Will More Expressive Graph Neural Networks do Better on Generative Tasks?

Xiandong Zou
Imperial College London, UK
xz1320@ic.ac.uk

Xiangyu Zhao
Imperial College London, UK
x.zhao22@imperial.ac.uk

Pietro Liò
University of Cambridge, UK
pietro.lio@cl.cam.ac.uk

Yiren Zhao
Imperial College London, UK
a.zhao@imperial.ac.uk

Abstract

Graph generation poses a significant challenge as it involves predicting a complete graph with multiple nodes and edges based on simply a given label. This task also carries fundamental importance to numerous real-world applications, including de-novo drug and molecular design. In recent years, several successful methods have emerged in the field of graph generation. However, these approaches suffer from two significant shortcomings: (1) the underlying Graph Neural Network (GNN) architectures used in these methods are often underexplored; and (2) these methods are often evaluated on only a limited number of metrics. To fill this gap, we investigate the expressiveness of GNNs under the context of the molecular graph generation task, by replacing the underlying GNNs of graph generative models with more expressive GNNs. Specifically, we analyse the performance of six GNNs in two different generative frameworks (GCPN and GraphAF), on six different molecular generative objectives on the ZINC-250k dataset. Through our extensive experiments, we demonstrate that advanced GNNs can indeed improve the performance of GCPN and GraphAF on molecular generation tasks, but GNN expressiveness is not a necessary condition for a good GNN-based generative model. Moreover, we show that GCPN and GraphAF with advanced GNNs can achieve state-of-the-art results across 17 other non-GNN-based graph generative approaches, such as variational autoencoders and Bayesian optimisation models, on the proposed molecular generative objectives (DRD2, Median1, Median2), which are important metrics for de-novo molecular design.

1 Introduction

Graph generation has always been viewed as a challenging task as it involves the prediction of a complete graph comprising multiple nodes and edges based on a given label. This task, however, holds paramount importance in a wide array of real-world applications, such as de-novo molecule design [1]. In de-novo molecule generation, the chemical space is discrete by nature, and the entire search space is huge, which is estimated to be between 10^{23} and 10^{60} [2]. The generation of novel and valid molecular graphs with desired physical, chemical and biological property objectives should also be considered, these together with the large search space makes it a difficult task, since *these property objectives are highly complex and non-differentiable* [3].

Recently, there has been significant progress in molecular graph generation with graph neural network (GNN)-based deep generative models, such as Graph Convolutional Policy Network (GCPN) [3] and Flow-based Autoregressive Model (GraphAF) [4], and they both use the Relational Graph Convolutional Network (R-GCN) [5] as their inner graph representation model. Meanwhile, several researchers have focused on improving GNN expressiveness by introducing architectural changes [6–8], leading to the discovery of diverse forms of GNNs that excel in graph classification and regression tasks. Naturally, it is worth considering *will these more expressive GNNs help in molecular graph generation and will they perform better than the de-facto network used in GCPN and GraphAF?*

In addition, nowadays, there are many metrics used in de-novo molecule design (*eg. molecule's bioactivity against its corresponding disease targets: DRD2 and JNK3*) [9]. However, GCPN and GraphAF only consider two molecular generative objectives related to drug design: the Penalized logP and the QED. This brings two major drawbacks. First, there is a lack of consideration of a broader set of generation metrics beyond those related to drug design. Secondly, both Penalized logP and the QED are incapable of effectively distinguishing between various generation models, rendering them disabled for such purpose [9].

In this work, we replace R-GCN in GCPN and GraphAF with more expressive GNNs. Then, we evaluate our proposed models: GCPN variants and GraphAF variants on a wide array of molecular generative objectives (e.g. DRD2, Median1, Median2), which are important metrics for de-novo molecular design, to derive more statistically significant results. We make the following contributions:

- Although GNN expressiveness defined by the Weisfeiler-Lehman (1-WL) graph isomorphism test works well on graph classification and regression tasks, it is not a necessary condition for a good GNN-based generative model. We observe empirically that the expressiveness of GNNs does not correlate well with their performance on GNN-based generative models, and GNNs incorporating edge feature extraction can improve GNN-based generative models.
- Although Penalised logP and QED are widely used in evaluating goal-directed graph generative models, they are not effective metrics to differentiate generative models. Other metrics, such as DRD2, Median1 and Median2, can better evaluate the ability of a graph generative model.
- Our findings reveal that while there is no direct correlation between expressiveness and performance in graph generation, substituting the inner GNNs of GCPN and GraphAF with advanced GNNs like GearNet yields better performance (e.g. 102.51% better in DRD2 and 48.96% better in Median2). By doing so, these models surpass or reach comparable performance to state-of-the-art non-graph-based generative methods for de-novo molecule generation.

2 Related Work

A variety of deep generative models have been proposed for molecular graph generation recently [3, 4, 10–17]. In this paper, we confine our scope to single-objective approaches for molecular generation. Specifically, our focus centres on the generation of organic molecules that possess a desired scalar metric, encompassing key physical, chemical, and biological properties.

2.1 GNN-based Graph Generative Models

Recently, there has been significant progress in molecular graph generation with GNN-based deep generative models, such as GCPN [3] and GraphAF [4]. Both of them use R-GCN [5], the state-of-the-art GNN at that time, as their inner graph representation model. Nevertheless, in recent years, the landscape has witnessed the emergence of increasingly expressive GNNs such as GATv2 [6], GSN [7] and GearNet [8]. These advanced GNN models have showcased superior performance in various tasks, including graph classification and regression, surpassing the capabilities of R-GCN. Furthermore, the methods GCPN and GraphAF only evaluate their performance in goal-directed molecule generation tasks using two commonly employed metrics in drug design: quantitative estimate of drug-likeness score (QED) and penalised octanol-water partition coefficient (Penalized logP). However, it is worth noting that many advanced graph generative approaches, as discussed in Section 2.2, can achieve state-of-the-art results on benchmarks for QED and Penalized logP [9, 15]. *QED is likely to have a global maximum of 0.948 and even random sampling could reach that value. Penalized logP is unbounded and the relationship between Penalized logP values and molecular structures is fairly simple: adding carbons monotonically increases the estimated Penalized logP value [3, 9, 18].* This is presented in Figure 3 in Appendix E. Consequently, one could argue that these scores have reached a saturation point, making them less meaningful as evaluation metrics. Only using these two metrics relevant to drug design on goal-directed graph generation tasks to assess the graph generative models is not convincing enough and cannot provide insights for distinguishing different algorithms' de-novo molecule generation ability [9].

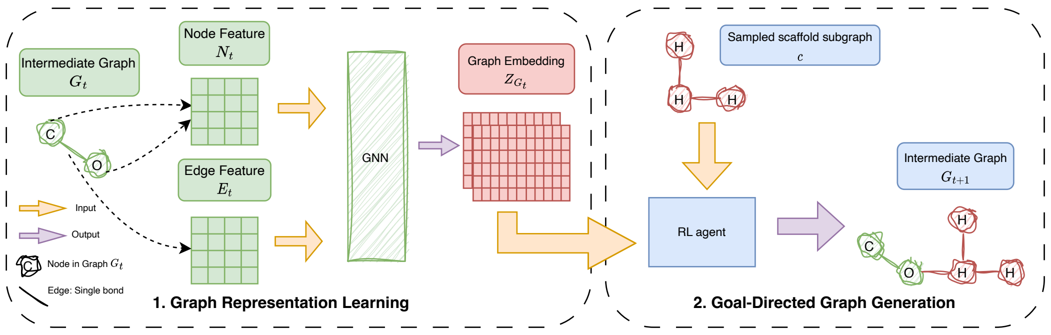

Intermediate Graph G_t
Node Feature N_t
Edge Feature E_t
GNN
Graph Embedding Z_{G_t}
Input
Output
Node in Graph G_t
Edge: Single bond
1. Graph Representation Learning
Sampled scaffold subgraph c
RL agent
Intermediate Graph G_{t+1}
2. Goal-Directed Graph Generation
C
O
H

Figure 1: An overview of the GCPN model: this is an example of iterative graph generation from an intermediate graph G_t to an intermediate graph G_{t+1}. Part 1 is the illustration of the graph representation learning process based on a GNN. Part 2 is the illustration of the graph generative procedure based on a reinforcement learning (RL) agent. New nodes or edges are marked in red.

2.2 Non-GNN-based Graph Generative Models

There are other non-GNN-based graph generative models. Genetic algorithms (GA) are generation approaches by relying on biologically inspired operators such as mutation, crossover and selection. Bayesian Optimization (BO) [19] is an approach that uses a sequential optimization technique that leverages probabilistic models to search for the optimal solution. Variational Autoencoders (VAEs) [20] is a type of generative model in machine learning that combines elements of both autoencoders and probabilistic latent variable models to learn and generate data by mapping it to a latent space with continuous distributions. Monte-Carlo Tree Search (MCTS) constructs a search tree by iteratively selecting actions, simulating possible outcomes, and propagating the results to inform future decisions, ultimately aiming to find the optimal solution. Hill Climbing (HC) is an iterative optimization algorithm that starts with an arbitrary solution to a problem, and then attempts to find a better solution by making an incremental change to the solution. Reinforcement Learning (RL) learns how intelligent agents take actions in an environment to maximize the cumulative reward by transitioning through different states. Details about non-GNN-based graph generative models we used in Section 4 can be found in Appendix A.

3 Method

In this section, we provide the theoretical background for graph generative models. A GNN-based graph generative model consists of a GNN model and a graph generative framework. The GNN learns the hidden representations of a graph, such as the node feature and the graph feature. The goal of the graph generation framework is to generate realistic molecular graph structures based on a given generative objective. The detail of GCPN is presented in Figure 1, and an illustration of GraphAF is displayed in Figure 2 in Appendix B.

