MolHF: A Hierarchical Normalizing Flow for Molecular Graph Generation

Yiheng Zhu[1], **Zhenqiu Ouyang**[2], **Ben Liao**[3], **Jialu Wu**[4], **Yixuan Wu**[5], **Chang-Yu Hsieh**[4*], **Tingjun Hou**[4*] and **Jian Wu**[5,6*]
[1]College of Computer Science and Technology, Zhejiang University
[2]Polytechnic Institute, Zhejiang University
[3]Tencent Quantum Laboratory, Tencent
[4]College of Pharmaceutical Sciences, Zhejiang University
[5]School of Public Health, Zhejiang University
[6]Second Affiliated Hospital School of Medicine, Zhejiang University
{zhuyiheng2020, oyzq, jialuwu, wyx_chloe, kimhsieh, tingjunhou, wujian2000}@zju.edu.cn, liao@hotmail.co.uk

Abstract

Molecular *de novo* design is a critical yet challenging task in scientific fields, aiming to design novel molecular structures with desired property profiles. Significant progress has been made by resorting to generative models for graphs. However, limited attention is paid to hierarchical generative models, which can exploit the inherent hierarchical structure (with rich semantic information) of the molecular graphs and generate complex molecules of larger size that we shall demonstrate to be difficult for most existing models. The primary challenge to hierarchical generation is the non-differentiable issue caused by the generation of intermediate discrete coarsened graph structures. To sidestep this issue, we cast the tricky hierarchical generation problem over discrete spaces as the reverse process of hierarchical representation learning and propose MolHF, a new hierarchical flow-based model that generates molecular graphs in a coarse-to-fine manner. Specifically, MolHF first generates bonds through a multi-scale architecture, then generates atoms based on the coarsened graph structure at each scale. We demonstrate that MolHF achieves state-of-the-art performance in random generation and property optimization, implying its high capacity to model data distribution. Furthermore, MolHF is the first flow-based model that can be applied to model larger molecules (polymer) with more than 100 heavy atoms. The code and models are available at https://github.com/violet-sto/MolHF.

1 Introduction

Designing novel and valuable molecular structures is a critical yet challenging task with great application potential in drug and material discovery. Owing to the enormously vast chemical space with discrete molecular structures, whose scale was estimated to be on the order of 10^{33}[Polishchuk *et al.*, 2013], it is time-consuming and costly to search exhaustively in this space. Recently, there has been a surge of interest in developing generative models for graphs to resolve this issue by exploring the chemical space in a data-driven manner and sampling novel molecules [Jin *et al.*, 2018].

Much prior work represents molecules as attributed graphs and leverages a variety of deep generative frameworks for molecular graph generation, including variational autoencoders (VAEs) [Simonovsky and Komodakis, 2018], generative adversarial networks (GANs) [De Cao and Kipf, 2018], and normalizing flows [Shi *et al.*, 2020; Zang and Wang, 2020]. However, most of them ignore the inherent hierarchical structure (with rich semantic information) of the molecular graphs. The significance of exploiting hierarchical structure lies in the fact that some particular substructures, such as molecular fragments and functional groups, appear frequently in molecules and are likely to determine molecular properties. Thereby, these models are sub-optimal for molecular generation and property optimization. Additionally, molecular graph generation is prone to errors, e.g., randomly removing or adding a covalent bond probably yields an invalid molecular structure. Thus, whilst existing models have demonstrated promising performance for small molecules, they tend to perform dreadfully for large molecules or even fail to effectively generate more complex molecules of larger size [Jin *et al.*, 2020]. Recently, substructure-based methods [Jin *et al.*, 2020], where the substructures are manually extracted beforehand and leveraged as building blocks, have been proposed to exploit the implicit hierarchical structure. Nevertheless, the substructure extraction strategies are mostly based on heuristics and prior domain knowledge [Jin *et al.*, 2018; Jin *et al.*, 2020], and the extraction process is cumbersome (e.g., counting the frequency of occurrence of all possible substructures). These practical drawbacks limit the flexibility and scalability of such methods.

Advances in hierarchical generative frameworks have made significant progress in high-resolution image generation [Karras *et al.*, 2018; Yu *et al.*, 2020] and super-resolution [Liang *et al.*, 2021], with the merits of better qual-

*Corresponding authors.

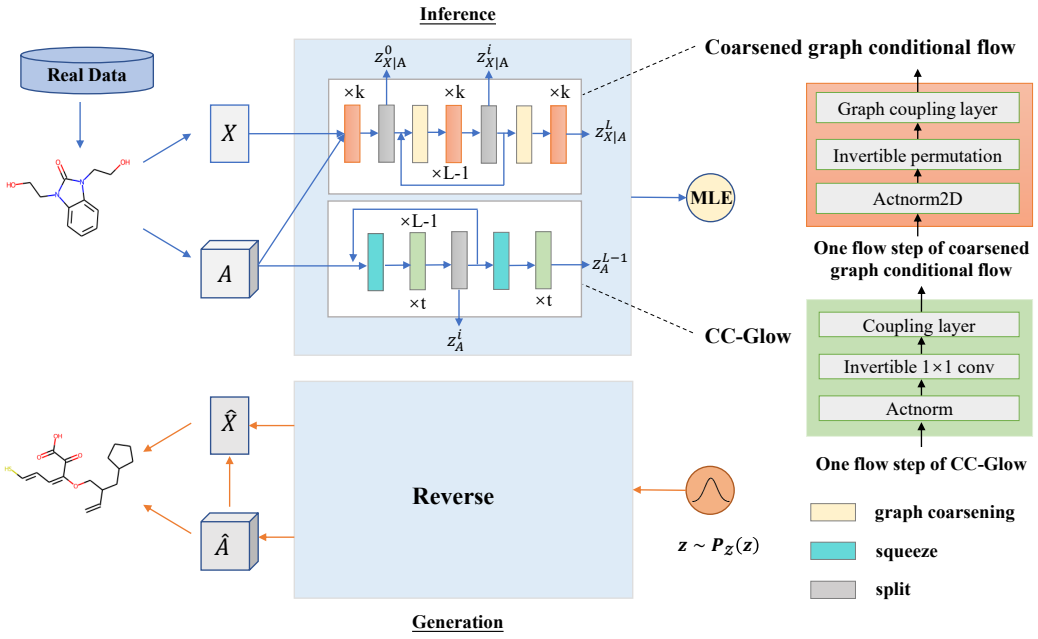

Figure 1: The framework of MolHF. A molecular graph G is represented with atom matrix X and bond tensor A. MolHF consists of two flows, namely a *coarsened graph conditional flow* $f_{X|A}$ for hierarchically transforming X given A, and a *CC-Glow* f_A for hierarchically transforming A. Right: the details of one flow step of $f_{X|A}$ and f_A.

