Improving Graph Generative Models via Expressive Graph Neural Networks

Xiangyu Zhao
Trinity College
xz398@cam.ac.uk

Abstract

Graph generation is a very challenging problem that requires predicting an entire graph with multiple nodes and edges from a given label, and is fundamental for many real-world tasks, such as molecular graph generation for drug discovery. A lot of successful methods have been explored on graph generation, including Graph Convolutional Policy Network (GCPN) and GraphAF, but the underlying graph neural network (GNN) structure for graph representation within both works remains untouched, which is Relational Graph Convolutional Network (R-GCN). In this mini-project, I investigate the expressivity of GNNs under the context of the graph generation problem, by replacing R-GCN in GCPN with more expressive GNNs, including Graph Isomorphism Network (GIN), Principal Neighbourhood Aggregation (PNA) and Graph Substructure Network (GSN). Experimental results show that more expressive GNNs can indeed significantly improve GCPN's performance on chemical property optimisation, with the only bottleneck coming from the sensitive nature of the graph generative method. [1]

1 Introduction

Graph generation is a very challenging problem that requires predicting an entire graph with multiple nodes and edges from a given label, and is fundamental for many real-world tasks, such as molecular graph generation for drug discovery. Similar to other generative models, a standard graph generative model consists of two parts: a representation module that learns graph embeddings from a pre-training graph space, and a generative module that decodes graph embeddings into a fine-tuning graph space. Recently, there have been significant progress in molecular graph generation with deep generative models, such as Graph Convolutional Policy Network (GCPN) [1] and [2], but they both use the Relational Graph Convolutional Network (R-GCN) [3], which was the state-of-the-art at the time of GCPN, as its inner graph representation model. However, graph neural networks (GNNs) has achieved remarkable success in graph classifications and graph regressions over the recent years, and it is of great research interest to find out whether more expressive GNNs that perform better in graph classification/regression tasks can also perform better in graph generation tasks.

In this mini-project, I investigate the expressivity of GNNs under the context of the graph generation problem, by replacing R-GCN in GCPN with more expressive GNNs, including Graph Isomorphism Network (GIN), Principal Neighbourhood Aggregation (PNA) and Graph Substructure Network (GSN), and compare their performance on the chemical property optimisation tasks. I begin this report by describing the GNNs investigated in my experiments, followed by an analysis of the results.

[1] Source code is available at https://github.com/VictorZXY/expressive-graph-gen

2 Theoretical Background

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ define a graph, where $\mathcal{V}$ denotes the set of nodes, and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ denotes the set of edges. Two graphs $\mathcal{G}_A = (\mathcal{V}_A, \mathcal{E}_A)$, $\mathcal{G}_B = (\mathcal{V}_B, \mathcal{E}_B)$ are isomorphic, denoted $\mathcal{G}_A \cong \mathcal{G}_B$, if and only if there exists an adjacency-preserving bijective mapping $f : \mathcal{V}_A \to \mathcal{V}_B$ between them, i.e.,

$$\forall i, j \in \mathcal{V}_A \,.\, (i,j) \in \mathcal{E}_A \iff \big(f(i), f(j)\big) \in \mathcal{E}_B \tag{1}$$

A molecular graph can be represented by a tuple of features $(\mathbf{A}, \mathbf{H}, \mathbf{E})$, where

- $\mathbf{A} \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}|}$ 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 $\{0,1\}^{|\mathcal{V}| \times |\mathcal{V}|}$ adjacency matrix format, since there are different types of bonds (i.e., single, double, triple, aromatic) in a molecule.
- $\mathbf{H} \in \mathbb{R}^{|\mathcal{V}| \times d}$ is the node feature matrix, with each row $\mathbf{h}_i \in \mathbb{R}^d$ being the d-dimensional features of node i.
- $\mathbf{E} \in \mathbb{R}^{|\mathcal{E}| \times d_e}$ is the edge feature matrix, with each row $\mathbf{e}_{ij} \in \mathbb{R}^{d_e}$ being the d_e-dimensional features of edge (i,j).

2.1 Graph neural networks

All GNNs investigated in this mini-project can be abstracted as Message Passing Neural Networks (MPNNs) [4]. 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, 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 L layers of message passing, the graph embedding $\mathbf{h}_\mathcal{G} \in \mathbb{R}^d$ can be obtained via a readout function:

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

The expressivity of a GNN is defined as the ability to distinguish non-isomorphic graphs, and can be analysed by comparing to the Weisfeiler-Lehman (1-WL) graph isomorphism test [5]. Similar to GNNs, the 1-WL test iteratively updates the node embeddings of a graph by neighbourhood aggregation: for each node $i \in \mathcal{V}$ 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* (proof in [6]).

