXiv*, abs/2011.13456, 2020.

[58] Richard S. Sutton and Andrew G. Barto. Reinforcement learning: An introduction. *IEEE Trans. Neural Networks*, 9:1054–1054, 1998.

[59] Richard S. Sutton, David A. McAllester, Satinder Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Neural Information Processing Systems*, 1999.

 https://doi.org/10.52202/079017-0305

[60] Oleg Trott and Arthur J. Olson. Autodock vina: Improving the speed and accuracy of docking with a new scoring function, efficient optimization, and multithreading. *Journal of Computational Chemistry*, 31, 2009.

[61] Clément Vignac, Igor Krawczuk, Antoine Siraudin, Bohan Wang, Volkan Cevher, and Pascal Frossard. Digress: Discrete denoising diffusion for graph generation. *ArXiv*, abs/2209.14734, 2022.

[62] Hongwei Wang, Jia Wang, Jialin Wang, Miao Zhao, Weinan Zhang, Fuzheng Zhang, Xing Xie, and Minyi Guo. Graphgan: Graph representation learning with generative adversarial nets. *ArXiv*, abs/1711.08267, 2017.

[63] Yulun Wu, Mikaela Cashman, Nicholas Choma, Erica T Prates, Verónica G Melesse Vergara, Manesh Shah, Andrew Chen, Austin Clyde, Thomas S Brettin, Wibe A de Jong, et al. Spatial graph attention and curiosity-driven policy for antiviral drug discovery. *arXiv preprint arXiv:2106.02190*, 2021.

[64] Saining Xie, Alexander Kirillov, Ross B. Girshick, and Kaiming He. Exploring randomly wired neural networks for image recognition. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1284–1293, 2019.

[65] Yutong Xie, Chence Shi, Hao Zhou, Yuwei Yang, Weinan Zhang, Yong Yu, and Lei Li. Mars: Markov molecular sampling for multi-objective drug discovery. *ArXiv*, abs/2103.10432, 2021.

[66] Carl Yang, Peiye Zhuang, Wenhan Shi, Alan Luu, and Pan Li. Conditional structure generation through graph variational generative adversarial nets. In *Neural Information Processing Systems*, 2019.

[67] Soojung Yang, Doyeong Hwang, Seul Lee, Seongok Ryu, and Sung Ju Hwang. Hit and lead discovery with explorative rl and fragment-based molecule generation. *ArXiv*, abs/2110.01219, 2021.

[68] Jiaxuan You, Bowen Liu, Rex Ying, Vijay S. Pande, and Jure Leskovec. Graph convolutional policy network for goal-directed molecular graph generation. In *Neural Information Processing Systems*, 2018.

[69] Jiaxuan You, Rex Ying, Xiang Ren, William L. Hamilton, and Jure Leskovec. Graphrnn: Generating realistic graphs with deep auto-regressive models. In *International Conference on Machine Learning*, 2018.

[70] Mengchun Zhang, Maryam Qamar, Taegoo Kang, Yuna Jung, Chenshuang Zhang, Sung-Ho Bae, and Chaoning Zhang. A survey on graph diffusion models: Generative ai in science for molecule, protein and material. *ArXiv*, abs/2304.01565, 2023.

[71] Zhenpeng Zhou, Steven Kearnes, Li Li, Richard N Zare, and Patrick Riley. Optimization of molecules via deep reinforcement learning. *Scientific reports*, 9(1):10752, 2019.

[72] Zhenpeng Zhou, Steven M. Kearnes, Li Li, Richard N. Zare, and Patrick F. Riley. Optimization of molecules via deep reinforcement learning. *Scientific Reports*, 9, 2018.

A Experimental Details and Additional Results

A.1 Implementation Details.

For all experiments, we use the graph transformer proposed in DiGress [61] as the graph DPMs, and the models are pre-trained on the training dataset before applying GDPO or DDPO. During fine-tuning, we keep all layers fixed except for attention, set the learning rate to 0.00001, and utilize gradient clipping to limit the gradient norm to be less than or equal to 1. In addition, due to significant numerical fluctuations during reward normalization, we follow DDPO [6] in constraining the normalized reward to the range from $[-5, 5]$. This means that gradients resulting from rewards beyond this range will not contribute to model updates. When there is insufficient memory to generate enough trajectories, we use gradient accumulation to increase the number of trajectories used for gradient estimation. We conducted all experiments on a single A100 GPU with 40GB of VRAM and an AMD EPYC 7352 24-core Processor.