ity, scalability, and stability. Their key motivation is to progressively generate higher resolution images based on lower resolution input. However, applying hierarchical generative frameworks to derive molecular graphs is non-trivial. The primary challenge unique to hierarchical graph generation is generating intermediate discrete coarsened graph structures, which leads to the inability to backpropagate through samples. Currently, hierarchical molecular graph generation is still in its infancy, and it remains unclear how exploiting the hierarchical structure of molecular graphs could enhance the performance of generative models.

To overcome the above issues, we cast the tricky problem of hierarchical generation over discrete spaces as the reverse process of hierarchical representation learning resorting to invertible normalizing flows [Dinh *et al.*, 2014; Dinh *et al.*, 2016], and propose a new hierarchical flow-based generative model called MolHF, which generates molecular graphs in a coarse-to-fine manner. By defining an invertible transformation to capture the hierarchical structural information in the inference (encoding) process, MolHF can also exploit it in the reverse generation (decoding) process. Specifically, we propose *CC-Glow*, a Criss-Cross network [Huang *et al.*, 2019] based multi-scale Glow [Kingma and Dhariwal, 2018] which offers a better relational inductive bias, to generate bonds at first, and a *coarsened graph conditional flow* to generate atoms by performing graph convolution [Kipf and Welling, 2017; Schlichtkrull *et al.*, 2018] on the pre-generated coarsened graph structure (bonds) of each scale. We evaluate our MolHF in aspects of random generation and property optimization on the standard ZINC250K dataset [Irwin *et al.*, 2012]. Further, we use the Polymer dataset [St. John *et al.*, 2019] to verify that MolHF is capable of modeling larger molecules that are impractical with prior flow-based methods. As demonstrated with extensive experiments, MolHF achieves state-of-the-art performance, indicating the effectiveness of the proposed hierarchical architecture. Finally, we analyze the generative mechanism of MolHF through hierarchical visualization. Our key contributions are summarized as follows.

- We propose MolHF for hierarchical molecular graph generation. MolHF is the first flow-based model that can exploit the hierarchical structural information.
- We cast the non-trivial hierarchical generation problem over discrete spaces as the reverse process of hierarchical representation learning to sidestep the non-differentiable issue.
- MolHF outperforms state-of-the-art baselines in random generation and property optimization. Meanwhile, experiments on the Polymer dataset show that MolHF is the first flow-based model that can model larger molecules with more than 100 heavy atoms.

2 Related Work

2.1 Normalizing Flows

Normalizing flows are a class of probabilistic generative models that aim to learn invertible and differentiable mappings between complex target distributions and simple prior distributions. Compared with the prevailing generative models, such as VAEs and GANs, normalizing flows possess the merits of exact latent variable inference and likelihood estimation, efficient sampling, and stable training [Kingma and Dhariwal, 2018]. Advances in normalizing flows have made

impressive progress in various applications, such as density estimation [Dinh *et al.*, 2016], image generation [Kingma and Dhariwal, 2018], and variational inference [Rezende and Mohamed, 2015]. Recently, normalizing flows also found their way into generating more valid and novel molecular graphs [Shi *et al.*, 2020]. Furthermore, much work has been devoted to improving normalizing flows, ranging from expressive power [Ho *et al.*, 2019] to numerical stability [Behrmann *et al.*, 2021]. For a more comprehensive review of normalizing flows, we refer the readers to [Papamakarios *et al.*, 2021].

2.2 Molecular Graph Generation

Over the past few years, there has been a surge in generating molecular graphs resorting to various deep generative frameworks, e.g., VAEs [Kingma and Welling, 2013], GANs [Goodfellow *et al.*, 2014], and autoregressive models. For example, JT-VAE [Jin *et al.*, 2018] first generates a junction tree that represents the scaffold over chemical substructures, then assembles them into a valid molecular graph. MolGAN [De Cao and Kipf, 2018] is the first GAN-based model but shows poor performance in generating unique molecules. GCPN [You *et al.*, 2018] formulates molecular graph generation as a sequential decision process generating the atoms and bonds alternately and leverages reinforcement learning to optimize the generated molecules.

There are also several flow-based models for molecular graph generation, which can be mainly grouped into two categories, i.e., autoregressive methods and one-shot methods. GraphAF [Shi *et al.*, 2020] and GraphDF [Luo *et al.*, 2021] adapt autoregressive flows [Papamakarios *et al.*, 2017] to generate atoms and bonds alternately in a sequential manner. GraphNVP [Madhawa *et al.*, 2019] and MoFlow [Zang and Wang, 2020] are representative one-shot methods, which generate atoms conditioned on the pre-generated bonds in a two-step manner. Our MolHF follows the latter approach to exploit the coarsened graph structure for hierarchical generation while ensuring reversibility. The most related to our method is MolGrow [Kuznetsov and Polykovskiy, 2021], which starts with a single node and splits every node into two recursively to generate the entire molecule. Our MolHF differs from MolGrow in two main aspects: 1) MolGrow transforms the atom matrix and bond tensor simultaneously, therefore the transformed bond tensor becomes continuous hidden states which cannot provide structural information for modeling atoms in the coarsened scale. In contrast, MolHF is able to perform graph convolution to capture the hierarchical structural information, resorting to the aforementioned two-step approach. 2) MolGrow is less flexible and inefficient as the number of atoms in the training molecular graphs must be padded to be a power of 2, leading to extra computational costs. In contrast, we propose a graph coarsening operation to merge any number of nodes and compute the coarsened graph structure, making MolHF adaptable to various molecular datasets by setting appropriate coarsening ratios.