3.1 Preliminaries

Let $G = (V, \mathcal{E})$ define a graph, where V denotes the set of nodes, and $\mathcal{E} \subseteq V \times V$ denotes the set of edges. A relational graph can be expressed as $G = (V, \mathcal{E}, \mathcal{R})$ where $\mathcal{R}$ denotes the set of edge relations or edge types. For example, (i, j, r) means the edge from node i to node j with edge type r. A molecular graph can be represented by a tuple of features $(\mathbf{A}, \mathbf{H}, \mathbf{E}, \mathbf{R})$, where

- $\mathbf{A} \in \mathbb{R}^{|V|\times|V|}$ is the adjacency matrix, with each entry a_{ij} representing an edge (if any) between nodes i and j; note that this is different from the conventional $0, 1^{|V|\times|V|}$ adjacency matrix format, since there are different types of bonds (*i.e.*, single, double, triple, aromatic).
- $\mathbf{H} \in \mathbb{R}^{|V|\times d}$ is the feature matrix, $\mathbf{h}_i \in \mathbb{R}^d$ is the d-dimensional features of node i.
- $\mathbf{E} \in \mathbb{R}^{|\mathcal{E}|\times d_e}$ is the edge feature matrix, $\mathbf{e}_{ij} \in \mathbb{R}^{d_e}$ is the d_e-dimensional features of edge (i, j).
- $\mathbf{R} \in \mathbb{N}^{|\mathcal{E}|}$ is a vector containing the edge types of each edge $(i, j) \in \mathcal{E}$ and $r_{(i,j)} \in \mathcal{R}$.

The degree matrix $\mathbf{D} \in \mathbb{R}^{|V| \times |V|}$ of a graph $G = (V, \mathcal{E})$ is a diagonal matrix with each diagonal entry $d_{ii} = \text{deg}(v_i)$, where the $\text{deg}(v_i)$ of a vertex counts the number of times an edge terminates at that vertex in an undirected graph.

3.1.1 Subgraph & Isomorphism

A graph $G' = (V', \mathcal{E}')$ is a *subgraph* of a graph $G = (V, \mathcal{E})$ (denoted $G' \subseteq G$) if and only if $V' \subseteq V$ and $\mathcal{E}' \subseteq \mathcal{E}$. Two graphs $G = (V, \mathcal{E})$ and $G' = (V', \mathcal{E}')$ are *isomorphic* (denoted $G \cong G'$) if and only if there exists an adjacency-preserving bijective mapping $f : V \rightarrow V'$, i.e.,

$$\forall i, j \in V, (i, j) \in \mathcal{E} \iff (f(i), f(j)) \in \mathcal{E}' \tag{1}$$

An *automorphism* of a graph $G = (V, \mathcal{E})$ is an isomorphism that maps G onto itself.

3.2 Graph Neural Networks

All the Graph Neural Networks (GNNs) investigated in this paper can be abstracted as Message Passing Neural Networks (MPNNs). A general MPNN operation iteratively updates the node features $\mathbf{h}_i^{(l)} \in \mathbb{R}^d$ from layer l to layer $l + 1$ via propagating messages through neighbouring nodes $j \in \mathcal{N}_i$, which can be formalised by the following equation:

$$\mathbf{h}_i^{(l+1)} = \text{UPDATE}\left(\mathbf{h}_i^{(l)}, \bigoplus_{j \in \mathcal{N}_i} \text{MESSAGE}\left(\mathbf{h}_i^{(l)}, \mathbf{h}_j^{(l)}, \mathbf{e}_{ij}\right)\right) \tag{2}$$

where MESSAGE and UPDATE are learnable functions, such as Multi-Layer Perceptrons (MLPs), $\mathcal{N}_i = \{j | (i, j) \in \mathcal{E}\}$ is the (1-hop) neighbourhood of node i, and $\bigoplus$ is a permutation-invariant local neighbourhood aggregation function, such as sum, mean or max. After k iterations of aggregation, node i's representation $\mathbf{h}_i^{(k)}$ can capture the structural information within its k-hop graph neighbourhood. Then, the graph embedding $\mathbf{h}_G \in \mathbb{R}^d$ can be obtained via a READOUT function:

$$\mathbf{h}_G = \text{READOUT}_{i \in V}\left(\mathbf{h}_i^{(k)}\right) \tag{3}$$

which aggregates the node features to obtain the entire graph's representation $\mathbf{h}_G$.

Ideally, a maximally powerful GNN could distinguish different graph structures by mapping them to different representations in the embedding space. This ability to map any two different graphs to different embeddings, however, implies solving the challenging graph isomorphism problem. That is, we want isomorphic graphs to be mapped to the same representation and non-isomorphic ones to different representations. Thus, the expressiveness of a GNN is defined as the ability to distinguish non-isomorphic graphs, and can be analysed by comparing to the 1-WL graph isomorphism test, which are summarised in Appendix C.

3.3 Graph Generative Frameworks

In this paper, Graph Convolutional Policy Network (GCPN) [3] and Flow-Based Autoregressive Model (GraphAF) [4] are used as graph generative frameworks for molecular graph generation tasks. They formalize the problem of goal-directed graph generation as a sequential decision process through RL, i.e. the decisions are generated from the generation policy network.

In GCPN, the iterative graph generation process is formulated as a general decision process: $M = (S, \mathcal{A}, P, R, \gamma)$, where $S = \{s_i\}$ is the set of states that consists of all possible intermediate and final graphs, $\mathcal{A} = \{a_i\}$ is the set of actions that describe the modification made to the current graph at each time step determined by the Graph Convolutional Policy Network (a group of MLPs predicting a distribution of actions), P is the Markov transition dynamics that specifies the possible outcomes of carrying out an action, $p(s_{t+1}|s_t, \cdots, s_0, a_t) = p(s_{t+1}|s_t, a_t)$. $R(s_t)$ is a reward function that specifies the reward after reaching state s_t, and γ is the discount factor. GCPN takes the intermediate graph G_t and the collection of proposed scaffold subgraphs C as inputs, and outputs the action a_t, which predicts a new link to be added.

GraphAF defines an invertible transformation from a base distribution (e.g. multivariate Gaussian) to a molecular graph structure $G = (V, \mathcal{E})$ as the generation policy network. Starting from an empty graph G_0, in each step a new node v_i is generated based on the current sub-graph structure G_i, $i.e.$, $p(v_i|G_i)$ by the policy network. Next, the edges between this new node and existing nodes are sequentially generated according to the current graph structure, i.e., $p(\mathcal{E}_{ij}|G_i, v_i, \mathcal{E}_{i,1:j-1}, \mathcal{E}_{1:j-1,i})$. This process is repeated until all the nodes and edges are generated.

4 Evaluation

4.1 Experimental Setup

Dataset. We use the ZINC-250k [21] dataset for both pre-training and fine-tuning proposed models for its pharmaceutical relevance, moderate size, and popularity. All molecules are presented in the kekulised form with hydrogen removed. ZINC-250k contains around 250,000 drug-like molecules with 9 atom types and 3 edge types, and with a maximum graph size of 38 sampled from the ZINC database [21]. The molecules in ZINC are readily synthesisable molecules: *It contains over 120 million purchasable "drug-like" compounds—effectively all organic molecules that are for sale—a quarter of which are available for immediate delivery.* [21]. Other datasets, such as QM9 [22], which is a subset of GDB9 [23], contain, at least to some degree, virtual molecules that are likely to be synthesisable but have not been made yet, including many molecules with complex annulated ring systems [15]. In addition, the original works of GCPN, GraphAF and the benchmark [9] we used to compare graph generative models have also been trained on ZINC-250k. Thus, ZINC-250k is well-suited for models to learn representations of drug-like and synthesisable molecules in a de-novo molecule generation task.

Implementation details. We use the open-source platform TorchDrug [24] in dataset preparation and graph generative models training. Advanced GNNs are implemented in PyTorch [25] in the MPNN framework aligned with TorchDrug [24]. The basic architectures of GCPN and GraphAF are implemented in TorchDrug. Note that the original GraphAF cannot allow GNN module to aggregate edge features in the message-passing mechanism, since the intermediate molecular graph generated by the autoregressive flow doesn't contain edge features. Therefore, we propose an improved version of GraphAF considering edge features, named GraphAF+e. To conduct a fairly analogous evaluation on GCPN, we considered GCPN without considering edge features, named GCPN–e. A detailed description of how to incorporate edge features is in Appendix D.

In the experiments, we replace R-GCN in GCPN (both with and without edge features) and GraphAF (both with and without edge features) with six more expressive GNNs: GIN [26], GAT [27], GATv2 [6], PNA [28], GSN [7] and GearNet [8]. We set all GNNs in the experiments to have 3 hidden layers with batch normalisation [29] and ReLU [30] activation applied after each layer. We find little improvement when further adding GCN layers. Each GNN uses a 1-layer MLP to transform the edge features in the graph to a hidden embedding and concatenates with the node features for message passing. In addition, they use summation as the READOUT function for graph representations. For PNA and GSN, the MESSAGE and UPDATE functions are parameterised by single-layer perceptrons. For GSN, we set the graph substructure set to contain cycle graphs of sizes between 3 and 8 (both inclusive), which are some of the most important substructures in molecules [7]. For GAT and GATv2, we use multi-head attention with $k = 3$ after a manual grid search for attention heads.

The GCPN and GraphAF models, with all different GNNs, are pre-trained on the ZINC-250k dataset for 1 epoch, as we do not observe too much performance gain from increasing pre-training epochs. They are then fine-tuned towards the target properties with RL, using the proximal policy optimisation (PPO) algorithm [31]. The models are fine-tuned for 5 epochs for all goal-directed generation tasks with early stopping if all generation results in one batch collapse to singleton molecules. We set the agent update interval for GCPN and GraphAF models to update their RL agents every 3 and 5 batches respectively, as considering the cost of computing resources. During generating process, we allow our RL agents in all models to make 20 resamplings on the intermediate generated graphs if they cannot generate chemically valid molecular graphs. The max graph size is set as 48 empirically. For GCPN, the collection of proposed scaffold subgraphs C for GCPN are sampled from non-overlapping scaffolds in the ZINC-250k dataset. For GraphAF, we define multivariate Gaussian as our base distribution and use node MLPs and edge MLPs which have two fully-connected layers equipped

Table 1: Comparison of the top-3 Penalised logP, QED and SA scores of the generated molecules by GCPN variants, with the top-3 property scores of molecules in the ZINC dataset for reference.