Training time and efficiency. Training DiGress on the ZINC250k dataset using a single A100 GPU typically takes 48-72 hours, whereas fine-tuning with GDPO takes only 10 hours (excluding the time for reward function computation). This high efficiency is in line with the findings in the practice of DDPO, which is different from traditional RL methods. Additionally, as in Fig. 3 and Sec 6.3, GDPO effectively improves the average reward of the model using only 10,000 queries. This sample size is notably small compared to the 250,000 samples present in the ZINC250k dataset, showing the impressive sample efficiency of GDPO.

A.2 Limitations and Broader Impact.

Below we list some limitations of the current work:

- Potential for overoptimization: As an RL-based approach, a recognized limitation is the risk of overoptimization, where the DPM distribution may collapse or diverge excessively from the original distribution. In Section 6.3, we demonstrated a failure case where, with a high weight on novelty in the reward function, GDPO encounters a sudden drop in reward after a period of optimization. Future research could explore the application of regularization techniques, similar to those utilized in recent works such as DPO [51], to mitigate this risk.
- Inherited limitations of DPMs: Our method inherits certain limitations inherent to diffusion models, particularly concerning their training and inference costs. As we do not modify the underlying model architecture, these constraints persist.
- Scalability to large graphs: The scalability of GDPO to larger graphs (e.g., with 500 or more nodes) remains unexplored.

For broader impact, this paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

A.3 General Graph Generation

Baselines. There are several baseline methods for general graph generation, we summarize them as follows:

- GraphRNN: a deep autoregressive model designed to model and generate complex distributions over graphs. It addresses challenges like non-uniqueness and high dimensionality by decomposing the generation process into node and edge formations.
- SPECTRE: a novel GAN for graph generation, approaches the problem spectrally by generating dominant parts of the graph Laplacian spectrum and matching them to eigenvalues and eigenvectors. This method allows for modeling global and local graph structures directly, overcoming issues like expressivity and mode collapse.
- GDSS: A novel score-based generative model for graphs is introduced to tackle the task of capturing permutation invariance and intricate node-edge dependencies in graph data generation. This model employs a continuous-time framework incorporating a novel graph diffusion process,

https://doi.org/10.52202/079017-0305

Algorithm 1: Graph Diffusion Policy Optimization

Input: graph DPM p_θ
Input: # of diffusion steps T, # of timestep samples $|\mathcal{T}|$
Input: reward signal $r(\cdot)$, # of trajectory samples K
Input: learning rate η and # of training steps N
Output: Final graph DPM p_θ
for $i = 1, \ldots, N$ **do**
  for $k = 1, \ldots, K$ **do**
    $\boldsymbol{G}_{0:T}^{(k)} \sim p_\theta$ // Sample trajectory
    $\mathcal{T}_k \sim \mathrm{Uniform}([\![1, T]\!])$ // Sample timesteps
    $r_k \leftarrow r(\boldsymbol{G}_0^{(k)})$ // Get rewards
  // Estimate reward mean and variance
  $\bar{r} \leftarrow \frac{1}{K}\sum_{k=1}^{K} r_k \qquad \mathrm{std}[r] \leftarrow \sqrt{\frac{\sum_{k=1}^{K}(r_k - \bar{r})^2}{K-1}}$
  // Estimate the eager policy gradient
  $\boldsymbol{g}(\theta) \leftarrow \frac{1}{K}\sum_{k=1}^{K} \frac{T}{|\mathcal{T}_k|} \sum_{t \in \mathcal{T}_k} (\frac{r_k - \bar{r}}{\mathrm{std}[r]}) \nabla_\theta \log p_\theta(\boldsymbol{G}_0^{(k)} | \boldsymbol{G}_t^{(k)})$
  // Update model parameter
  $\theta \leftarrow \theta + \eta \cdot \boldsymbol{g}(\theta)$

characterized by stochastic differential equations (SDEs), to simultaneously model distributions of nodes and edges.