Relational Graph Convolutional Network (R-GCN) The graph convolution operation of the original GCN [7] 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 constant for each pair of i and j (exact definition not relevant for this mini-project), $\mathbf{W} \in \mathbb{R}^{d\times d}$ is a learnable weight matrix, and σ is a non-linear activation function. R-GCN [3] makes use of the relational data of the graphs, and extends the graph convolution operation to the following: let $\mathcal{R}$ be the 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 $r \in \mathcal{R}$, and $c_{i,r}$ is a problem-specific normalisation constant that can either be learnt or chosen in advance. It has been shown that *neither GCN nor R-GCN is as expressive as the 1-WL test* [6].

Graph Isomorphism Network (GIN) Each GIN [6] 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 ϕ 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 [6]).

Principal Neighbourhood Aggregation (PNA) The PNA [8] 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:

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

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 ψ, ϕ are MLPs. According to the theorem that *in order to discriminate between multisets of size n whose underlying set is $\mathbb{R}$, at least n aggregators are needed* (proof in [8]), 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 [9] adopts a feature-augmented message passing style by counting the appearance of certain graph substructures and encoding them into the features. Firstly, it is necessary to define what is a subgraph: a graph $\mathcal{G}' = (\mathcal{V}', \mathcal{E}')$ is a subgraph of $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, denoted $\mathcal{G}' \sqsubseteq \mathcal{G}$, if and only if $\mathcal{V}' \subseteq \mathcal{V}$ and $\mathcal{E}' \subseteq \mathcal{E}$. The feature augmentation of GSN then works as follows: let $\mathfrak{G} = \{\mathcal{G}_1, \cdots, \mathcal{G}_K\}$ be a set of pre-computed small (connected) graphs. For each

Model	MAE (no edge features)	MAE (edge features)
GCN	0.469±0.002	—
GIN	0.408±0.008	—
PNA	0.320±0.032	0.188±0.004
GSN	0.140±0.006	0.115±0.012

Table 1: Molecular graph regression results of the above-mentioned GNNs on the ZINC dataset. Results of GCN and GIN are obtained in [11], PNA in [8], and GSN in [9]. R-GCN is not evaluated on the ZINC dataset in its original paper [3], nor is it analysed in [11], and is therefore not listed.

$\mathcal{G}_k \in \mathfrak{G}$, we first find its isomorphic subgraphs $\mathcal{G}'_k$ in $\mathcal{G}$. Then, for each node $i \in \mathcal{V}$ and $1 \leq k \leq K$, we count the number of subgraphs $\mathcal{G}'_k$ node i belongs to, as defined by the below equation:

$$x^{\mathcal{V}}_{\mathcal{G}_k}(i) = \left| \left\{ \mathcal{G}'_k \cong \mathcal{G}_k \middle| i \in \mathcal{V}'_k \right\} \right| \tag{10}$$

We then obtain the node structural features $\mathbf{x}^{\mathcal{V}}_i = \begin{bmatrix} x^{\mathcal{V}}_{\mathcal{G}_1}(i) & \cdots & x^{\mathcal{V}}_{\mathcal{G}_K}(i) \end{bmatrix} \in \mathbb{N}^K$ of node i. Similarly, we can define the edge structural features $\mathbf{x}^{\mathcal{E}}_{ij} = \begin{bmatrix} x^{\mathcal{E}}_{\mathcal{G}_1}(i,j) & \cdots & x^{\mathcal{E}}_{\mathcal{G}_K}(i,j) \end{bmatrix} \in \mathbb{N}^K$ of each edge $(i,j) \in \mathcal{E}$ by counting the numbers of subgraphs it belongs to:

$$x^{\mathcal{E}}_{\mathcal{G}_k}(i,j) = \left| \left\{ \mathcal{G}'_k \cong \mathcal{G}_k \middle| (i,j) \in \mathcal{E}'_k \right\} \right| \tag{11}$$

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

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

where ψ, ϕ are MLPs. It can be proved that *GSN is strictly more expressive than the 1-WL test, when $\mathcal{G}_k$ 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 [9]). This essentially suggests that GSN is more expressive than GCN, R-GCN, GIN and PNA. Note that this does not violate the previous theorem that GNNs are at most as expressive as the 1-WL test, as here GSN is only conditionally more expressive than the 1-WL test.