Model	Penalised logP			QED			SA		
	1st	**2nd**	**3rd**	**1st**	**2nd**	**3rd**	**1st**	**2nd**	**3rd**
ZINC	4.52	4.30	4.23	0.948	0.948	0.948	1.0	1.0	1.0
R-GCN (baseline)	7.98	7.85	7.80	0.948	0.947	0.946	-	-	-
R-GCN (ours)	8.67	8.67	8.67	0.948	0.948	0.948	**1.0**	**1.0**	**1.0**
GIN	**11.19**	**11.19**	**11.19**	0.942	0.926	0.923	1.2	1.2	1.2
GAT	7.70	7.47	7.44	0.926	0.911	0.911	**1.0**	**1.0**	**1.0**
GATv2	8.08	7.48	7.34	0.945	0.908	0.907	**1.0**	**1.0**	**1.0**
PNA	8.66	8.61	8.22	0.833	0.825	0.754	**1.0**	**1.0**	**1.0**
GSN	**11.19**	**11.19**	**11.19**	0.804	0.783	0.783	**1.0**	**1.0**	**1.0**
GearNet	**11.19**	**11.19**	**11.19**	**0.948**	**0.948**	**0.948**	**1.0**	**1.0**	**1.0**

with tanh non-linearity to generate the nodes and edges respectively. We notice limited improvements in performance when increasing the number of MLP layers. Adam [32] is used as the optimiser for both pre-training and fine-tuning tasks. The experiments were run with a mix of NVIDIA A100 GPU with 40GB memory and NVIDIA V100 (Volta) GPU with 16GB. The total amount of training time for all GCPN and GraphAF variants under all metrics is around 1200 GPU hours. Reward temperature for each metric, the number of neurons in each hidden layer and the learning rate for fine-tuning) for each model on each task through grid search by Optuna [33] independently. All details are provided in Appendix D.

Baselines. We compare our proposed models based on the original GCPN (both with and without edge features) and GraphAF (both with and without edge features) on six goal-directed molecule generation tasks: Penalised logP [34], QED [35], synthetic accessibility (SA) [36], DRD2 [1], Median1 and Median2. All generation metrics are taken from the Therapeutic Data Commons (TDC) [37]. Due the practicality of de-novo molecule design, we only consider the single generation objective for all generation tasks. Specifically, SA stands for how hard or how easy it is to synthesize a given molecule. Penalized logP is a logP score that also accounts for ring size and SA, while QED is an indicator of drug-likeness. DRD2 is derived from a support vector machine (SVM) classifier with a Gaussian kernel fitting experimental data to predict the bioactivities against their corresponding disease targets. Median1 [37] measures the average score of the molecule's Tanimoto similarity [38] to Camphor and Menthol. Median2 [37] measures the average score of the molecule's Tanimoto similarity to Tadalafil and Sildenafil. Penalized logP has an unbounded range, while QED, DRD2, Median1, Median2 and SA have a range of $[0, 1]$ by definition. Higher scores in Penalized logP, QED, DRD2, Median1 and Median2 and a lower score in SA are desired. Note that all scores are calculated from empirical prediction models.

We choose to use DRD2, Median1 and Median2 to evaluate generative models, since they are mature and representative generative metrics [9, 15]. In addition, some other metrics are data-missing or inappropriate and thus cannot reflect the ability of generative models properly. For example, GSK3 cannot evaluate all the generated molecules; multi-property objectives (MPO) measure the geometric means of several scores, which will be 0 if one of the scores is 0; Valsartan Smarts is implemented incorrectly way in TDC: it computes the geometric means of several scores instead of arithmetic means of several scores, which lead to incorrect results in the benchmark [9].

We compare the best GCPN variant and GraphAF variant with eight state-of-the-art approaches for molecule generation [9] on 6 generation metrics: Penalised logP, SA, DRD2, Median1, Median2 and QED. All results of baselines are taken from original papers unless stated.

4.2 Results

4.2.1 Improving GNN-based Graph Generative Methods

De-novo molecule design with more expressive GNNs. As we re-evaluate the Penalised logP and QED scores of the top-3 molecules found by GCPN, we note that our results turn out to be higher than the results reported in the original GCPN paper. We hypothesise this to be due to our more

Table 2: Comparison of the top-3 DRD2, Median1 and Median2 scores of the generated molecules by GCPN–e variants, with the top-3 property scores of molecules in the ZINC dataset for reference.

Model	DRD2			Median1			Median2		
	1st	**2nd**	**3rd**	**1st**	**2nd**	**3rd**	**1st**	**2nd**	**3rd**
ZINC	0.9872	0.9815	0.9773	0.3243	0.3096	0.3096	0.2913	0.2765	0.2749
R-GCN	0.8315	0.7576	0.7551	0.3152	0.3152	0.3001	0.1932	0.1613	0.1592
GIN	0.2791	0.1980	0.1752	0.3152	0.3152	0.3152	0.1140	0.1113	0.1069
GAT	0.1980	0.1580	0.1580	0.3243	0.3243	0.3175	0.1196	0.1100	0.1098
GATv2	0.2992	0.2992	0.2992	0.3281	0.3281	0.3281	0.1042	0.1009	0.0911
PNA	0.4448	0.4448	0.4448	0.3243	0.3202	0.3175	0.0911	0.0764	0.0716
GSN	0.4790	0.4790	0.4448	0.3175	0.3175	0.3015	0.0982	0.0978	0.0897
GearNet	**0.9990**	**0.9705**	**0.9574**	**0.3482**	**0.3482**	**0.3482**	**0.2084**	**0.2043**	**0.2037**

Table 3: Comparison of the top-3 DRD2, Median1 and Median2 scores of the generated molecules by GCPN variants, with the top-3 property scores of molecules in the ZINC dataset for reference.

Model	DRD2			Median1			Median2		
	1st	**2nd**	**3rd**	**1st**	**2nd**	**3rd**	**1st**	**2nd**	**3rd**
ZINC	0.9872	0.9815	0.9773	0.3243	0.3096	0.3096	0.2913	0.2765	0.2749
R-GCN	0.4790	0.4790	0.4790	0.3367	0.3367	0.3242	0.1921	0.1891	0.1891
GIN	0.3460	0.3094	0.3094	0.3243	0.3243	0.3235	0.1770	0.1766	0.1730
GAT	0.4946	0.4946	0.4946	0.3367	0.3367	0.3328	0.1648	0.1640	0.1637
GATv2	0.5101	0.4946	0.4946	0.3367	0.3367	0.3331	0.1759	0.1720	0.1697
PNA	0.5828	0.4448	0.4448	0.3472	0.3254	0.3254	0.1629	0.1619	0.1605
GSN	0.5363	0.4946	0.4946	0.3243	0.3243	0.3235	0.1770	0.1766	0.1730
GearNet	**0.9696**	**0.9684**	**0.9404**	**0.3367**	**0.3367**	**0.3367**	**0.2862**	**0.2794**	**0.2794**

extensive hyperparameter searching. In Table 1, we explore a set of simpler generation metrics based on the GCPN framework, such as Penalised logP, QED and SA, which have been widely used as objectives in previous work on GNN-based graph generative models.

As summarised in Table 1, after replacing the inner R-GCN in GCPN with more expressive GNNs, we can observe a significant improvement of GCPN in Penalised logP: GCPN with GIN, GearNet and GSN can achieve the saturated 11.19 in Penalised logP. Moreover, with GearNet, the performance of the GCPN variants can outperform the original GCPN on all three metrics.

However, we find that *these benchmarks are not suited to differentiate between different models*, since we see many GCPN variants with different GNNs achieve similar and saturated results on those metrics. In addition, these metrics are not representative enough to obtain meaningful conclusions, as discussed in Section 2.1. Therefore, *there is a need of better graph generation objectives*.

De-novo molecule design with better graph generation objectives. From Table 1, we notice that both Penalised logP and SA are saturating on advanced GNNs, making them inappropriate metrics for distinguishing the capability of different GNN models, and we need better de-novo molecule generation metrics. Therefore, we introduce three more representative metrics: DRD2, Median1 and Median2, as described in Section 4.1, and report the top-3 property scores of molecules generated by each model trained on those three metrics in Table 3 (GCPN) and Table 6 (GraphAF) in Appendix B. The results show that GCPN and GraphAF with more expressive GNNs, such as GearNet, can outperform the original GCPN and GraphAF with R-GCN on all generation tasks by a significant margin. Specifically, on metrics such as DRD2 and Meidan2, GCPN and GraphAF with more expressive GNNs can vastly improve the original performance. This observation further indicates that by combining with more expressive GNNs, GCPN and GraphAF can successfully capture the distribution of desired molecules. Therefore, we suggest that DRD2, Median1, Median2 and QED are better graph generation metrics for differentiating different GNNs.

Table 4: Comparison of the top-1 DRD2, Median1, Median2 and QED scores with the selected non-GNN-based generative models. The full table can be found in Table 8 in Appendix B.

Model	DRD2	Median1	Median2	QED
GCPN (RGCN)	0.479	0.337	0.192	0.948
GCPN (GearNet)	0.970 (+**102.51**%)	0.337 (+**0.00**%)	0.286 (+**48.96**%)	0.948 (+**0.00**%)
GraphAF (RGCN)	0.928	0.281	0.143	0.946
GraphAF (GearNet)	0.987 (+**6.36**%)	0.290 (+**3.20**%)	0.183 (+**27.97**%)	0.947 (+**0.11**%)
LSTM HC (SMILES) [15]	0.999	0.388	0.339	0.948
DoG-Gen [16]	0.999	0.322	0.297	0.948
GP BO [12]	0.999	0.345	0.337	0.947
SynNet [10]	0.999	0.244	0.259	0.948
GA+D [11]	0.836	0.219	0.161	0.945
VAE BO (SMILES) [13]	0.940	0.231	0.206	0.947
Graph MCTS [14]	0.586	0.242	0.148	0.928
MolDQN [17]	0.049	0.188	0.108	0.871

Table 5: Graph classification metrics among GCPN and GraphAF variants. NLL_{all} means the average negative log likelihood loss for all node and edge classification tasks. Acc means the average accuracy for all node and edge classification tasks. NLL_e and NLL_n mean the average edge negative log likelihood evaluated on classification tasks for edge and node respectively.