- DiGress: DiGress is a discrete denoising diffusion model designed for generating graphs with categorical attributes for nodes and edges. It employs a discrete diffusion process to iteratively modify graphs with noise, guided by a graph transformer network. By preserving the distribution of node and edge types and incorporating graph-theoretic features, DiGress achieves state-of-the-art performance on various datasets.
- MOOD: MOOD introduces Molecular Out-Of-distribution Diffusion, which employs out-of-distribution control in the generative process without added costs. By incorporating gradients from a property predictor, MOOD guides the generation process towards molecules with desired properties, enabling the discovery of novel and valuable compounds surpassing existing methods.

Metrics. The metrics of general graph generations are all taken from GraphRNN [38]. The reported metrics compare the discrepancy between the distribution of certain metrics on a test set and the distribution of the same metrics on a generated graph. The metrics measured include degree distributions, clustering coefficients, and orbit counts (which measure the distribution of all substructures of size 4). Following DiGress [61], we do not report raw numbers but ratios computed as follows:

$$r = \mathrm{MMD}(\textit{generated}, \textit{test})^2 / \mathrm{MMD}(\textit{training}, \textit{test})^2 \tag{13}$$

Besides, we explain some metrics that are used in the general graph generation:

- *Clus*: the clustering coefficient measures the tendency of nodes to form clusters in a network. Real-world networks, especially social networks, often exhibit tightly knit groups with more ties between nodes than expected by chance. There are two versions of this measure: global, which assesses overall clustering in the network, and local, which evaluates the clustering around individual nodes.
- *Orb*: Graphlets are induced subgraph isomorphism classes in a graph, where occurrences are isomorphic or non-isomorphic. They differ from network motifs, which are over- or under-represented graphlets compared to a random graph null model. Orb will count the occurrences of each type of graphlet in a graph. Generally, if two graphs have similar numbers of graphlets, they are considered to be relatively similar.

A.4 Molecule Property Optimization

Implementation Details. Following FREED [67], we selected five proteins, PARP-1 (Poly [ADP-ribose] polymerase-1), FA7 (Coagulation factor VII), 5-HT1B (5-hydroxytryptamine receptor 1B),

BRAF (Serine/threonine-protein kinase B-raf), and JAK2 (Tyrosine-protein kinase JAK2), which have the highest AUROC scores when the protein-ligand binding affinities for DUD-E ligands are approximated with AutoDock Vina [13], as the target proteins for which the docking scores are calculated. QED and SA scores are computed using the RDKit library.

Baselines. There are several baseline methods for molecular graph generation under the given objectives, they are diverse in methodology and performance, we summarize them as follows:

- GCPN: Graph Convolutional Policy Network (GCPN) is a general graph convolutional network-based model for goal-directed graph generation using reinforcement learning. The GCPN is trained to optimize domain-specific rewards and adversarial loss through policy gradient, operating within an environment that includes domain-specific rules.
- REINVENT: This method enhances a sequence-based generative model for molecular design by incorporating augmented episodic likelihood, enabling the generation of structures with specified properties. It successfully performs tasks such as generating analogs to a reference molecule and predicting compounds active against a specific biological target.
- HierVAE: a hierarchical graph encoder-decoder for drug discovery, overcoming limitations of previous approaches by using larger and more flexible graph motifs as building blocks. The encoder generates a multi-resolution representation of molecules, while the decoder adds motifs in a coarse-to-fine manner, effectively resolving attachments to the molecule.
- FREED: a novel reinforcement learning (RL) framework for generating effective acceptable molecules with high docking scores, crucial for drug design. FREED addresses challenges in generating realistic molecules and optimizing docking scores through a fragment-based generation method and error-prioritized experience replay (PER).
- MOOD: please refer to Appendix A.3.