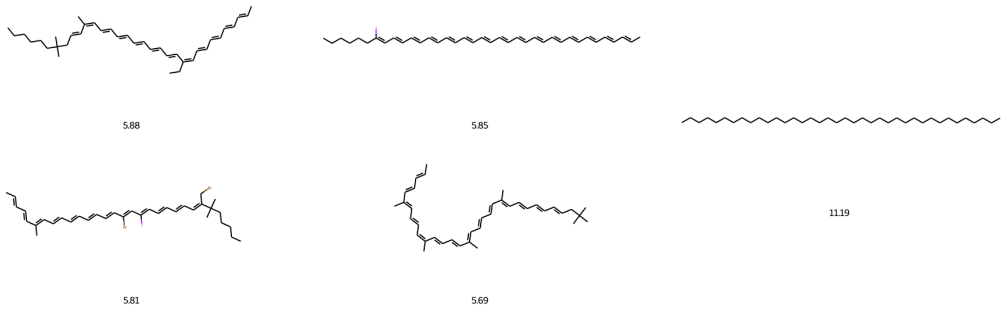
The expressivity rankings of the above-mentioned GNNs are also backed by the molecular graph regression benchmark on the ZINC dataset [10] proposed in [11], as shown in Table 1.

2.2 Molecular graph generation

In this mini-project, GCPN [1] is used as the graph generative model for the experiments. GCPN is based on reinforcement learning (RL) of a generative adversarial network (GAN). It is designed as an RL agent that operates within a chemistry-aware graph generation environment. A molecule is successively constructed by either connecting a new substructure, or an atom with an existing molecular graph, or adding a bond to connect existing atoms. GCPN predicts the action of the bond addition, and is trained via the policy gradient method to optimize a reward composed of molecular property objectives and adversarial loss. The adversarial loss is provided by a GNN-based discriminator trained jointly on a dataset of example molecules. Since this mini-project focuses mainly on the inner GNNs of a graph generative model, rather than the graph generative method itself, the detailed GCPN algorithm is not required.

Model	Penalised logP			QED			Fine-tuning epochs (QED)
	1st	2nd	3rd	1st	2nd	3rd	
ZINC	4.52	4.30	4.23	0.948	0.948	0.948	—
R-GCN	5.88	5.85	5.81	0.948	0.948	0.948	10
GIN	11.19	11.19	11.19	0.824	0.823	0.806	2
PNA	11.19	11.19	11.19	0.751	0.728	0.728	2
GSN	11.19	11.19	11.19	0.934	0.933	0.933	2

Table 2: Comparison of the top-3 property scores of the generated molecules by GCPN with different GNNs, with early stopping.



(a) R-GCN (b) GIN, PNA, and GSN

Figure 1: Molecules generated with the top penalised logP scores by GCPN with different GNNs.

3 Experiments

3.1 Experimental setup

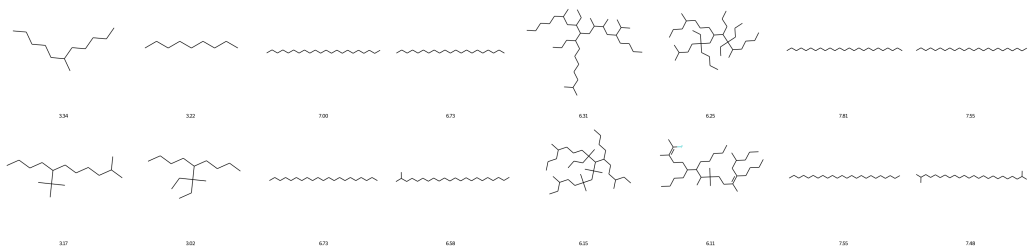
In this mini-project, I investigate the expressivity of the above-mentioned GNNs under the context of molecular graph generation, and conduct experiments by replacing R-GCN in GCPN with GIN, PNA and GSN, then compare their performances in the property optimisation task. This task is aimed at generating novel and valid molecules whose specified molecular properties are optimised. This can be very useful in drug discovery, whose goal is to find molecules with highly optimised properties of interest. I choose the penalised octanol-water partition coefficient (penalised logP), which measures the solubility and synthetic accessibility of a molecule, and the quantitative estimate of drug-likeness (QED) score, as my target property. I use the ZINC dataset [10] for both pre-training and fine-tuning for this task. ZINC contains 250,000 drug-like molecules with 9 atom types and 3 edge types, and with a maximum graph size of 38. All molecules are presented in the kekulised form with hydrogen removed.