Model	GCPN		GraphAF	
	NLL_{all}	Acc	NLL_e	NLL_n
RGCN	2.1427 ± 0.1326	0.8656 ± 0.0088	-1.0674 ± 0.0204	-3.3312 ± 0.1557
GIN	2.1500 ± 0.3158	0.8680 ± 0.0220	-1.1691 ± 0.1167	-3.6067 ± 0.2589
GAT	2.8705 ± 0.2354	0.7957 ± 0.0211	-1.2737 ± 0.0340	-3.1541 ± 0.0844
GATv2	2.8285 ± 0.2878	0.8013 ± 0.0244	-1.2426 ± 0.0370	-3.1325 ± 0.0681
PNA	2.1172 ± 0.0036	0.8716 ± 0.0003	-1.1043 ± 0.1148	-3.2840 ± 0.2154
GSN	**1.4219 ± 0.0026**	**0.9308 ± 0.0002**	-1.1319 ± 0.0678	-3.1280 ± 0.0566
GearNet	2.1265 ± 0.0017	0.8677 ± 0.0003	**-0.9275 ± 0.0048**	**-2.8981 ± 0.0398**

Comparison with non-GNN-based graph generative methods. We report the top-1 DRD2, Median1, Median2 and QED scores found by all the GNN-based and non-GNN-based graph generative models in Table 4. As displayed in Table 4, original GCPN and GraphAF are not competitive among graph generative models on all the goal-directed molecule generation tasks. However, after modifying their inner GNNs with more advanced GNNs, such as GearNet, they can outperform or match state-of-the-art results across other generative approaches on the de-novo molecule generation task. Specifically, Replacing R-GCN with GearNet can achieve an average of 50.49% improvement on GCPN and an average of 12.51% improvement on GraphAF in de-novo molecule design, with the proposed generation metrics.

Visualisations of the generated molecules with desired generative metrics by GCPN and GraphAF variants are presented in Figure 4 in Appendix E. In addition, we only report eight selected graph generative methods in the benchmark [9] in Table 4, and GCPN with GearNet can achieve comparable results across 17 other non-GNN-based graph generative methods on the proposed metrics, which are fully reported in Table 8 in Appendix B.

4.2.2 Correlation Between GNN Expressivenenss and Graph Generation

In the pre-training phase, the GNNs are used to predict all node types and edge types in all masked graphs in the training data. We report the graph classification results for all GNNs in Table 5. In the GCPN framework, we can see GSN perform the best with 93.08% accuracy on the graph classification task. In the GraphAF framework, we can see GearNet perform the best with the lowest edge and node average negative log likelihood: -0.9275 and -2.8981 respectively. Both GSN and GearNet are more expressive GNNs than R-GCN according to the 1-WL test demonstrated in Appendix C. It is not surprising that expressive GNNs can outperform other GNNs on the graph classification task, since the expressivity of a GNN is defined as the ability to distinguish non-isomorphic graphs.

However, by comparing Table 5 with Tables 2, 3, 6, 7, it is worth noting that *more expressive GNNs cannot ensure better performance of GNN-based graph generative models in molecular generation* *tasks*. For example, PNA and GSN perform better than R-GCN on graph classification tasks (Table 5), but GCPN with PNA or GSN cannot surpass the original GCPN on all generation metrics (e.g. Median2 in Table 3). Therefore, we conclude that the graph generation task is a different task from graph classification and requires other abilities of GNNs, such as edge feature extraction.

De-novo molecule design with GNNs incorporating edge features. We investigate the performance of GCPN and GraphAF using GNNs with and without edge features. It is worth mentioning that the original GCPN considers edge features but GraphAF does not. The results of the GCPN without edge features (GCPN–e) are summarised in Table 2, and those of the original GCPN are in Table 3. Both sets of results demonstrate that including edge features can significantly improve the top-3 scores on all three metrics for most GNNs. For instance, with the help of edge feature extraction, GCPN with GIN, GAT, GATv2, PNA and GSN can increase by 24.0%, 149.8%, 70.5%, 31.0% and 12.0% on the top-1 score on the metric DRD2, compared with GCPN–e.

The original GraphAF comes without edge feature extraction and we thus improved it by incorporating the edge features. Table 6 (the orginal GraphAF) and Table 7 (GraphAF+e) in Appendix B summarised the results of both implementations accordingly. The results align with the GCPN case, where GraphAF+e significantly improves the top-3 scores on all three metrics for most GNNs than the original GraphAF. More specifically, with the help of edge feature extraction, GraphAF+e with GIN, GAT, GATv2, PNA and GSN can be improved by 43.5%, 60.0%, 20.0%, 230.4% and 312.4% on the top-1 score on the metric DRD2.

The results above indicate *the importance of aggregating edge features for GNN-based generative models*. By harnessing the power to extract knowledge from edge information, the potential for generating more refined and accurate graphs is enhanced. We also notice that, when the GNNs used are either R-GCN or GearNet, we find GCPN and GraphAF perform well with and without considering edge information, so we hypothesise the reason is that they absorb the number of edge relations as prior information in the model.

In summary, we conclude the following results:

1. More expressive GNN can lead to better results in the graph classification task. However, GNN expressiveness defined by the 1-WL test is not a necessary condition for a good GNN-based generative model. Generation tasks require other abilities of GNNs, such as edge feature extraction and edge relation detection.
2. Although Penalised logP and QED are widely used for generative metrics in evaluating goal-directed graph generative models, they are not effective metrics to differentiate different generative models. Other metrics, such as DRD2, Median1 and Median2, can better evaluate the ability of a graph generative model. Under those metrics, we can see the performance of GCPN and GraphAF can be enhanced by using more robust GNNs.
3. After applying advanced GNN to current GNN-based graph generative methods, such as GCPN and GraphAF, they can outperform or match state-of-the-art results across 17 other generative approaches in the de-novo molecule generation task.

5 Limitation and Conclusion

Due to computation cost, we acknowledge several limitations of the current study: we cannot exhaustively explore every method, such as other Relational GNNs, and thoroughly tune every hyperparameter; we cannot evaluate all generative models on other complicated datasets besides Zinc 250k, such as ChEMBL [39], and other generation metrics [9]. However, our efforts have still provided valuable insights into investigating the expressiveness of GNN on the graph generation task, because of our focus on many different generative models and diverse generation objectives.

After exploring (1) unexplored underlying GNNs and (2) non-trivial generative objectives, we would like to conclude that expressiveness is not a necessary condition for a good GNN-based generative model. By evaluating GCPN variants and GraphAF variants on effective metrics, we demonstrate that GCPN and GraphAF can be improved by using more robust GNNs (e.g. strong edge features extraction and edge relation detection). With more robust GNNs, GNN-based graph generative models can outperform or match state-of-the-art results across 17 other generative approaches on de-novo molecule design tasks [9]. In the future, we plan to explore the necessary conditions for GNN to enhance the performance of GNN-based graph generative models.

References

[1] Marcus Olivecrona, Thomas Blaschke, Ola Engkvist, and Hongming Chen. Molecular de-novo design through deep reinforcement learning. In *Journal of Cheminformatics*, 2017. 1, 6, 15

[2] Pavel G Polishchuk, Timur I Madzhidov, and Alexandre Varnek. Estimation of the size of drug-like chemical space based on GDB-17 data. In *Journal of Computer-Aided Molecular Design*, 2013. 1

[3] Jiaxuan You, Bowen Liu, Rex Ying, Vijay Pande, and Jure Leskovec. Graph Convolutional Policy Network for Goal-Directed Molecular Graph Generation. In *NeurIPS*, 2018. 1, 2, 4

[4] Chence Shi, Minkai Xu, Zhaocheng Zhu, Weinan Zhang, Ming Zhang, and Jian Tang. GraphAF: a Flow-based Autoregressive Model for Molecular Graph Generation. In *ICLR*, 2020. 1, 2, 4, 20

[5] Michael Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. Modeling Relational Data with Graph Convolutional Networks. In *The Semantic Web: 15th International Conference*, 2018. 1, 2, 16, 17

[6] Shaked Brody, Uri Alon, and Eran Yahav. How Attentive are Graph Attention Networks? In *ICLR*, 2022. 1, 2, 5, 18, 19

[7] Giorgos Bouritsas, Fabrizio Frasca, Stefanos Zafeiriou, and Michael M. Bronstein. Improving Graph Neural Network Expressivity via Subgraph Isomorphism Counting. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022. 2, 5, 17

[8] Zuobai Zhang, Minghao Xu, Arian Jamasb, Vijil Chenthamarakshan, Aurelie Lozano, Payel Das, and Jian Tang. Protein Representation Learning by Geometric Structure Pretraining. In *ICLR*, 2023. 1, 2, 5, 17, 18

[9] Wenhao Gao, Tianfan Fu, Jimeng Sun, and Connor W. Coley. Sample Efficiency Matters: A Benchmark for Practical Molecular Optimization. In *NeurIPS*, 2022. 2, 5, 6, 8, 9, 15

[10] Wenhao Gao, Rocío Mercado, and Connor W Coley. Amortized tree generation for bottom-up synthesis planning and synthesizable molecular design. In *ICLR*, 2022. 2, 8, 13, 15

[11] AkshatKumar Nigam, Pascal Friederich, Mario Krenn, and Alán Aspuru-Guzik. Augmenting genetic algorithms with deep neural networks for exploring the chemical space. In *ICLR*, 2020. 8, 13, 15

[12] Austin Tripp, Gregor NC Simm, and José Miguel Hernández-Lobato. A fresh look at de novo molecular design benchmarks. In *NeurIPS 2021 AI for Science Workshop*, 2021. 8, 13, 15

[13] Rafael Gómez-Bombarelli, Jennifer N Wei, David Duvenaud, José Miguel Hernández-Lobato, Benjamín Sánchez-Lengeling, Jorge Aguilera-Iparraguirre Dennis Sheberla, Timothy D Hirzel, Ryan P Adams, and Alán Aspuru-Guzik. Automatic chemical design using a data-driven continuous representation of molecules. In *ACS central science*, 2018. 8, 13, 15

[14] Jan H Jensen. A graph-based genetic algorithm and generative model/monte carlo tree search for the exploration of chemical space. In *Chemical science*, 2019. 8, 13, 15

[15] Nathan Brown, Marco Fiscato, Marwin H.S. Segler, and Alain C. Vaucher. GuacaMol: Benchmarking Models for de Novo Molecular Design. In *Journal of Chemical Information and Modeling*, 2019. 2, 5, 6, 8, 13, 15