Metrics. There are several metrics for evaluating the molecule properties, we summarize the meaning of these metrics as follows:

- Docking Score: Docking simulations aim to find the best binding mode based on scoring functions. Scoring functions in computational chemistry and molecular modeling predict binding affinity between molecules post-docking. They are commonly used for drug-protein interactions, but also for protein-protein or protein-DNA interactions. After defining the score function, we can optimize to find the optimal drug-protein matching positions and obtain the docking score.
- *QED*: Drug-likeness evaluation in drug discovery often lacks nuance, leading to potential issues with compound quality. We introduce QED, a measure based on desirability, which considers the distribution of molecular properties and allows the ranking of compounds by relative merit. QED is intuitive, transparent, and applicable to various settings. We extend its use to assess molecular target druggability and suggest it may reflect aesthetic considerations in medicinal chemistry.
- *SA*: a scoring method for rapid evaluation of synthetic accessibility, considering structural complexity, similarity to available starting materials, and strategic bond assessments. These components are combined using an additive scheme, with weights determined via linear regression analysis based on medicinal chemists' accessibility scores. The calculated synthetic accessibility values align well with chemists' assessments.

A.5 Additional Results of the GDPO

Table 5: General graph generation on SBM and Planar datasets with different reward signals.

Method	Planar Graphs			
	Deg ↓	*Clus* ↓	*Orb* ↓	*V.U.N (%)* ↑
Validity (0.6)	0.03 ± 0.03	0.54 ± 0.08	0.02 ± 0.01	72.34 ± 2.78
Validity (0.7)	0.03 ± 0.04	0.62 ± 0.11	0.02 ± 0.01	73.83 ± 2.49
Validity (0.8)	0.12 ± 0.04	0.88 ± 0.34	0.24 ± 0.07	78.68 ± 3.12
Validity (0.9)	0.86 ± 0.12	2.17 ± 0.84	1.46 ± 0.78	81.26 ± 3.02

Study of the Reward Signals. In Table. 5, we showcase the performance of GDPO on Planar under different configurations of reward weights. We keep the three weights related to distance the same

https://doi.org/10.52202/079017-0305

and adjust the weight of validity while ensuring that the sum of weights is 1. The results indicate that GDPO is not very sensitive to the weights of several reward signals for general graph generation, even though these weight configurations vary significantly, they all achieve good performance. Additionally, we found that GDPO can easily increase *V.U.N* to above 80 while experiencing slight losses in the other three indicators. When applying GDPO in practice, one can make a tradeoff between them based on the specific application requirements.

Table 6: Study of the Important Sampling on ZINC250k.

METHOD	METRIC	TARGET PROTEIN				
		parp1	*fa7*	*5ht1b*	*braf*	*jak2*
DDPO	*Hit Ratio*	0.419 ± 0.280	0.342 ± 0.685	5.488 ± 1.989	0.445 ± 0.297	1.717 ± 0.684
	DS (top 5%)	-9.247 ± 0.242	-7.739 ± 0.244	-9.488 ± 0.287	-9.470 ± 0.373	-8.990 ± 0.221
DDPO-IS	*Hit Ratio*	0.945 ± 0.385	0.319 ± 0.237	10.304 ± 1.277	0.436 ± 0.272	2.697 ± 0.462
	DS (top 5%)	-9.633 ± 0.206	-7.530 ± 0.225	-9.877 ± 0.174	-9.468 ± 0.252	-9.120 ± 0.149
GDPO-IS	*Hit Ratio*	0.850 ± 0.602	0.826 ± 0.827	16.283 ± 1.190	1.339 ± 0.392	4.381 ± 0.501
	DS (top 5%)	-9.482 ± 0.300	-8.254 ± 0.180	-10.361 ± 0.319	-9.771 ± 0.120	-9.583 ± 0.202
GDPO (OURS)	*Hit Ratio*	$\mathbf{9.814} \pm 1.352$	$\mathbf{3.449} \pm 0.188$	$\mathbf{34.359} \pm 2.734$	$\mathbf{9.039} \pm 1.473$	$\mathbf{13.405} \pm 1.151$
	DS (top 5%)	$\mathbf{-10.938} \pm 0.042$	$\mathbf{-8.691} \pm 0.074$	$\mathbf{-11.304} \pm 0.093$	$\mathbf{-11.197} \pm 0.132$	$\mathbf{-10.183} \pm 0.124$

The Impact of Important Sampling. The importance sampling technique in DDPO, aims to facilitate multiple steps of optimization using the same batch of trajectories. This is achieved by weighting each item on the trajectory with an importance weight derived from the density ratio estimated using the model parameters from the previous step θ_{prev} and the current step θ (referred to as DDPO-IS):

$$\nabla_\theta \mathcal{J}_{\text{DDPO-IS}}(\theta) = E_\tau \left[r(G_0) \sum_{t=1}^{T} \frac{p_\theta(G_{t-1}|G_t)}{p_{\theta_{\text{prev}}}(G_{t-1}|G_t)} \nabla_\theta \log p_\theta(G_{t-1}|G_t) \right]. \tag{14}$$

Our eager policy gradient, independently motivated, aims to address the high variance issue of the policy gradient in each step of optimization, as elaborated in Sec. 4.2. Intuitively, the eager policy gradient can be viewed as a biased yet significantly less fluctuating gradient estimation.