3 Problem Statement and Background

3.1 Hierarchical Molecular Graph Generation

The hierarchical molecular graph generation problem seeks to design a network (e.g., the decoder of VAE or generator of GAN) to generate the entire molecular graph in a coarse-to-fine manner. Put differently, the molecular graph is refined with increasingly fine details, conditioned on the coarser structure. Unfortunately, applying hierarchical generative frameworks to derive molecular graphs is non-trivial, since graphs are discrete objects. The primary challenge is generating intermediate discrete coarsened graph structures, which leads to the inability to backpropagate through samples. To sidestep this non-differentiable issue, we resort to normalizing flows [Dinh *et al.*, 2014] which construct a generative model by defining an invertible transformation. From the perspective of normalizing flows, the challenging problem of hierarchical molecular graph generation can be regarded as the reverse process of hierarchical representation learning.

For a molecular graph $G = (X, A)$ with n atoms, we represent it with an atom matrix $X \in \{0,1\}^{n\times d}$ and a bond tensor $A \in \{0,1\}^{b\times n\times n}$, where d and b are the number of different types of atoms and bonds, respectively. The element $X_{i,:}$ and $A_{:,i,j}$ are one-hot vectors indicating the type of the i^{th} atom and the type of the bond between the i^{th} atom and j^{th} atom, respectively. Our ultimate purpose is to learn the distribution $P(G)$ via a hierarchical probabilistic generative model.

3.2 Normalizing Flows

The flow-based generative models define a parameterized invertible transformation $f_\theta : \mathcal{X} \to \mathcal{Z}$ from observational data $x \sim P_\mathcal{X}(x)$ to latent variables $z \sim P_\mathcal{Z}(z)$, where $P_\mathcal{X}(x)$ is a complex target distribution and $P_\mathcal{Z}(z)$ is a simple base distribution (e.g., standard isotropic Gaussian distribution). Under the change-of-variables formula, the exact log-likelihood of x is tractable:

$$\log P_\mathcal{X}(x) = \log P_\mathcal{Z}(f_\theta(x)) + \log \left| \det \frac{\partial f_\theta(x)}{\partial x} \right| \quad (1)$$

where $\det \frac{\partial f_\theta(x)}{\partial x}$ is the Jacobian determinant of f_θ at x. Thereby, unlike VAEs, the flow-based models can be trained directly with the maximum likelihood estimation. As for generation, a sample is efficiently generated via the inverse transformation $x = f_\theta^{-1}(z)$ where z is sampled from $P_\mathcal{Z}(z)$.

In practice, $f_\theta = f_1 \circ \cdots \circ f_L$ is composed of a sequence of invertible functions whose Jacobian determinant is convenient to calculate. Herein, we consider using the affine coupling layer, actnorm, and invertible permutation.

Affine coupling layer. One of the most widely used invertible functions is the affine coupling layer [Dinh *et al.*, 2016]. Formally, given $x \in \mathbb{R}^D$ and $d < D$, the output $y \in \mathbb{R}^D$ can be computed as follows:

$$y_{1:d} = x_{1:d} \quad (2)$$

$$y_{d+1:D} = x_{d+1:D} \odot r(s(x_{1:d})) + t(x_{1:d}) \quad (3)$$

where $\odot$ stands for Hadamard product, s and t are scale and translation functions, respectively, and r is a rescale function that is commonly instantiated as an exponential function.

Since the log-determinant of the triangular Jacobian of this transformation is efficiently computed as $\sum_j \log r(s(x_{1:d}))_j$, s and t can be arbitrarily complex functions to improve the expressive power [Dinh *et al.*, 2016]. To mitigate numerical instability, we adopt the sigmoid function following the Swish activation function [Ramachandran *et al.*, 2017] as r with a restricted range, whose effect was thoroughly investigated in [Behrmann *et al.*, 2021].

Actnorm. Actnorm [Kingma and Dhariwal, 2018] is proposed to not only alleviate the problems associated with the numerical instability but also sidestep the noise introduced by the batch normalization. The main idea of actnorm is to normalize the activations to zero mean and unit variance before starting training. Specifically, the parameters of actnorm are initialized with the per-channel mean and standard deviation of the first batch of data, and optimized as regular parameters. As for the feature matrix, actnorm is extended to normalize the feature dimension. For consistency, we continue to call this variant actnorm2D [Zang and Wang, 2020].

Invertible permutation. Invertible 1×1 convolution with weight matrix $W \in \mathbb{R}^{c \times c}$ is a learnable permutation proposed to replace the fixed random permutation [Kingma and Dhariwal, 2018]. With 1×1 convolution, all channels are allowed to influence one another, and the expressive power of the following affine coupling layers is subsequently improved. Different from 3D tensor (images), the feature matrix can be permuted by directly multiplying W. To reduce the cost of computing log-determinant of W, we formulate it with LU decomposition [Kingma and Dhariwal, 2018].

4 Methods

In this section, we first introduce the overview of our proposed MolHF, then we elucidate the specifics of MolHF. Lastly, we describe the generation process of MolHF.

4.1 Overview of MolHF

Similar to existing one-shot flow-based models such as GraphNVP [Madhawa *et al.*, 2019] and MoFlow [Zang and Wang, 2020], MolHF aims to generate molecular graphs in one pass through the model, but in a coarse-to-fine manner. To exploit the coarsened graph structure for hierarchical generation while ensuring the reversibility of MolHF, we conditionally factorize the distribution $P(G)$ according to the combinatorial structure of molecular graphs, giving

$$P(G) = P(X, A) = P(A)P(X|A) \tag{4}$$

Based on this factorization, MolHF wishes to hierarchically transform the atom matrix X and bond tensor A respectively. As illustrated in Figure 1, MolHF consists of two flows, namely a *coarsened graph conditional flow* $f_{X|A}$ for atoms and a *CC-Glow* f_A for bonds. We adopt the well-established dequantization technique [Shi *et al.*, 2020; Zang and Wang, 2020] to convert both X and A into continuous data by adding uniform noise, since discrete data do not fit into the change-of-variables formula. Under the maximum likelihood principle, the parameters of MolHF are updated by gradient descent to minimize the negative log-likelihoods $\log P(G)$ over all training molecular graphs:

$$\arg\min_{f_A, f_{X|A}} \mathbb{E}_{X,A \sim P_{data}}[-\log P(A; f_A) - \log P(X|A; f_{X|A})] \tag{5}$$

4.2 Coarsened Graph Conditional Flow

Conditioning on the bond tensor A, *coarsened graph conditional flow* $f_{X|A}$ aims to learn $P(X|A)$ by transforming the atom matrix X into latent variables $z_{X|A} = f_{X|A}(X, A)$ in a hierarchical manner. Inspired by this motivation, we propose a graph coarsening operation and corresponding coarsened graph coupling layers to extend the multi-scale architecture proposed in [Dinh *et al.*, 2016].