[16] John Bradshaw, Brooks Paige, Matt J Kusner, Marwin Segler, and José Miguel Hernández-Lobato. Barking up the right tree: an approach to search over molecule synthesis dags. In *Advances in Neural Information Processing Systems*, 2020. 8, 13, 15

[17] Zhenpeng Zhou, Steven Kearnes, Li Li, Richard N Zare, and Patrick Riley. Optimization of molecules via deep reinforcement learning. In *Scientific reports*, 2019. 2, 8, 13, 15

[18] Tianfan Fu, Wenhao Gao, Cao Xiao, Jacob Yasonik, Connor W. Coley, and Jimeng Sun. Differentiable Scaffolding Tree for Molecular Optimization. In *ICLR*, 2022. 2, 15

[19] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P Adams, and Nando De Freitas. A review of Bayesian optimization. In *Proceedings of the IEEE*, 2015. 3, 13

[20] Diederik P Kingma and Max Welling. Auto-encoding variational Bayes. In *ICLR*, 2014. 3, 13

[21] Teague Sterling and John J Irwin. ZINC 15–ligand discovery for everyone. In *Journal of chemical information and modeling*, 2015. 5

[22] Raghunathan Ramakrishnan, Pavlo O Dral, Matthias Rupp, and O Anatole von Lilienfeld. Quantum chemistry structures and properties of 134 kilo molecules. In *Scientific data*, 2014. 5

[23] Lars Ruddigkeit, Ruud van Deursen, Lorenz C. Blum, and Jean-Louis Reymond. Enumeration of 166 billion organic small molecules in the chemical universe database GDB-17. In *Journal of Chemical Information and Modeling*, 2012. 5

[24] Zhaocheng Zhu, Chence Shi, Zuobai Zhang, Shengchao Liu, Minghao Xu, Xinyu Yuan, Yangtian Zhang, Junkun Chen, Huiyu Cai, Jiarui Lu, Chang Ma, Runcheng Liu, Louis-Pascal Xhonneux, Meng Qu, and Jian Tang. TorchDrug: A Powerful and Flexible Machine Learning Platform for Drug Discovery. 2022. 5

[25] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in PyTorch. In *NIPS 2017 Workshop Autodiff*, 2017. 5

[26] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How Powerful are Graph Neural Networks? In *ICLR*, 2019. 5, 16

[27] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph Attention Networks. In *ICLR*, 2018. 5, 18

[28] Gabriele Corso, Luca Cavalleri, Dominique Beaini, Pietro Liò, and Petar Veličković. Principal Neighbourhood Aggregation for Graph Nets. In *NeurIPS*, 2020. 5, 16, 17

[29] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning*, 2015. 5

[30] Abien Fred Agarap. Deep Learning using Rectified Linear Units (ReLU). 2018. 5

[31] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms. 2017. 5

[32] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference for Learning Representations*, 2015. 6

[33] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A Next-generation Hyperparameter Optimization Framework. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery and data mining*, 2019. 6

[34] Matt J. Kusner, Brooks Paige, and José Miguel Hernández-Lobato. Grammar Variational Autoencoder. In *Proceedings of the 34th International Conference on Machine Learning*, 2017. 6

[35] G. Richard Bickerton, Gaia V. Paolini, Jérémy Besnard, Sorel Muresan, and Andrew L. Hopkins. Quantifying the chemical beauty of drugs. In *Nature Chemistry*, 2012. 6

[36] Peter Ertl and Ansgar Schuffenhauer. Estimation of synthetic accessibility score of drug-like molecules based on molecular complexity and fragment contributions. In *Journal of Cheminformatics*, 2009. 6

[37] Kexin Huang, Tianfan Fu, Wenhao Gao, Yue Zhao, Yusuf Roohani, Jure Leskovec, Connor W Coley, Cao Xiao, Jimeng Sun, and Marinka Zitnik. Therapeutics data commons: Machine learning datasets and tasks for therapeutics. In *NeurIPS Track Datasets and Benchmarks*, 2021. 6

[38] Dávid Bajusz, Anita Rácz, and Károly Héberger. Why is Tanimoto index an appropriate choice for fingerprint-based similarity calculations? In *Journal of Cheminformatics*, 2015. 6

[39] Anna Gaulton, Louisa J. Bellis, A. Patricia Bento, Jon Chambers, Mark Davies, Anne Hersey, Yvonne Light, Shaun McGlinchey, David Michalovich, Bissan Al-Lazikani, and John P. Overington. ChEMBL: a large-scale bioactivity database for drug discovery. In *Nucleic Acids Research*, 2012. 9

[40] AkshatKumar Nigam, Robert Pollice, Mario Krenn, Gabriel dos Passos Gomes, and Alan Aspuru-Guzik. Beyond generative models: superfast traversal, optimization, novelty, exploration and discovery (STONED) algorithm for molecules using SELFIES. In *Chemical science*, 2021. 15

[41] Yutong Xie, Chence Shi, Hao Zhou, Yuwei Yang, Weinan Zhang, Yong Yu, and Lei Li. MARS: Markov molecular sampling for multi-objective drug discovery. In *ICLR*, 2021. 15

[42] David E Graff, Eugene I Shakhnovich, and Connor W Coley. Accelerating high-throughput virtual screening through molecular pool-based active learning. In *Chemical science*, 2021. 15

[43] Tianfan Fu, Cao Xiao, Xinhao Li, Lucas M Glass, and Jimeng Sun. MIMOSA: Multi-constraint molecule sampling for molecule optimization. In *AAAI*, 2021. 15

[44] Fredrik Svensson, Ulf Norinder, and Andreas Bender. Improving screening efficiency through iterative screening using docking and conformal prediction. In *Journal of chemical information and modeling*, 2017. 15

[45] Yoshua Bengio, Tristan Deleu, Edward J. Hu, Salem Lahlou, Mo Tiwari, and Emmanuel Bengio. GFlowNet foundations. In *CoRR*, 2021. 15

[46] Cynthia Shen, Mario Krenn, Sagi Eppel, and Alan Aspuru-Guzik. Deep molecular dreaming: Inverse machine learning for de-novo molecular design and interpretability with surjective representations. In *Machine Learning: Science and Technology*, 2021. 15

[47] Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Junction tree variational autoencoder for molecular graph generation. In *ICML*, 2018. 15

[48] Thomas N. Kipf and Max Welling. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*, 2017. 16

A Appendix: Non-GNN-based Graph Generative Models

Genetic Algorithm (GA). The Genetic Algorithm (GA) is a widely used heuristic technique that draws inspiration from natural evolutionary mechanisms. It combines mutation and/or crossover perturbing a mating pool to enable exploration in the design space. SynNet [10] utilizes a genetic algorithm on binary fingerprints and subsequently decodes them into synthetic pathways. GA+D [11] constitutes a genetic algorithm improved through the incorporation of a neural network (DNN) based discriminator model.

Bayesian Optimization (BO) [19]. Bayesian Optimization (BO) represents a broad category of techniques that constructs a surrogate model for the objective function through the application of Bayesian machine learning methods like Gaussian process (GP) regression. It then employs an acquisition function that integrates information from the surrogate model and its associated uncertainty to determine optimal sampling points. GPBO [12] optimizes the GP acquisition function with Graph GA methods in an inner loop.

Variational Autoencoders (VAEs) [20]. Variational Autoencoders (VAEs) belong to a category of generative techniques that focus on maximizing a lower bound of the likelihood, known as the evidence lower bound (ELBO), as opposed to directly estimating the likelihood itself. A VAE typically learns to map molecules to and from real space to enable the indirect optimization of molecules by numerically optimizing latent vectors. SMILES-VAE [13] uses a VAE to model molecules represented as SMILES strings.

Monte-Carlo Tree Search (MCTS). Monte-Carlo Tree Search (MCTS) conducts a localized and stochastic exploration of each branch originating from the present state, which could be a molecule or an incomplete molecule. It then identifies the most promising branches, which usually exhibit the highest property scores, to be considered for the subsequent iteration. Graph-MCTS [14] is an MCTS algorithm based on atom-level searching over molecular graphs.

Hill Climbing (HC). Hill Climbing (HC) is an iterative learning method that incorporates the generated high-scored molecules into the training data and fine-tunes the generative model for each iteration. SMILES-LSTM [15] leverages a LSTM to learn the molecular distribution represented in SMILES strings, and modifies it to a SELFIES version. DoG-Gen [16] instead learn the distribution of synthetic pathways as Directed Acyclic Graph (DAGs) with an RNN generator.

Reinforcement Learning (RL). In molecular design, a state is usually a partially generated molecule; actions are manipulations at the level of graph or string representations; rewards are determined by the desired properties of the molecules generated. MolDQN [17] uses a deep Q-network to generate molecular graphs in an atom-wise manner.

B Appendix: Additional Results

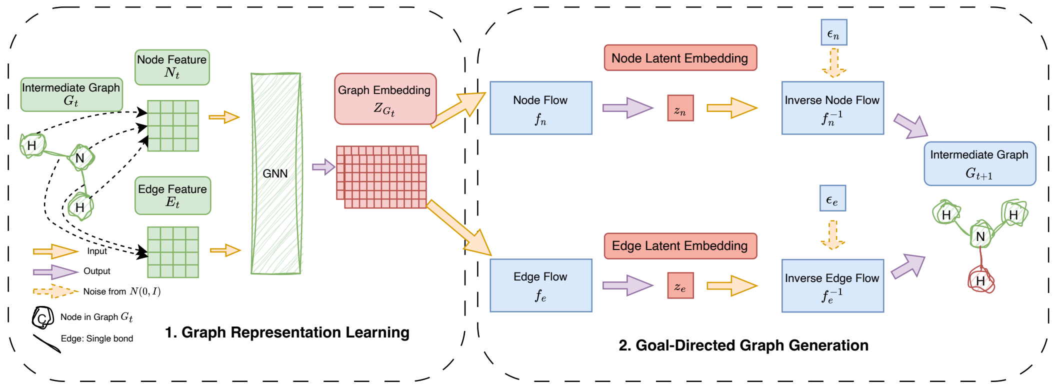


Figure 2: An overview of the GraphAF model: demonstrating an example of iterative graph generation from an intermediate graph G_t to an intermediate graph G_{t+1}. Part 1 is the illustration of the graph representation learning process based on a GNN. Part 2 is the illustration of the graph generative procedure based on a flow-based model. New nodes or edges are marked in red.