We conducted a series of experiments on ZINC250k to compare DDPO, DDPO-IS, and GDPO. The experimental setup remains consistent with the description in Section 6.2. Additionally, considering that the importance sampling technique in DDPO and our eager policy gradient appear to be orthogonal, we also explored combining them simultaneously (referred to as GDPO-IS):

$$\nabla_\theta \mathcal{J}_{\text{GDPO-IS}}(\theta) = E_\tau \left[r(G_0) \sum_{t=1}^{T} \frac{p_\theta(G_0|G_t)}{p_{\theta_{\text{prev}}}(G_0|G_t)} \nabla_\theta \log p_\theta(G_0|G_t) \right]. \tag{15}$$

In Table. 6, while importance sampling enhances the performance of DDPO, consistent with the results reported in the DDPO paper, it does not yield improvements for GDPO-IS over GDPO. We speculate that this discrepancy may be due to the biasness of the eager policy gradient, rendering it incompatible with the importance sampling technique. We intend to investigate the mechanism and address this in our future work. Nevertheless, it is noteworthy that the performance of DDPO-IS remains inferior to GDPO, indicating the superiority of our proposed GDPO method.

Table 7: Novelty and Diversity on ZINC250k.

METRIC	TARGET PROTEIN				
	parp1	*fa7*	*5ht1b*	*braf*	*jak2*
IOU	0.0763%	0.0752%	0.0744%	0.113%	0.0759%
UNIQ	94.86%	97.35%	99.86%	99.74%	97.02%

Novelty and Diversity of GDPO. To provide further insight into the novelty and diversity of our approach, we introduce two additional metrics:

- Intersection over Union (IoU): We compare two sets of molecules: 1) 500 molecules generated by GDPO (denoted as GDPO) and 2) top 500 molecules among 10,000 molecules generated by our base DPM before finetuning (denoted as TopPrior). We then compute IoU=$100 \times \frac{|\text{GDPO} \cap \text{TopPrior}|}{|\text{GDPO} \cup \text{TopPrior}|}\%$. We report an average IoU of 5 independent runs.

- Uniqueness in 10k samples (Uniq): We generate 10,000 molecules and compute the ratio of unique molecules Uniq $= 100 \times \frac{\text{\# unique molecules}}{\text{\# all molecules}}\%$.

In Table. 7, these results show that GDPO has not converged to a trivial solution, wherein it merely selects a subset of molecules generated by the prior diffusion model. Instead, GDPO has learned an effective and distinct denoising strategy from the prior diffusion model.

The Gap between Image DPMs and Graph DPMs. GDPO is tackling the high variance issue inherent in utilizing policy gradients on graph DPMs, as stated and discussed in Sec. 4.2. To provide clarity on what GDPO tackles, we would like to elaborate more on the high variance issue of policy gradients on graph DPMs. Consider the generation trajectories in image and graph DPMs:

In image DPMs, the generation process follows a (discretization of) continuous diffusion process $(\mathbf{x}_t)_{t\in[0,T]}$. The consecutive steps $\mathbf{x}_{t-1}$ and $\mathbf{x}_t$ are typically close due to the Gaussian reverse denoising distribution $p(\mathbf{x}_{t-1}|\mathbf{x}_t)$ (typically with a small variance).

In graph DPMs, the generation process follows a discrete diffusion process $(G_T, \dots, G_0)$, where each G_t is a concrete sample (i.e., one-hot vectors) from categorical distributions. Therefore, consecutive steps G_{t-1} and G_t can be very distant. This makes the trajectory of graph DPMs more fluctuating than images and thus leads to a high variance of the gradient $\nabla_\theta \log p(G_{t-1}|G_t)$ (and the ineffectiveness of DDPO) when evaluated with same number of trajectories as in DDPO.