Multi-scale architecture. At each scale, we perform three successive operations. Firstly, a graph coarsening operation is applied to a fine-grained molecular graph, except at the finest scale. Secondly, several steps of transformation which contains actnorm2D followed by an invertible permutation, followed by a coarsened graph coupling layer, are utilized to extract the coarse-grained features. Thirdly, half the features are split into latent variables, except at the coarsest scale. This procedure can be defined recursively,

$$h^1, z^0_{X|A}, A^0 = f^1_{X|A}(h^0, A^0) \tag{6}$$

$$h^{i+1}, z^i_{X|A}, A^i = f^{i+1}_{X|A}(h^i, A^{i-1}) \tag{7}$$

$$z^L_{X|A}, A^L = f^{L+1}_{X|A}(h^L, A^{L-1}) \tag{8}$$

$$z_{X|A} = (z^0_{X|A}, \ldots, z^L_{X|A}) \tag{9}$$

where $h^0 = X$ and $A^0 = A$. Note that the correlations between latent variables of different scales are modeled by assuming $z^i_{X|A} \sim \mathcal{N}(\mu_i(h^{i+1}), \sum_i(h^{i+1}))$.

Graph coarsening operation. We merge every k_{i-1} nodes to construct the coarsened graph, where k_{i-1} is a hyperparameter controlling the degree of coarsening at each scale. The merged node features are obtained with concatenation, while the coarsened graph structure A^i is computed as:

$$A^i_{m,:,:} = S^T A^{i-1}_{m,:,:} S, \quad m = 1, \ldots, b \tag{10}$$

where $S \in \{0, 1\}^{n_{i-1} \times \frac{n_{i-1}}{k_{i-1}}}$ is a predefined hard assignment matrix assigning every k_{i-1} nodes as a cluster in order, n_{i-1} is the number of nodes in the coarsened graph at the $i - 1^{th}$ scale, and A^i indicates the connectivity between each pair of clusters [Ying *et al.*, 2018]. As a notorious problem in graph generation, it is tough to define the most suitable atom ordering. We use breadth-first search ordering to preserve spatial locality, which is vital for hierarchical modeling.

Coarsened graph coupling layers. We transform the atom matrix via the affine coupling layers. To achieve permutation equivariance at each scale, we split atom features into two parts along the feature (column) dimension and adopt R-GCNs [Schlichtkrull *et al.*, 2018] in the coupling networks, leveraging the coarsened graph structure. The convolution operation of R-GCN can be written as follows:

$$H^l = \sum_{i=1}^{b} D^{-1} A_i H^{l-1} W_i + H^{l-1} W_0 \tag{11}$$

where H^l is the node embeddings in l^{th} layer, A_i is the i^{th} slice of adjacency tensor, $D_{jj} = \sum_{b,i} A_{b,i,j}$, and $\{W\}_{i=0}^{b}$ are weight matrices. Compared with the graph coupling layers used in MoFlow [Zang and Wang, 2020] that split atom features along the row dimension, ours are more flexible since the number of coupling layers is decoupled from the number of nodes.

4.3 CC-Glow

CC-Glow f_A aims to learn $P(A)$ by transforming bond tensor A into latent variables $z_A = f_A(A)$ in analogy with multi-channel images. We adopt the architecture of multi-scale Glow but propose a novel Criss-Cross Attention (CCA) [Huang *et al.*, 2019] based coupling layer to offer a better relational inductive bias for modeling bonds.

CCA-based coupling layers. For modeling bonds, it is essential to utilize the local information of the incident atoms. However, the atom matrix is unavailable at this stage in ensuring the reversibility of MolHF. Therefore, we decide to assign atoms with features aggregated from relevant bonds (attached to the atom) through message passing [Gilmer *et al.*, 2017]. Based on the above analysis, we achieve this goal by introducing CCA to extract features from the bond tensor in horizontal and vertical directions as the contextual messages of two atoms connected by a bond, respectively. See Appendix A.1 for more implementation details. We build the coupling network with three layers, where the first and last layers are 3×3 convolutions, while the center layer is CCA.

4.4 Two-step Molecular Graph Generation

Benefiting from the reversibility of MolHF, molecular graph generation is simply executed by drawing $z = (z_{X|A}, z_A)$ from $P_{\mathcal{Z}}(z)$ and calling the inverse transformation of MolHF. In our two-step generative manner, the bond tensor $\hat{A}$ is generated by the inverse transformation of f_A at first, then the atom matrix $\hat{X}$ is generated by the inverse transformation of $f_{X|A}$, conditioned on the pre-generated $\hat{A}$. Formally, this process can be summarized as,

$$\begin{aligned} \hat{A} &= \text{one hot}(\arg\max(\widetilde{A})), \quad \widetilde{A} = f_A^{-1}(z_A), \\ \hat{X} &= \text{one hot}(\arg\max(\widetilde{X})), \quad \widetilde{X} = f_{X|A}^{-1}(z_X, \hat{A}) \end{aligned} \tag{12}$$

where the composite operation $\text{one hot} \cdot \arg\max$ maps the dequantized continuous data back to the discrete one-hot data. Lastly, a post-hoc validity correction mechanism [Zang and Wang, 2020] is incorporated to ensure that the generated molecular graphs are valid. This generation process can exploit the hierarchical structural information captured by the inverse transformation of MolHF.

5 Experiments

Following previous works [Jin *et al.*, 2018; Zang and Wang, 2020], we demonstrate the effectiveness of our MolHF in various aspects: generation and reconstruction, property optimization, and hierarchical visualization.