Table 6: Comparison of the top-3 DRD2, Median1 and Median2 scores of the generated molecules by GraphAF variants, with the top-3 property scores of molecules in the ZINC dataset for reference.

Model	DRD2			Median1			Median2		
	1st	2nd	3rd	1st	2nd	3rd	1st	2nd	3rd
ZINC	0.9872	0.9815	0.9773	0.3243	0.3096	0.3096	0.2913	0.2765	0.2749
R-GCN	0.9277	0.9133	0.9080	0.2810	0.2641	0.2449	0.1426	0.1426	0.1417
GIN	0.5847	0.1835	0.1495	0.2651	0.2649	0.2393	0.1094	0.1042	0.1037
GAT	0.2992	0.2992	0.2992	0.2453	0.2212	0.2208	0.1031	0.1025	0.1023
GATv2	0.5909	0.4790	0.4790	0.2437	0.2335	0.2331	0.1239	0.1239	0.1232
PNA	0.1495	0.1411	0.1411	0.2897	0.2651	0.2651	0.1023	0.0764	0.0761
GSN	0.1238	0.0614	0.0601	0.2896	0.2773	0.2449	0.1025	0.1017	0.0977
GearNet	**0.9872**	**0.9725**	**0.9714**	**0.2897**	**0.2896**	**0.2651**	**0.1826**	**0.1666**	**0.1616**

Table 7: Comparison of the top-3 DRD2, Median1 and Median2 scores of the generated molecules by improved GraphAF+e variants, with the top-3 property scores of molecules in the ZINC dataset for reference.

Model	DRD2			Median1			Median2		
	1st	2nd	3rd	1st	2nd	3rd	1st	2nd	3rd
ZINC	0.9872	0.9815	0.9773	0.3243	0.3096	0.3096	0.2913	0.2765	0.2749
R-GCN	0.7833	0.7702	0.5714	**0.2651**	**0.2667**	**0.2632**	0.1526	0.1507	0.1481
GIN	0.8391	0.8016	0.7145	0.2735	0.2343	0.2304	0.1505	0.1496	0.1459
GAT	0.4790	0.4790	0.4790	0.2651	0.2509	0.2471	0.1358	0.1328	0.1152
GATv2	0.7087	0.5452	0.3986	0.2887	0.2342	0.2331	0.1362	0.1343	0.1316
PNA	0.4940	0.4940	0.4940	0.2633	0.2633	0.2449	0.1423	0.1313	0.111
GSN	0.5106	0.4940	0.4940	0.2572	0.2530	0.2518	0.1448	0.1420	0.1090
GearNet	**0.9699**	**0.9688**	**0.9623**	0.2810	0.2582	0.2453	**0.1829**	**0.1795**	**0.1672**

Table 8: Comparison of the top-1 DRD2, Median1, Median2 and QED scores among all non-GNN-based graph generative models in the benchmark [9].

Model	DRD2	Median1	Median2	QED
GCPN (RGCN)	0.479	0.337	0.192	0.948
GCPN (GearNet)	0.970 (+**102.51**%)	0.337 (+**0.00**%)	0.286 (+**48.96**%)	0.948 (+**0.00**%)
GraphAF (RGCN)	0.928	0.281	0.143	0.946
GraphAF (GearNet)	0.987 (+**6.36**%)	0.290 (+**3.20**%)	0.183 (+**27.97**%)	0.947 (+**0.11**%)
REINVENT (SMILES) [1]	0.999	0.399	0.332	0.948
LSTM HC (SMILES) [15]	0.999	0.388	0.339	0.948
Graph GA [14]	0.999	0.350	0.324	0.948
REINVENT (SELFIES) [1]	0.999	0.399	0.313	0.948
DoG-Gen [16]	0.999	0.322	0.297	0.948
GP BO [12]	0.999	0.345	0.337	0.947
STONED [40]	0.997	0.295	0.265	0.947
LSTM HC (SELFIES) [15]	0.999	0.362	0.274	0.948
DST [18]	0.999	0.281	0.201	0.947
SMILES GA [15]	0.986	0.207	0.210	0.948
SynNet [10]	0.999	0.244	0.259	0.948
MARS [41]	0.994	0.233	0.203	0.946
MolPal [42]	0.964	0.309	0.273	0.948
MIMOSA [43]	0.993	0.296	0.238	0.947
GA+D [11]	0.836	0.219	0.161	0.945
VAE BO (SELFIES) [13]	0.940	0.231	0.206	0.947
DoG-AE [16]	0.999	0.203	0.201	0.944
Screening [44]	0.949	0.271	0.244	0.947
GFlowNet [45]	0.951	0.237	0.198	0.945
VAE BO (SMILES) [13]	0.940	0.231	0.206	0.947
Pasithea [46]	0.592	0.216	0.194	0.943
JT-VAE BO [47]	0.778	0.212	0.192	0.946
GFlowNet-AL [45]	0.863	0.229	0.191	0.944
Graph MCTS [14]	0.586	0.242	0.148	0.928
MolDQN [17]	0.049	0.188	0.108	0.871

C Appendix: Graph Neural Networks

1-WL test. Similar to GNNs, the 1-WL test iteratively updates the node embeddings of a graph by neighbourhood aggregation: for each node $i \in V$ in a graph, an initial colour $c_i^{(0)}$ is assigned, and is iteratively updated using random hashes of sums:

$$c_i^{(t+1)} = \text{HASH}\left(\sum_{j \in \mathcal{N}_i} c_j^{(t)}\right) \tag{4}$$

The 1-WL test terminates when stable node colouring of the graph is reached, and outputs a histogram of colours. Two graphs with different colour histograms are non-isomorphic, and two graphs with the same colour histograms are possibly, but not necessarily, isomorphic. The neighbourhood aggregation in the 1-WL test can also be seen as a form of message passing, with GNNs being the learnable analogue. It has been proved that *GNNs are at most as expressive as the 1-WL test over discrete features*.

Relational Graph Convolutional Network (R-GCN). The graph convolution operation of the original GCN [48] can be defined as follows:

$$\mathbf{h}_i^{(l+1)} = \sigma\left(\sum_{j \in \mathcal{N}_i} c_{ij} \mathbf{W}^{(l)} \mathbf{h}_j^{(l)}\right) \tag{5}$$

where c_{ij} is a normalization constant for each edge $\mathcal{E}_{ij}$ which originates from using the symmetrically normalized adjacency matrix $\mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$, $\mathbf{W}^{(l)}$ is a learnable weight matrix, and σ is a non-linear activation function. R-GCN [5] makes use of the relational data of the graphs, and extends the graph convolution operation to the following: let $\mathcal{R}$ be the edge relation type (for molecular graphs, this can be the bond type), then

$$\mathbf{h}_i^{(l+1)} = \sigma\left(\sum_{r \in \mathcal{R}} \sum_{j \in \mathcal{N}_i^r} c_{i,r} \mathbf{W}_r^{(l)} \mathbf{h}_j^{(l)} + \mathbf{W}_0^{(l)} \mathbf{h}_i^{(l)}\right) \tag{6}$$

where $\mathcal{N}_i^r$ denotes the set of neighbouring nodes of node i under relation $r \in \mathcal{R}$, $c_{i,r}$ is a problem-specific normalisation constant that can either be learnt or chosen in advance and $\mathbf{W}_r^{(l)}$ denotes the learnable matrix for edge type r. It has been shown that *neither GCN nor R-GCN is as expressive as the 1-WL test* [26].

Graph Isomorphism Network (GIN). Each GIN [26] layer updates the node features as follows:

$$\mathbf{h}_i^{(l+1)} = \phi^{(l)}\left(\left(1 + \epsilon^{(l)}\right) \mathbf{h}_i^{(l)} + \sum_{j \in \mathcal{N}_i} \mathbf{h}_j^{(l)}\right) \tag{7}$$

where $\phi^{(l)}$ is an MLP, and $\epsilon^{(l)}$ is a learnable scalar. *GIN is provably as expressive as the 1-WL test*, which makes it one of the maximally-expressive GNNs (proof in [26]).

Principal Neighbourhood Aggregation (PNA). The PNA [28] operator defines its aggregation function $\bigoplus$ as a combination of neighbourhood-aggregators and degree-scalers, as defined by the following equation, with $\otimes$ being the tensor product:

$$\bigoplus = \underbrace{\begin{bmatrix} \text{identity} \\ \text{amplification} \\ \text{attenuation} \end{bmatrix}}_{\text{scalers}} \otimes \underbrace{\begin{bmatrix} \text{mean} \\ \text{max} \\ \text{min} \\ \text{std} \end{bmatrix}}_{\text{aggregators}} \tag{8}$$

The PNA operator can then be inserted into the standard MPNN framework, obtaining the following PNA layer:

$$\mathbf{h}_i^{(l+1)} = \phi^{(l)} \left(\mathbf{h}_i^{(l)}, \bigoplus_{j \in \mathcal{N}_i} \psi^{(l)} \left(\mathbf{h}_i^{(l)}, \mathbf{h}_j^{(l)}, \mathbf{e}_{ij} \right) \right) \tag{9}$$

where $\phi^{(l)}$ and $\psi^{(l)}$ are MLPs. According to the theorem that *in order to discriminate between multisets of size n whose underlying set is R, at least n aggregators are needed* (proof in [28]), PNA pushes its expressivity closer towards the 1-WL limit than GIN, by including more aggregators and therefore increasing the probability that at least one of the aggregators can distinguish different graphs.