Regarding the "distance" between two consecutive steps G_t and G_{t-1}, our intuition stems from the fact that graphs generation trajectories are inherently discontinuous. This means that each two consecutive steps can differ significantly, such as in the type/existence of edges. In contrast, the generation trajectories of images, governed by reverse SDEs, are continuous. This continuity implies that for fine-grained discretization (i.e., large T), $\mathbf{x}_t$ and $\mathbf{x}_{t-1}$ can be arbitrarily close to each other (in the limit case of $T \to \infty$).

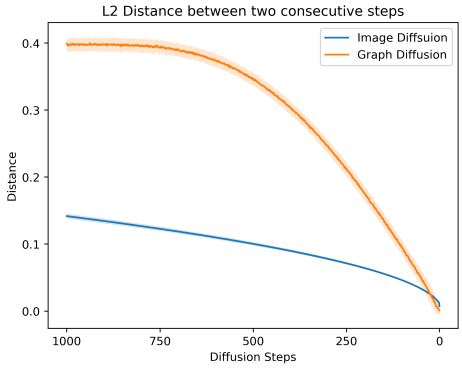


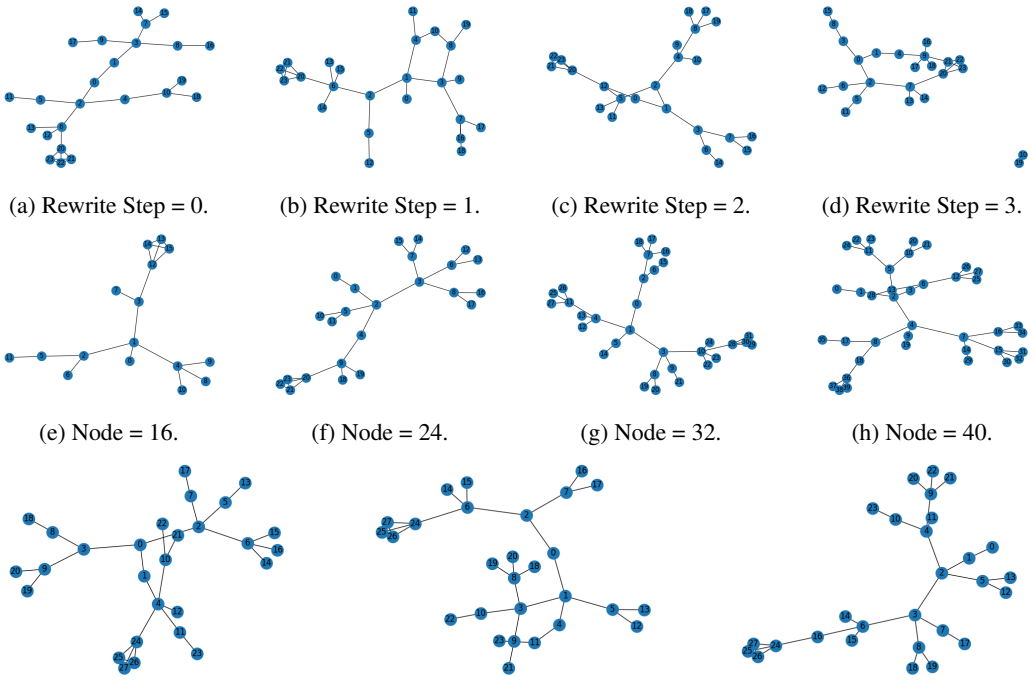
Figure 4: We investigate the L2 distance between two consecutive steps in two types of DPMs. The diffusion step is 1000 for two models.

To provide quantitative support for this discussion, we conduct an analysis comparing the distances between consecutive steps in both image and graph DPMs. We employ a DDPM [a] pre-trained on CIFAR-10 for image diffusion and DiGress [b] pre-trained on the Planar dataset for graph diffusion, both with a total of $T = 1000$ time steps. In these models, graphs are represented with one-hot vectors (as described in Sec. 3) and image pixels are rescaled to the range $[0, 1]$, ensuring their scales are comparable. We then directly compare the per-dimension L2 distances in both spaces, denoted as $\|G_t - G_{t-1}\|_2/\sqrt{D_G}$ and $\|\mathbf{x}_t - \mathbf{x}_{t-1}\|_2/\sqrt{D_I}$, where D_G and D_I are the dimensions of graphs and images, respectively. (Dividing by $\sqrt{D}$ is to eliminate the influence of different dimensionalities.) We sample 512 trajectories from each DPM and plot the mean and deviation of distances with respect to the time step t.