The environment of this mini-project is implemented based on the newly developed TorchDrug library [12]. Both R-GCN and GIN have been provided by TorchDrug, and I implemented PNA and GSN on my own, by consulting their original code repositories [8, 9], and adapting them to fit into the TorchDrug interface. In order to compare the GNNs fairly, I set all GNNs in the experiments to have the same hyperparameters: all GNNs have 3 hidden layers, with batch normalisation and ReLU activation applied after each layer. All GNNs use summation as the READOUT function. For PNA and GSN, the MESSAGE and UPDATE functions are parameterised by single-layer perceptrons. For GSN, I 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.

The GCPN models, with all different GNNs, are pre-trained on the ZINC dataset for 10 epochs with a batch size of 128. They are then fine-tuned towards the target properties with RL, using the proximal policy optimisation (PPO) algorithm. It has been observed during the experiments that RL fine-tuning

Model	Penalised logP			Approx. no. of batches required for convergence
	1st	2nd	3rd	
ZINC	4.52	4.30	4.23	—
R-GCN	3.34	3.22	3.17	—
GIN	7.00	6.73	6.73	255
PNA	6.31	6.25	6.15	300
GSN	**7.81**	**7.55**	**7.55**	260

Table 3: Comparison of the top-3 property scores of the generated molecules by GCPN with different GNNs.



(a) R-GCN (b) GIN (c) PNA (d) GSN

Figure 2: Molecules generated with the top penalised logP scores by GCPN with different GNNs at the early training stage.

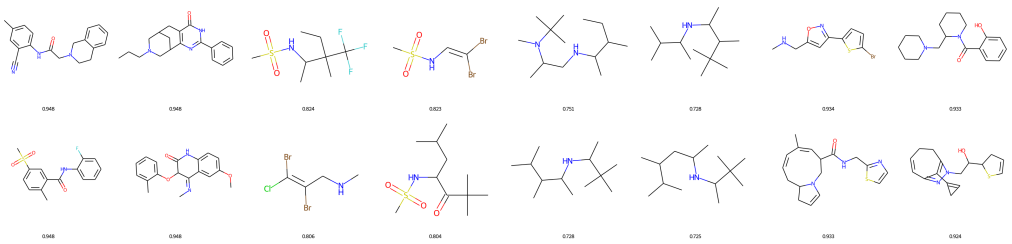
takes much longer than non-RL pre-training, and QED optimisation is much slower than penalised logP optimisation, the models are fine-tuned for 5 epochs for penalised logP optimisation, and 2 epochs for QED optimisation, in order to save the computing resources, except for R-GCN, which is lightweight enough to be fully fine-tuned for 10 epochs on both targets. All fine-tuning experiments are run with a batch size of 64. Adam is used as the optimiser for both pre-training and fine-tuning, with learning rates of 10^{-3} for pre-training and 10^{-5} for fine-tuning. These hyperparameters are obtained from the TorchDrug molecule generation tutorials, with slight adaptations in order to reduce the demand for computing resources. The experiments are run on an NVIDIA A100 GPU with 80GB memory.

3.2 Results

The training results of the GCPN models with different GNNs, evaluated as the top-3 property scores of molecules generated by each model, are summarised in Table 2, with the top-3 property scores of molecules in the ZINC dataset for reference.

Penalised logP optimisation The results on Table 2 and the visualisation of the generated molecules on Figure 1 clearly shows that GIN, PNA, GSN can all perform better than R-GCN, but they all converge to the same molecule consists of a long chain of carbon. This suggests that all 3 models have overfitted the GCPN's RL policy network, making it overly favours the action that attaches a carbon to the carbon chain in order to maximise the penalised logP score, with other actions having probabilities close to zero. This is a common phenomenon of RL training, where one action dominates the action space, and can be caused by the fact that RL and GANs are both sensitive to hyperparameters. Since all GNNs share the same hyperparameters with R-GCN, this may cause the other GNNs to stuck in one particular action whereas R-GCN stays alright, as this set of hyperparameters is already validated for R-GCN by both the original GCPN paper [1] and the TorchDrug tutorial.