Graph Substructure Network (GSN). GSN [7] adopts a feature-augmented message passing style by counting the appearance of certain graph substructures and encoding them into the features. The feature augmentation of GSN then works as follows: let $\mathcal{G} = \{G_1, \cdots, G_K\}$ be a set of pre-computed small (connected) graphs. For each $G_k = (V_k, \mathcal{E}_k)$ in $\mathcal{G}$, we first find its isomorphic subgraphs $G'_k = (V'_k, \mathcal{E}'_k)$ in $G = (V, \mathcal{E})$. Then, for each node $i \in V$ and $1 \leq k \leq K$, we count the number of subgraphs G'_k node i belongs to, as defined by the equation below:

$$x_{G_k}^V(i) = |\{G'_k \cong G_k | i \in V'_k\}| \tag{10}$$

We then obtain the node structural features for each node $i \in V$: $\mathbf{x}_i^V = \left[x_{G_1}^V(i), \cdots, x_{G_k}^V(i)\right] \in \mathbb{N}^K$. Similarly, we can derive the edge structural features for each edge $(i,j) \in \mathcal{E}$: $\mathbf{x}_{ij}^{\mathcal{E}} = \left[x_{G_1}^{\mathcal{E}}(i,j), \cdots, x_{G_k}^{\mathcal{E}}(i,j)\right] \in \mathbb{N}^K$ by counting the numbers of subgraphs it belongs to:

$$x_{G_k}^{\mathcal{E}}(i,j) = |\{G'_k \cong G_k | i \in \mathcal{E}'_k\}| \tag{11}$$

The augmented features can then be inserted into the messages and follow the standard MPNN, obtaining two variants of GSN layer, GSN-v (vertex-count) and GSN-e (edge-count):

$$\begin{aligned} \mathbf{h}_i^{(l+1)} &= \phi^{(l)} \left(\mathbf{h}_i^{(l)}, \bigoplus_{j \in \mathcal{N}_i} \psi^{(l)} \left(\mathbf{h}_i^{(l)}, \mathbf{h}_j^{(l)}, \mathbf{x}_i^V, \mathbf{x}_j^V, \mathbf{e}_{ij} \right) \right) \text{(GSN-v)} \\ \mathbf{h}_i^{(l+1)} &= \phi^{(l)} \left(\mathbf{h}_i^{(l)}, \bigoplus_{j \in \mathcal{N}_i} \psi^{(l)} \left(\mathbf{h}_i^{(l)}, \mathbf{h}_j^{(l)}, \mathbf{x}_{ij}^{\mathcal{E}}, \mathbf{e}_{ij} \right) \right) \text{(GSN-e)} \end{aligned} \tag{12}$$

where $\phi^{(l)}$ and $\psi^{(l)}$ are MLPs and $\bigoplus$ is a permutation-invariant local neighbourhood aggregation function, such as sum, mean or max. It can be proved that *GSN is strictly more expressive than the 1-WL test, when G_k is any graph except for the star graphs (i.e., one center nodes connected to one of multiple outer nodes) of any size, and structural features are inferred by subgraph matching* (proof in [7]). This essentially suggests that GSN is more expressive than R-GCN which is at most as expressive as the 1-WL test in general.

Geometry Aware Relational Graph Neural Network (GearNet). GearNet [8] uses an R-GCN [5] as a fundamental framework to develop a node features and edge features message passing mechanism. Each GearNet layer updates the node features as follows:

$$\mathbf{h}_i^{(l+1)} = \sigma \left(\text{BN} \left(\sum_{r \in \mathcal{R}} \mathbf{W}_r^{(l)} \sum_{j \in \mathcal{N}_i^r} \mathbf{h}_j^{(l)} \right) \right) + \mathbf{h}_i^{(l)} \text{(GearNet-v)} \tag{13}$$

where $\mathcal{R}$ is the edge relation type, $\mathcal{N}_i^r$ denotes the set of neighbouring nodes of node i under relation $r \in \mathcal{R}$, BN denotes a batch normalization layer, $\mathbf{W}_r^{(l)}$ is the learnable convolutional kernel matrix for edge type r, and σ is a non-linear activation function.

To model the interactions between edges, we first construct a relational graph $G' = (V', \mathcal{E}', \mathcal{R}')$ among edges. Each node in the graph G' corresponds to an edge in the original graph. G' links edge (i, j, r_1) in the original graph to edge (w, k, r_2) if and only if $j = w$ and $i \neq k$. The type of this edge is determined by the angle between (i, j, r_1) and (w, k, r_2). *The angular information reflects the relative position between two edges that determines the strength of their interaction*[8]. Similar to R-GCN, the GearNet edge message passing layer is defined as:

$$\mathbf{e}_{i,j,r_1}^{(l+1)} = \sigma\left(\text{BN}\left(\sum_{r\in\mathcal{R}'} \mathbf{W}_r'^{(l)} \sum_{(w,k,r_2)\in\mathcal{N}'^{r}_{(i,j,r_1)}} \mathbf{e}_{(i,j,r_1)}^{(l)}\right)\right) \text{(GearNet-e)} \tag{14}$$

Similar as Eq.(13), the message function for edge (i, j, r_1) will be updated by aggregating features from its neighbours $\mathcal{N}'^{r}_{(i,j,r_1)}$, where $\mathcal{N}'^{r}_{(i,j,r_1)} = \{(w, k, r_2) \in V' | ((w, k, r_2), (i, j, r_1), r) \in \mathcal{E}'\}$.

Finally, the entire GearNet message passing layer can be expressed as:

$$\mathbf{h}_i^{(l+1)} = \sigma\left(\text{BN}\left(\sum_{r\in\mathcal{R}} \mathbf{W}_r^{(l)} \sum_{j\in\mathcal{N}_i^r} \left(\mathbf{h}_j^{(l)} + \text{FC}\left(\mathbf{e}_{j,i,r_1}^{(l)}\right)\right)\right)\right) + \mathbf{h}_i^{(l)} \tag{15}$$

where FC$(\cdot)$ denotes a linear transformation upon the message function. GearNet is more expressive than R-GCN due to its sparse edge message passing mechanism which encodes spatial information in a graph.

Graph Attention Network (GAT). In order to generalize the standard averaging or max-pooling aggregators in GNNs, GAT [27] applies attention-based neighbourhood aggregation as its aggregation function to obtain sufficient expressive power to transform the input features into higher-level features. The normalized masked attention coefficient for node i is defined as:

$$\forall j \in \mathcal{N}_i, \alpha_{ij} = \frac{\exp\left(\text{LeakyReLU}\left(\mathbf{a}\left[\mathbf{W}\mathbf{h}_i \| \mathbf{W}\mathbf{h}_j\right]\right)\right)}{\sum_{k\in\mathcal{N}_i} \exp\left(\text{LeakyReLU}\left(\mathbf{a}[\mathbf{W}\mathbf{h}_i \| \mathbf{W}\mathbf{h}_k]\right)\right)} \tag{16}$$

where **a** is a learnable weight vector, representing the attention mechanism a: a single-layer feedforward neural network, **W** is a learnable input linear transformation's weight matrix and $\|$ represents concatenation operation.

Each K multi-head attention GAT layer updates the node features as follows:

$$\mathbf{h}_i^{(l+1)} = \Big\|_{k=1}^{K} \sigma\left(\sum_{j\in\mathcal{N}_i} \alpha_{ij}^k \mathbf{W}^k \mathbf{h}_j^{(l)}\right) \tag{17}$$

where $\|$ represents concatenation, α_{ij}^k are normalized attention coefficients computed by the k-th attention mechanism (a^k), and $\mathbf{W}^k$ is the corresponding input linear transformation's weight matrix. GAT computes a representation for every node as a weighted average of its neighbours through the attention mechanism, which is more flexible than the neighbourhood aggregation in R-GCN.

Graph Attention Network v2 (GATv2). GATv2 [6] adopts a strictly more expressive *dynamic graph attention mechanism* [6] in its aggregation function to learn the graph representation. The graph attention variant that has a universal approximator attention function.

The normalized masked dynamic attention coefficient for node i is defined as:

$$\forall j \in \mathcal{N}_i, \alpha_{ij} = \frac{\exp\left(\mathbf{a}\left(\text{LeakyReLU}\left(\mathbf{W} \cdot [\mathbf{h}_i \| \mathbf{h}_j]\right)\right)\right)}{\sum_{k\in\mathcal{N}_i} \exp\left(\mathbf{a}\left(\text{LeakyReLU}\left(\mathbf{W} \cdot [\mathbf{h}_i \| \mathbf{h}_k]\right)\right)\right)} \tag{18}$$

where **a** is a learnable weight vector, representing the attention mechanism a: a single-layer feedforward neural network, **W** is a learnable input linear transformation's weight matrix and $\|$ represents concatenation operation.

Each K multi-head attention GATv2 layer updates the node features as follows:

$$\mathbf{h}_i^{(l+1)} = \overset{K}{\underset{k=1}{\Big\|}} \sigma\left(\sum_{j\in\mathcal{N}_i} \alpha_{ij}^k \mathbf{W}^k \mathbf{h}_j^{(l)}\right) \tag{19}$$

where $\|$ represents concatenation, α_{ij}^k are normalized dynamic attention coefficients computed by the k-th attention mechanism (a^k), and $\mathbf{W}^k$ is the corresponding input linear transformation's weight matrix. It has been proved that *GATv2 – a graph attention variant that has a universal approximator attention function, and is thus strictly more expressive than GAT* (proof in [6]).

D Appendix: Experiment Details

Implementation of GNN incorporating edge features. As illustrated in Equation 2, a general MPNN can aggregate both node and edge embeddings to update the node embedding of the targeted node. In order to incorporate edge embeddings in GNNs, we use a single-layer MLP $f : \mathbb{R}^{d_e} \to \mathbb{R}^d$ to project relevant edge features of the targeted node to the same dimension of node features and then concatenate them to node features during the neighbourhood aggregation step in GNNs.

In GCPN–e, we use GNNs without incorporating edge features as the representational module in GCPN, $i.e.$ there is no edge feature aggregated during the neighbourhood aggregation step in the message-passing mechanism. The general GNN without considering edge features iteratively updates the node features $\mathbf{h}_i^{(l)} \in \mathbb{R}^d$ from layer l to layer $l+1$ via propagating messages through neighbouring nodes $j \in \mathcal{N}_i$, which can be formalised by the following equation:

$$\mathbf{h}_i^{(l+1)} = \text{UPDATE}\left(\mathbf{h}_i^{(l)}, \bigoplus_{j \in \mathcal{N}_i} \text{MESSAGE}\left(\mathbf{h}_i^{(l)}, \mathbf{h}_j^{(l)}\right)\right) \tag{20}$$

where MESSAGE and UPDATE are learnable functions, such as Multi-Layer Perceptrons (MLPs), $\mathcal{N}_i = \{j | (i,j) \in \mathcal{E}\}$ is the (1-hop) neighbourhood of node i, and $\bigoplus$ is a permutation-invariant local neighbourhood aggregation function, such as sum, mean or max.