In Fig. 4, the results support the explanation of GDPO. While we acknowledge that graphs and images reside in different spaces and typically have different representations, we believe the comparison with L2 distance can provide valuable insights into the differences between graph and image DPMs.

GDPO on the Synthetic Tree-like Dataset. We first generate a tree and then connect a clique to the nodes of the tree, performing a specified number of rewrite operations as suggested. Based on

https://doi.org/10.52202/079017-0305

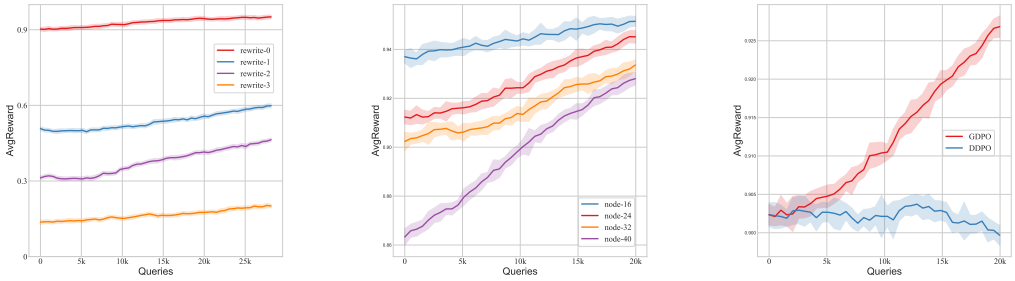


(a) Rewrite Step = 0. (b) Rewrite Step = 1. (c) Rewrite Step = 2. (d) Rewrite Step = 3.

(e) Node = 16. (f) Node = 24. (g) Node = 32. (h) Node = 40.

(i) Shallow Clique Position. (j) Middle Clique Position. (k) Deep Clique Position.

Figure 5: Tree with Different Parameters. Node 0 is the root node.



(a) Performance of GDPO under different rewrite steps.

(b) Performance of GDPO under different graph sizes.

(c) Comparison with DDPO and GDPO.

Figure 6: Ablation Study on the Synthetic Tree-like Dataset.

the number of rewrite steps, graph size, and clique position, we generate multiple datasets, each containing 400 samples. Of these, 256 samples are used for training Graph DPMs, with the remaining samples allocated for validation and testing. In Fig. 5, we present some examples. Fig. 5(a)illustrates a tree structure with a clique of size 4. When the number of rewrite steps is 3, Fig. 5(d) demonstrates that the overall structure of the samples is disrupted. After training the Graph DPMs, we apply GDPO. The model receives a reward of 1 when it generates a tree with a clique; otherwise, the reward is 0. We then ablate the following factors to test the performance of GDPO.

Rewrite Steps: In Fig. 6(a), we demonstrate GDPO's performance across different rewrite steps, with four curves representing steps ranging from 0 to 3. Despite a notable decrease in the initial reward as the number of rewrite steps increases, GDPO consistently optimizes the Graph DPMs effectively to generate the desired graph structure.

Graph Size: In Fig. 6(b), we gradually increase the number of nodes from 16 to 40. The results show that graph size affects the initial reward but does not impact GDPO's optimization performance.

https://doi.org/10.52202/079017-0305

Clique Position: We experiment with inserting the clique at different levels of the tree but find no significant difference. We believe this is because the position of the clique does not affect the initial reward of the Graph DPMs, leading to similar optimization results with GDPO.

Comparison with Baseline: In Fig. 6(c), we compare GDPO with DDPO. The results, consistent with those in Figure 2 of the paper, reveal a clear distinction between GDPO and DDPO in handling challenging data generation tasks.