Datasets. We evaluate our method on the ZINC250K dataset [Irwin *et al.*, 2012] for a fair comparison. In addition, we also use the Polymer dataset [St. John *et al.*, 2019] to verify that our method is capable of scaling to larger molecules. The ZINC250K dataset consists of 249,455 molecules with up to 38 atoms of 9 different types, and the Polymer dataset consists of 86,973 larger polymer-like molecules with up to 122 atoms of 7 different types. For the preprocessing strategy, we mostly follow [Zang and Wang, 2020], the difference is that we pad all the molecules in the ZINC250K and Polymer with 2 and 6 extra virtual atoms respectively, to ensure that each molecular graph can be coarsened over multiple scales.

Baselines. We compare MolHF with the following baselines. JT-VAE [Jin *et al.*, 2018] and HierVAE [Jin *et al.*, 2020] are VAE-based models. GCPN [You *et al.*, 2018] is an autoregressive model. GraphAF [Shi *et al.*, 2020] and GraphDF [Luo *et al.*, 2021] are autoregressive flow-based models. MoFlow [Zang and Wang, 2020], GraphCNF [Lippe and Gavves, 2021], and MolGrow [Kuznetsov and Polykovskiy, 2021] are one-shot flow-based models.

Implementation. For the generation and reconstruction, the latent variables are assumed to follow a prior isotropic Gaussian distribution $\mathcal{N}(0, \sigma^2\text{I})$, where σ is a learnable parameter denoting the standard deviation. For property optimization and hierarchical visualization, the same model trained on the ZINC250K dataset is used for all experiments. Further implementation details can be found in Appendix C.

5.1 Generation and Reconstruction

Setup. We evaluate the ability of MolHF to generate and reconstruct molecular graphs via the widely used metrics: **Validity** and **Validity w/o correction** are the percentages of chemically valid molecules among all the generated graphs with and without validity correction, respectively. **Uniqueness** is the percentage of unique molecules among all the generated valid molecules. **Novelty** is the percentage of generated valid molecules that do not appear in the training set. **Reconstruction** is the percentage of molecules in the training set that can be reconstructed from latent variables. As for the experiments on the Polymer dataset, to verify that MolHF does generate more large molecules, we introduce **Validity w/ filter** to calculate the percentage of valid large molecules with more than 38 atoms (the maximum number of atoms in the ZINC250K dataset) among all the generated graphs without validity correction. In practice, higher-quality samples are drawn from the reduced-temperature models [Kingma and Dhariwal, 2018]. To control the trade-off between quality and diversity, we sample from the distribution $\mathcal{N}(0, (t\sigma)^2\text{I})$ with the temperature $t < 1$, similar to MoFlow and MolGrow. To be comparable with the aforementioned baselines, we sample 10,000 molecular graphs and report the mean and standard deviation of results over 5 runs.

Results. Table 1 shows that MolHF outperforms the state-of-the-art baselines on all the metrics for the ZINC250K dataset. Thanks to the validity correction mechanism, MolHF achieves 100% **Validity**. We argue that **Validity w/o correction** can better assess how well models capture the data distri-

Method	Validity	Validity w/o correction	Uniqueness	Novelty	Reconstruct
JT-VAE	100%	n/a	100%	100%	76.7%
GCPN	100%	20%	99.97%	99.97%	n/a
GraphAF	100%	68%	99.10%	100%	100%
GraphDF	100%	89.03%	99.16%	100%	100%
MoFlow	100%	$81.76 \pm 0.21\%$	99.99%	100%	100%
GraphCNF	96.35%	83.41%	99.99%	100%	100%
MolGrow	100%	$57.80 \pm 7.75\%$	99.06%	99.96%	100%
MolHF	**100**%	$\mathbf{94.89 \pm 0.20}\%$	**100**%	**100**%	**100**%

Table 1: Random generation and reconstruction performance on the ZINC250k dataset.

Method	Validity	Validity w/o correction	Validity w/ filter	Uniqueness	Novelty	Reconstruct
JT-VAE	100%	n/a	n/a	94.10%	-	58.5%
HierVAE	100%	n/a	n/a	97.00%	-	79.9%
GraphAF	100%	$28.58 \pm 2.98\%$	$1.74 \pm 0.32\%$	100%	100%	100%
GraphDF	100%	$17.23 \pm 0.11\%$	0%	98.89%	100%	100%
MoFlow	100%	0%	0%	100%	100%	100%
GraphCNF	$25.32 \pm 1.10\%$	$19.32 \pm 0.96\%$	$11.82 \pm 1.65\%$	100%	100%	100%
MolHF	**100**%	$\mathbf{42.75 \pm 0.47}\%$	$\mathbf{35.62 \pm 0.43}\%$	**100**%	**100**%	**100**%

Table 2: Random generation and reconstruction performance on the Polymer dataset. Results of flow-based baselines are obtained by running their official source code. Note that we omit the results of MolGrow since the source code is not released.

bution, since the corrected graphs can be completely different from the original model outputs. Compared with one-shot flow-based models MoFlow and GraphCNF, MolHF attains an increase of 13.1% and 11.5% in **Validity w/o correction** respectively, implying the superiority of modeling the molecular graphs with hierarchical architecture. Compared with another hierarchical flow-based model MolGrow, MolHF outperforms it by a large margin (37.1%) with respect to **Validity w/o correction**, indicating that exploiting the coarsened graph structure is crucial for enhancing the performance of hierarchical generative models. Owing to the invertible nature, MolHF is able to reconstruct all molecules, whereas JT-VAE and GCPN can not. Furthermore, experiments on the Polymer dataset demonstrate that MolHF can be applied to model larger molecules that are impractical with previous flow-based models, as shown in Table 2. We find that the performance of all flow-based baselines degrades substantially when modeling larger molecules. MoFlow even cannot fit the data distribution to generate valid molecules. While GraphAF and GraphDF can generate a few small molecules in the early stage of the autoregressive generation process, they become fallible as the current molecules get larger so that only a tiny fraction of the valid molecules are large. GraphCNF, the best flow-based baseline, only generates 11.8% large molecules, which is still unsatisfactory. By contrast, MolHF can generate approximately 43% valid molecules, most of which are large. These results imply that the hierarchical architecture exploiting the coarsened graph structures enables MolHF to model the more complex distribution by better capturing local and global information of large molecules. Compared with the substructure-based model HierVAE, MolHF generates more unique molecules and reconstructs all molecules. We illustrate several valid large molecules (with rich topological complexity) generated by MolHF in Appendix B.1.