However, this does not mean we cannot compare between GIN, PNA, and GSN: a closer look at the training results in Table 3 show that all 3 models can quickly converge to the maximum score, but GSN and GIN do converge faster than PNA according to the number of batches they take to reach convergence. An early-stopping record can also show that GSN and GIN can perform better



(a) R-GCN (b) GIN (c) PNA (d) GSN

Figure 3: Molecules generated with the top QED scores by GCPN with different GNNs.

than PNA at the early training stage, but PNA is able to produce more complex molecules before overfitting, as shown in Figure 2.

QED optimisation Unfortunately, the results on Table 2 and the visualisation of the generated molecules on Figure 3 do not show that GIN, PNA and GSN perform better than R-GCN on the QED optimisation task. However, it is worth pointing out that GIN, PNA and GSN are only fine-tuned for 2 epochs, and still have the potential to outperform R-GCN if given enough time to fine-tune for a full 10 epochs.

4 Conclusion

In this mini-project, I investigate the expressivity of GNNs under the context of the graph generation problem, by replacing R-GCN in GCPN with more expressive GNNs, namely GIN, PNA and GSN, and compare their performance on the chemical property optimisation tasks. Results show that the more expressive GNNs (i.e., GIN, PNA and GSN) can indeed perform better in molecular graph generation, with the bottleneck coming from the sensitive nature of the graph generative method. In addition, since nearly all of the recent works on new GNN architectures are focused on pushing node/graph classification benchmarks, which are comparatively simpler than graph generation modelling in terms of the combinatorial complexity, This mini-project also hopes to challenge the graph representation learning community's notion for benchmarking the expressivity of GNNs.

5 Acknowledgements

I thank Chaitanya Joshi and Cristian Bodnar for supervising me on this mini-project, and Dr Dominique Beaini for helping me understanding the GSN. This work was performed using resources provided by the Cambridge Service for Data Driven Discovery (CSD3) operated by the University of Cambridge Research Computing Service (`www.csd3.cam.ac.uk`), provided by Dell EMC and Intel using Tier-2 funding from the Engineering and Physical Sciences Research Council (capital grant EP/T022159/1), and DiRAC funding from the Science and Technology Facilities Council (`www.dirac.ac.uk`).

References

[1] Jiaxuan You, Bowen Liu, Zhitao Ying, Vijay Pande, and Jure Leskovec. Graph convolutional policy network for goal-directed molecular graph generation. In *Advances in Neural Information Processing Systems (NeurIPS 2018)*, volume 31, pages 6410–6421. Curran Associates, Inc., 2018.

[2] Chence Shi, Minkai Xu, Zhaocheng Zhu, Weinan Zhang, Ming Zhang, and Jian Tang. GraphAF: a flow-based autoregressive model for molecular graph generation. In *8th International Conference on Learning Representations (ICLR 2020)*. OpenReview.net, 2020.

[3] Michael Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *European semantic web conference*, pages 593–607. Springer, 2018.

[4] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning (ICML 2017)*, volume 70, pages 1263–1272. PMLR, 2017.

[5] Boris Weisfeiler and Andrei Leman. A reduction of a graph to canonical form and an algebra arising during this reduction. *Nauchno-Technicheskaya Informatsia*, 2(9):12–16, 1968. English translation is available at `https://www.iti.zcu.cz/wl2018/pdf/wl_paper_translation.pdf`.

[6] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *7th International Conference on Learning Representations (ICLR 2019)*. OpenReview.net, 2019.

[7] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations (ICLR 2017)*. OpenReview.net, 2017.

[8] Gabriele Corso, Luca Cavalleri, Dominique Beaini, Pietro Liò, and Petar Veličković. Principal neighbourhood aggregation for graph nets. In *Advances in Neural Information Processing Systems (NeurIPS 2020)*, volume 33, pages 13260–13271. Curran Associates, Inc., 2020.

[9] Giorgos Bouritsas, Fabrizio Frasca, Stefanos P. Zafeiriou, and Michael Bronstein. Improving graph neural network expressivity via subgraph isomorphism counting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[10] John J. Irwin, Teague 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(7):1757–1768, 2012.

[11] Vijay P. Dwivedi, Chaitanya K. Joshi, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Benchmarking graph neural networks. *arXiv preprint arXiv:2003.00982*, 2020.

[12] Zhaocheng Zhu, Chence Shi, Zuobai Zhang, Shengchao Liu, Minghao Xu, Xinyu Yuan, Yangtian Zhang, Junkun Chen, Huiyu Cai, Jiarui Lu, et al. TorchDrug: A powerful and flexible machine learning platform for drug discovery. *arXiv preprint arXiv:2202.08320*, 2022.