A.6 Discussions

Comparison with the x_0-prediction Formulation. Indeed, our eager policy gradient in Eq. 10, compared to the policy gradient of REINFORCE in Eq. 8, resembles the idea of training a denoising network to predict the original uncorrupted graph rather than performing one-step denoising. However, we note that training a denoising network to predict the original data is fundamentally a matter of parametrization of one-step denoising. Specifically, the one-step denoising $p_\theta(x_{t-1}|G_t)$ is parameterized as a weighted sum of x_0-prediction, as described in Eq. 1. Our method in Eq. 8 is motivated differently, focusing on addressing the variance issue as detailed in Sections 4.2 and 4.3.

Pros and Cons of the RL Approach against Classifier-based and Classifier-free Guidance for Graph DPMs. Compared to graph diffusion models using classifier-based and classifier-free guidance, RL approaches such as GDPO have at least two main advantages:

- Compatibility with discrete reward signals and discrete graph representations: As guidance for diffusion models is based on gradients, a differentiable surrogate (e.g., property predictors [65, 37]) is needed for non-differentiable reward signals (e.g., results from physical simulations). RL approaches naturally accommodate arbitrary reward functions without the need for intermediate approximations.
- Better sample efficiency: For graph diffusion models with classifier-based or classifier-free guidance, labeled data are required at the beginning and are independently collected with the graph diffusion models. In contrast, RL approaches like GDPO collect labeled data during model training, thus allowing data collection from the current model distribution, which can be more beneficial. We also empirically observe a significant gap in sample efficiency.

Analysis on the Bias-variance Trade off. The main bias of GDPO arises from modifying the "weight" term in Eq. 9, which shifts the model's focus more towards the generated results rather than the intermediate process, thereby reducing potential noise. Due to the discrete nature of Graph DPMs, the x_0-prediction and x_{t-1}-prediction formulations cannot be related through denoising objectives as in continuous DPMs. This issue also complicates the connection between DDPO and GDPO. We have not yet identified a relevant solution and are still working on it. In our empirical study, we do not observe significant performance variance and tradeoff for GDPO given the current scale of experiments. This may be due to the graph sizes we explored not being sufficiently large. In future implementations, we will incorporate support for sparse graphs to assess GDPO's performance on larger graph datasets and investigate the tradeoff more thoroughly.

https://doi.org/10.52202/079017-0305

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: We carefully check the claims, and they align with our evaluation.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: We illustrate the limitations of our work with a failure case. Additionally, we discuss the limitations from various perspectives in the appendix.

   Guidelines:

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

3. **Theory Assumptions and Proofs**

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

   Answer: [NA]

Justification: Our paper does not propose any theoretical results.

Guidelines:

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

4. **Experimental Result Reproducibility**

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

   Answer: [Yes]

   Justification: All the implementation details are discussed in the appendix.

   Guidelines:

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

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

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

https://doi.org/10.52202/079017-0305

Answer: [Yes]

Justification: In the supplementary materials, we provide the code, dataset, and instructions for reproduction.

Guidelines:

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

6. **Experimental Setting/Details**

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

   Answer: [Yes]

   Justification: In the experimental setup section, we provide detailed instructions.

   Guidelines:

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

7. **Experiment Statistical Significance**

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

   Answer: [Yes]

   Justification: All statistical results are obtained by repeating the experiment five times, and the corresponding standard deviations are provided.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).

- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

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

   Answer: [Yes]

   Justification: In the appendix, we provide these contents.

   Guidelines:

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

9. **Code Of Ethics**

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

   Answer: [Yes]

   Justification: This work falls under the general machine learning domain, and during the research process, we adhered to the NeurIPS Code of Ethics.

   Guidelines:

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

10. **Broader Impacts**

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

    Answer: [Yes]

    Justification: In the appendix, we discuss the potential impacts of this work in detail.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

https://doi.org/10.52202/079017-0305

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

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

    Answer: [NA]

    Justification: This work does not involve any relevant datasets or models.

    Guidelines:

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

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: All datasets used in this paper are open-source, and there are no copyright issues involved.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

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

    Answer: [NA]

    Justification: This work is based on existing datasets.

    Guidelines:

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

14. **Crowdsourcing and Research with Human Subjects**

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

    Answer: [NA]

    Justification: This work does not involve any related issues.

    Guidelines:

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

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

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

    Answer: [NA]

    Justification: This work does not involve any related issues.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

https://doi.org/10.52202/079017-0305