Method	1st	2nd	3rd	4th
ZINC250K (dataset)	0.948	0.948	0.948	0.948
JT-VAE	**0.948**	0.911	0.911	-
GCPN	**0.948**	0.947	0.946	-
GraphAF	**0.948**	**0.948**	0.947	0.946
GraphDF	**0.948**	**0.948**	**0.948**	-
MoFlow	**0.948**	**0.948**	**0.948**	**0.948**
MolGrow	**0.948**	**0.948**	**0.948**	-
MolHF	**0.948**	**0.948**	**0.948**	**0.948**

Table 3: Discovered novel molecules with the best QED scores.

Empirical running time. MolHF is significantly more efficient in aspects of training and sampling. For the ZINC250K dataset, the reported running time of MoFlow and MolGrow is 22 hours and 2 days respectively, while MolHF converges faster and can get the results within 12 hours. For the Polymer dataset, due to the huge memory cost for parallel training, autoregressive GraphAF and GraphDF cost roughly 50 minutes/epoch to train, while MolHF is $3\times$ faster. Besides, GraphAF and GraphDF cost 2 hours and 20 minutes to randomly sample 10,000 molecules (without validity correction) respectively, whereas MolHF only takes 50 seconds, $144\times$ and $24\times$ faster. The efficiency is compared on the same computing facilities using 1 Tesla V100 GPU.

5.2 Property Optimization

Setup. The unconstrained property optimization task aims at discovering novel molecules with the best **QED**, which measures the druglikeness of the molecules. The constrained property optimization task aims at modifying the given

	JT-VAE			GCPN		
δ	**Improvement**	**Similarity**	**Success**	**Improvement**	**Similarity**	**Success**
0.0	1.91±2.04	0.28±0.15	97.5%	4.20±1.28	0.32±0.12	100.0%
0.2	1.68±1.85	0.33±0.13	97.1%	4.12±1.19	0.34±0.11	100.0%
0.4	0.84±1.45	0.51±0.10	83.6%	2.49±1.30	0.48±0.08	100.0%
0.6	0.21±0.71	0.69±0.06	46.4%	0.79±0.63	0.68±0.08	100.0%
	GraphDF			MolHF		
δ	**Improvement**	**Similarity**	**Success**	**Improvement**	**Similarity**	**Success**
0.0	5.93±1.97	0.30±0.12	100.0%	**6.98±3.37**	0.25±0.13	100.0%
0.2	5.62±1.65	0.32±0.10	100.0%	**6.29±3.36**	0.32±0.13	99.8%
0.4	**4.13±1.41**	0.47±0.07	100.0%	3.97±2.78	0.52±0.12	90.9%
0.6	1.72±1.15	0.67±0.06	93.0%	**2.55±2.88**	0.72±0.11	55.9%

Table 4: Constrained optimization performance on the test set.

	MoFlow			MolGrow		
δ	**Improvement**	**Similarity**	**Success**	**Improvement**	**Similarity**	**Success**
0.0	8.61±5.44	0.30±0.20	98.88%	14.84±5.79	0.05±0.04	100.00%
0.2	7.06±5.04	0.43±0.20	96.75%	**11.99±6.45**	0.23±0.05	99.80%
0.4	4.71±4.55	0.61±0.18	85.75%	**8.34±6.85**	0.44±0.05	99.80%
0.6	2.10±2.86	0.79±0.14	58.25%	4.01±5.61	0.65±0.07	97.70%
	GraphAF			MolHF		
δ	**Improvement**	**Similarity**	**Success**	**Improvement**	**Similarity**	**Success**
0.0	13.13±6.89	0.29±0.15	100.00%	**15.22±5.99**	0.15±0.16	100.00%
0.2	11.90±6.86	0.33±0.12	100.00%	11.02±6.20	0.36±0.20	98.38%
0.4	8.21±6.51	0.49±0.09	99.88%	7.18±7.06	0.61±0.18	86.13%
0.6	4.98±6.49	0.66±0.05	96.88%	**6.46±7.67**	0.78±0.13	63.25%

Table 5: Constrained optimization performance on the entire set.

molecules to improve **PlogP**, which denotes logP score penalized by ring size and synthetic accessibility, with the constraint that the Tanimoto similarity of Morgan fingerprint between the modified and original molecules is above a threshold δ [Jin *et al.*, 2018]. We formulate property optimization as a latent space optimization (LSO) problem [Gómez-Bombarelli *et al.*, 2018]. Formally, we first train a surrogate network h which takes as input the latent variables z encoded with the trained MolHF to predict the property values, then perform gradient ascent $z \leftarrow z + \alpha \cdot \nabla_z h$ iteratively to navigate the latent space. We decode the latest latent variables with MolHF to obtain the optimized molecule. For constrained property optimization, we start from the latent variables of 800 molecules with the lowest **PlogP** [Jin *et al.*, 2018]. Note that two different sets of molecules selected from the test set [Jin *et al.*, 2018] or the entire set [Shi *et al.*, 2020] are used in prior methods. We thus report results separately.

Results. We show the best scores discovered in the unconstrained optimization task in Table 3. Like MoFlow, MolHF finds more novel molecules with the best QED score (0.948) appearing in the dataset. Further, we illustrate the histogram of the optimized property scores in Figure 2 to verify that MolHF finds molecules with higher QED scores than MoFlow on average (0.70 vs. 0.56). As shown in Table 4 and Table 5, MolHF achieves higher property improvement than GraphDF under 3 of 4 similarity constraints and outperforms GraphAF and MolGrow under 2 of 4 similarity constraints, while maintaining higher similarities. Compared with the autoregressive GCPN, GraphDF, and GraphAF which fine-tune the generator with reinforcement learning and optimize each molecule hundreds of times, the optimization strategy of MolHF is more conveniently implementable and efficient. This nonetheless results in the issues of success rate and robustness. Fortunately, these issues can be alleviated with the recent advanced LSO approaches [Tripp *et al.*, 2020; Notin *et al.*, 2021], potentially leading to better performance.