In the original GraphAF, the intermediate molecular graph generated by the autoregressive flow doesn't contain edge features. Thus, in GraphAF+e, during the graph generation step, we apply a one-hot encoding to each edge according to the edge type to obtain edge features in the intermediate generated molecular graph.

Reward design implementation. For the property optimization task, we use the same reward design in GraphAF, which incorporates both intermediate and final rewards for training the policy network. *A small penalization will be introduced as the intermediate reward if the edge predictions violate the valency check. The final rewards include both the score of targeted-properties of generated molecules and the chemical validity reward. The final reward is distributed to all intermediate steps with a discounting factor to stabilize the training.* [4] The property-targeted reward r for a molecule m on a metric d is defined as follows:

$$r(m) = \exp\left(\frac{d(m)}{t}\right) \tag{21}$$

where t is the temperature for reward design decided by the grid search.

Hyper-parameter tuning. All the pre-training works are trained with an Adam optimizer with a learning rate of 0.001. For GCPN, we fixed batch sizes for pre-training and fine-tuning as 128 and 32. For GraphAF, we fixed batch sizes for pre-training and fine-tuning as 32 and 32. Each reward scale factor for Penalised logP, QED and SA is set as 1 after a manual grid search in the space $\{1, 5, 10\}$ based on evaluating each generative metric based on the performance of the original GCPN and GraphAF. The reward scale factors for DRD2, Median1 and Median2 are set as 0.5, 0.05 and 0.05 respectively after a manual grid search in the space $\{0.0001, 0.001, 0.05, 0.1, 0.5\}$ based on evaluating each generative metric based on the performance of original GCPN and GraphAF. We use Optuna to conduct a parallelized hyper-parameter grid search to determine the optimal hyper-parameters: the number of neurons in each hidden layer in the search space $\{64, 128, 256\}$ and the learning rate for fine-tuning in the search space $\{0.001, 0.0001, 0.00001, 0.000001\}$. The number of neurons in each hidden layer and the learning rate for fine-tuning for each generative task are summarised below in Table 9 and Table 10.

Table 9: Detailed setup for GCPN variants on Penalised logP, QED and SA.

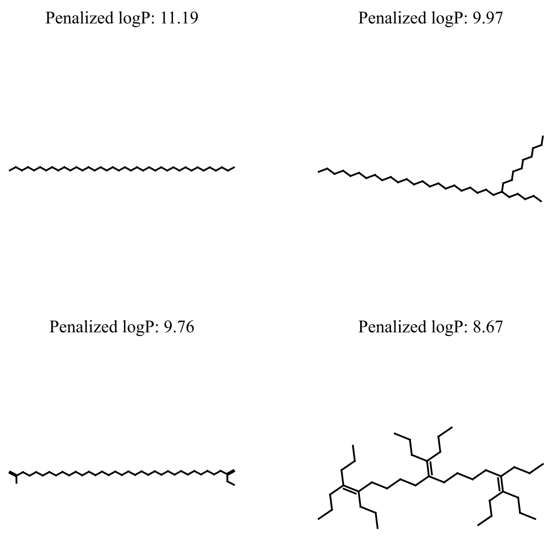
Model	Penalised logP	QED	SA
GCPN (R-GCN)	256, 0.00001	256, 0.00001	128, 0.001
GCPN (GIN)	256, 0.0001	256, 0.00001	256, 0.0001
GCPN (GAT)	128, 0.0001	256, 0.00001	64, 0.001
GCPN (GATv2)	256, 0.0001	256, 0.00001	256, 0.0001
GCPN (PNA)	256, 0.00001	64, 0.001	256, 0.0001
GCPN (GSN)	256, 0.0001	64, 0.001	256, 0.0001
GCPN (GearNet)	256, 0.0001	256, 0.00001	128, 0.001
GraphAF (R-GCN)	64, 0.0001	256, 0.00001	128, 0.0001
GraphAF (GearNet)	64, 0.0001	256, 0.000001	128, 0.0001

Table 10: Detailed setup for GCPN variants and GraphAF variants on DRD2, Median1 and Median2.

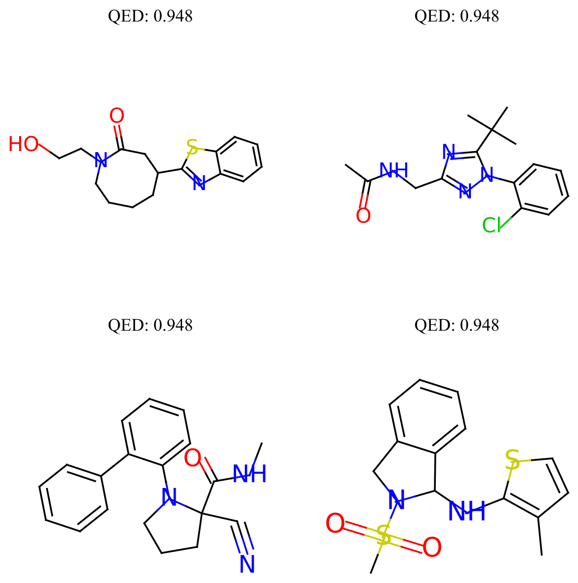
Model	DRD2	Median1	Median2
GCPN (R-GCN)	256, 0.0001	128, 0.0001	256, 0.00001
GCPN (GIN)	256, 0.00001	256, 0.00001	256, 0.00001
GCPN (GAT)	256, 0.00001	64, 0.001	256, 0.00001
GCPN (GATv2)	128, 0.0001	128, 0.001	256, 0.00001
GCPN (PNA)	64, 0.001	256, 0.0001	256, 0.00001
GCPN (GSN)	256, 0.00001	64, 0.001	256, 0.00001
GCPN (GearNet)	256, 0.00001	256, 0.0001	256, 0.00001
GraphAF (R-GCN)	256, 0.000001	256, 0.00001	256, 0.000001
GraphAF (GIN)	256, 0.000001	256, 0.000001	256, 0.000001
GraphAF (GAT)	256, 0.000001	256, 0.00001	256, 0.00001
GraphAF (GATv2)	256, 0.000001	256, 0.000001	256, 0.000001
GraphAF (PNA)	256, 0.000001	256, 0.000001	256, 0.000001
GraphAF (GSN)	256, 0.0001	256, 0.000001	256, 0.000001
GraphAF (GearNet)	256, 0.00001	256, 0.00001	256, 0.000001

E Appendix: Visualization of Generated Molecule Graphs

We present visualizations of generated molecules with the highest score on metric Penalized logP and QED. The visualizations illustrate that generated molecules with the highest Penalized logP score only contain a long chain of carbons.



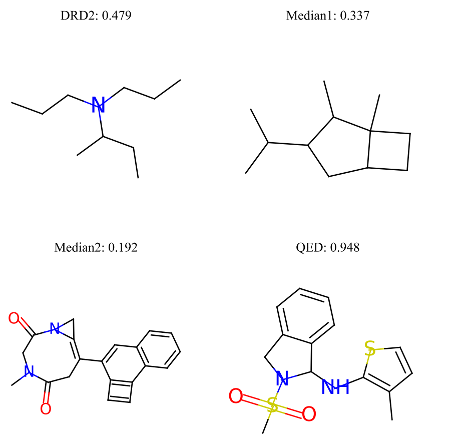
(a) Generated molecules with the highest Penalized logP scores



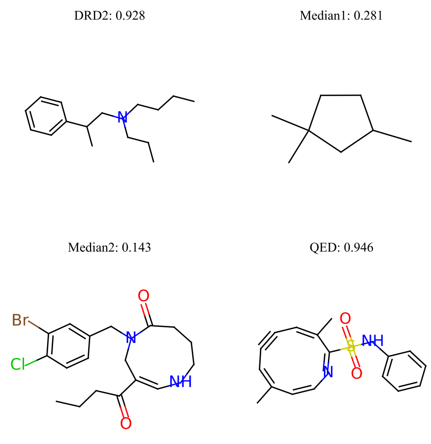
(b) Generated molecules with the highest QED scores

Figure 3: Molecules with highest generation metrics: Penalized logP and QED generated by GNN-based graph generative models on de-novo molecule design tasks.

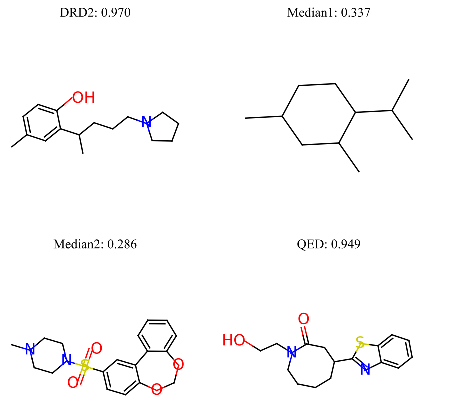
We present visualizations of generated molecules by GCPN, GraphAF and their optimal variants with GearNet in Figure 4(a), Figure 4(b), Figure 4(c) and Figure 4(d) respectively. The visualizations demonstrate that GCPN and GraphAF with advanced GNN have strong abilities to model different graph structures on the de-novo molecule design task.



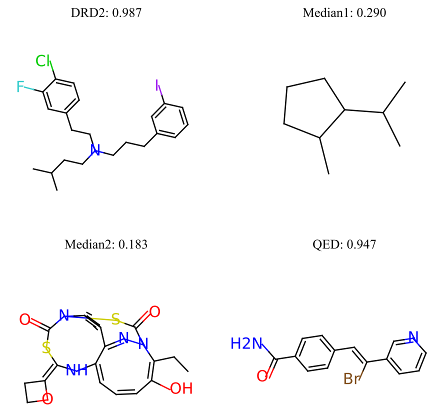
(a) Molecules with most desired generation metrics generated by GCPN (R-GCN)



(b) Molecules with most desired generation metrics generated by GraphAF (R-GCN)



(c) Molecules with most desired generation metrics generated by GCPN (GearNet)



(d) Molecules with most desired generation metrics generated by GraphAF (GearNet)

Figure 4: Molecules with highest generation metrics: DRD2, Median1, Median2 and QED generated by proposed GNN-based graph generative models: (a) GCPN with R-GCN (b) GraphAF with R-GCN (c) GCPN with GearNet (d) GraphAF with GearNet on de-novo molecule design tasks.