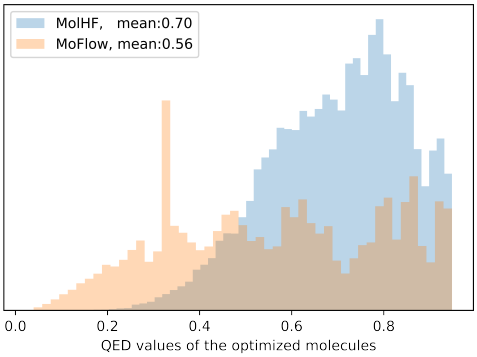

Figure 2: The histogram of the optimized QED.

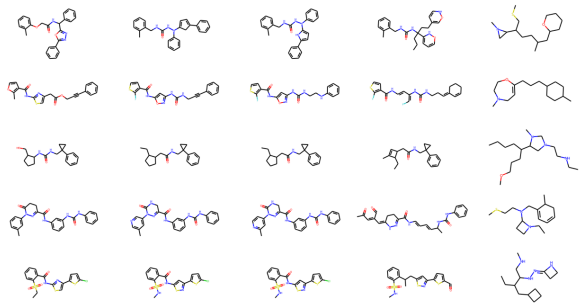

Figure 3: The leftmost column represents the original molecules, the subsequent columns are obtained by resampling the latent variables of coarser scale, resampling more and more as we go right.

5.3 Hierarchical Visualization

To investigate the generative mechanism of MolHF, we visualize the generated molecules in terms of the semantics of the learned latent variables and the generated substructures.

Semantics of the learned latent variables. We infer the latent variables and resample the latent variables of the finest scale, increasing the coarsest scale affected by this resampling [Dinh *et al.*, 2016]. We decode the resampled latent variables and illustrate several cases in Figure 3, where the leftmost column represents the original molecules, and the subsequent columns (from left to right) are obtained by resampling the latent variables of coarser scale, resampling more and more as we go right. As we can see, latent variables of finer scale affect the local structure (atom and bond), while those of coarser scale affect the global structure (skeletal structure).

Generated substructures. In Figure 4, we illustrate some of the most frequent substructures generated by MolHF. These learned substructures are chemically meaningful, demonstrating that MolHF does capture the rich valid structure and semantic information of molecular graphs. By comparison, we find that these substructures (such as 2-butylene, methylethylamine, and acetamide, to name a few) often appear in the fragment vocabulary [Jin *et al.*, 2018; Jin *et al.*, 2020; Xie *et al.*, 2021; Kong *et al.*, 2022] manually

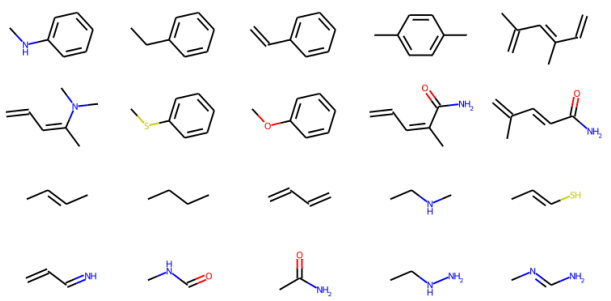

Figure 4: Illustration of the most frequent substructures generated by MolHF.

extracted based on frequency of occurrence. Thus, we argue that MolHF successfully extracts common substructures from the data without any need for domain knowledge, and matches the prior distribution of fragments to some extent. Meanwhile, due to the hierarchical generation, some of the substructures generated by MolHF are composed of smaller structural motifs. For instance, the substituted benzene rings can be further divided according to the retrosynthesis-based BRICS algorithm [Degen *et al.*, 2008]. To link coarsened graph structure with substructures, we decompose several molecules with the same coarsened graph structure in Appendix B.2, where atoms are colored according to the substructures to which they belong. We observe that MolHF learns to hierarchically generate molecular graphs in a coarse-to-fine manner by first generating a coarsened graph structure (skeletal structure) and then refining the substructures.

6 Discussion and Conclusion

In this paper, we present MolHF, a hierarchical flow-based model to generate molecular graphs in a coarse-to-fine manner. As demonstrated with extensive experiments, MolHF achieves state-of-the-art performance in random generation and property optimization, while being capable of modeling larger molecules that are impractical with prior flow-based methods. We hope that our MolHF can spur further research to unleash the great potential of hierarchical generative models in the graph generation field.

Limitations and future work. The strategy used in the property optimization task is inapplicable for real-world drug discovery, as it is expensive to evaluate the properties of molecules with wet-lab experiments [Zhu *et al.*, 2023]. We believe that developing novel sample-efficient optimization approaches is a fruitful avenue. In the future, one might extend MolHF to generate other types of graphs with hierarchical structures.

Acknowledgments

This research was partially supported by the National Key R&D Program of China under grant No.2019YFC0118802, Key R&D Program of Zhejiang Province under grant No.2020C03010, National Natural Science Foundation of China under grant No.62176231. Tingjun Hou's research was supported in part by the National Natural Science Foundation of China under grant No.22220102001.

Contribution Statement

Yiheng Zhu and Zhenqiu Ouyang contributed equally to this work.

References

[Behrmann *et al.*, 2021] Jens Behrmann, Paul Vicol, Kuan-Chieh Wang, Roger Grosse, and Jörn-Henrik Jacobsen. Understanding and mitigating exploding inverses in invertible neural networks. In *AISTATS*, 2021.

[De Cao and Kipf, 2018] Nicola De Cao and Thomas Kipf. MolGAN: An implicit generative model for small molecular graphs. *ICML workshop*, 2018.

[Degen *et al.*, 2008] Jörg Degen, Christof Wegscheid-Gerlach, Andrea Zaliani, and Matthias Rarey. On the art of compiling and using'drug-like'chemical fragment spaces. *ChemMedChem: Chemistry Enabling Drug Discovery*, 3(10):1503–1507, 2008.

[Dinh *et al.*, 2014] Laurent Dinh, David Krueger, and Yoshua Bengio. Nice: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*, 2014.

[Dinh *et al.*, 2016] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2016.

[Gilmer *et al.*, 2017] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *ICML*, pages 1263–1272. PMLR, 2017.

[Gómez-Bombarelli *et al.*, 2018] Rafael Gómez-Bombarelli, Jennifer N Wei, David Duvenaud, José Miguel Hernández-Lobato, Benjamín Sánchez-Lengeling, Dennis Sheberla, Jorge Aguilera-Iparraguirre, Timothy D Hirzel, Ryan P Adams, and Alán Aspuru-Guzik. Automatic chemical design using a data-driven continuous representation of molecules. *ACS central science*, 4(2):268–276, 2018.

[Goodfellow *et al.*, 2014] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[Ho *et al.*, 2019] Jonathan Ho, Xi Chen, Aravind Srinivas, Yan Duan, and Pieter Abbeel. Flow++: Improving flow-based generative models with variational dequantization and architecture design. In *ICML*, 2019.

[Huang *et al.*, 2019] Zilong Huang, Xinggang Wang, Lichao Huang, Chang Huang, Yunchao Wei, and Wenyu Liu. Ccnet: Criss-cross attention for semantic segmentation. In *ICCV*, 2019.

[Irwin *et al.*, 2012] John J Irwin, Teague Sterling, Michael M Mysinger, Erin S Bolstad, and Ryan G Coleman. Zinc: a free tool to discover chemistry for biology. *JCIM*, 52(7):1757–1768, 2012.

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

[Jin *et al.*, 2020] Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Hierarchical generation of molecular graphs using structural motifs. In *ICML*, 2020.

[Karras *et al.*, 2018] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. In *ICLR*, 2018.

[Kingma and Dhariwal, 2018] Diederik P Kingma and Prafulla Dhariwal. Glow: generative flow with invertible $1\times$ 1 convolutions. In *NeurIPS*, 2018.

[Kingma and Welling, 2013] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[Kipf and Welling, 2017] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.

[Kong *et al.*, 2022] Xiangzhe Kong, Wenbing Huang, Zhixing Tan, and Yang Liu. Molecule generation by principal subgraph mining and assembling. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *NeurIPS*, 2022.

[Kuznetsov and Polykovskiy, 2021] Maksim Kuznetsov and Daniil Polykovskiy. Molgrow: A graph normalizing flow for hierarchical molecular generation. In *AAAI*, 2021.

[Liang *et al.*, 2021] Jingyun Liang, Andreas Lugmayr, Kai Zhang, Martin Danelljan, Luc Van Gool, and Radu Timofte. Hierarchical conditional flow: A unified framework for image super-resolution and image rescaling. In *ICCV*, pages 4076–4085, 2021.

[Lippe and Gavves, 2021] Phillip Lippe and Efstratios Gavves. Categorical normalizing flows via continuous transformations. In *ICLR*, 2021.

[Luo *et al.*, 2021] Youzhi Luo, Keqiang Yan, and Shuiwang Ji. Graphdf: A discrete flow model for molecular graph generation. In *ICML*, 2021.

[Madhawa *et al.*, 2019] Kaushalya Madhawa, Katushiko Ishiguro, Kosuke Nakago, and Motoki Abe. Graphnvp: An invertible flow model for generating molecular graphs. *arXiv preprint arXiv:1905.11600*, 2019.

[Notin *et al.*, 2021] Pascal Notin, José Miguel Hernández-Lobato, and Yarin Gal. Improving black-box optimization in vae latent space using decoder uncertainty. *Advances in Neural Information Processing Systems*, 34, 2021.

[Papamakarios *et al.*, 2017] George Papamakarios, Theo Pavlakou, and Iain Murray. Masked autoregressive flow for density estimation. *Advances in neural information processing systems*, 30, 2017.

[Papamakarios *et al.*, 2021] George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing flows for probabilistic modeling and inference. *JMLR*, 2021.

[Polishchuk *et al.*, 2013] Pavel G Polishchuk, Timur I Madzhidov, and Alexandre Varnek. Estimation of the size of drug-like chemical space based on gdb-17 data. *Journal of computer-aided molecular design*, 27(8):675–679, 2013.

[Ramachandran *et al.*, 2017] Prajit Ramachandran, Barret Zoph, and Quoc V Le. Searching for activation functions. *arXiv preprint arXiv:1710.05941*, 2017.

[Rezende and Mohamed, 2015] Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *ICML*, 2015.

[Schlichtkrull *et al.*, 2018] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *ESWC*, 2018.

[Shi *et al.*, 2020] 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.

[Simonovsky and Komodakis, 2018] Martin Simonovsky and Nikos Komodakis. Graphvae: Towards generation of small graphs using variational autoencoders. In *ICANN*. Springer, 2018.

[St. John *et al.*, 2019] Peter C St. John, Caleb Phillips, et al. Message-passing neural networks for high-throughput polymer screening. *The Journal of chemical physics*, 150(23):234111, 2019.

[Tripp *et al.*, 2020] Austin Tripp, Erik Daxberger, and José Miguel Hernández-Lobato. Sample-efficient optimization in the latent space of deep generative models via weighted retraining. *Advances in Neural Information Processing Systems*, 33:11259–11272, 2020.

[Xie *et al.*, 2021] 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.

[Ying *et al.*, 2018] Rex Ying, Jiaxuan You, Christopher Morris, Xiang Ren, William L Hamilton, and Jure Leskovec. Hierarchical graph representation learning with differentiable pooling. In *NeurIPS*, 2018.

[You *et al.*, 2018] Jiaxuan You, Bowen Liu, Rex Ying, Vijay Pande, and Jure Leskovec. Graph convolutional policy network for goal-directed molecular graph generation. In *NeurIPS*, 2018.

[Yu *et al.*, 2020] Jason J Yu, Konstantinos G Derpanis, and Marcus A Brubaker. Wavelet flow: Fast training of high resolution normalizing flows. *NeurIPS*, 33:6184–6196, 2020.

[Zang and Wang, 2020] Chengxi Zang and Fei Wang. Moflow: an invertible flow model for generating molecular graphs. In *SIGKDD*, 2020.

[Zhu *et al.*, 2023] Yiheng Zhu, Jialu Wu, Chaowen Hu, Jiahuan Yan, Chang-Yu Hsieh, Tingjun Hou, and Jian Wu. Sample-efficient multi-objective molecular optimization with gflownets. *arXiv preprint arXiv:2302.04040*, 2023.