GF-VAE: A Flow-based Variational Autoencoder for Molecule Generation

Changsheng Ma

King Abdullah University of Science and Technology
Jeddah, Saudi Arabia
changsheng.ma@kaust.edu.sa

Xiangliang Zhang*

King Abdullah University of Science and Technology
Jeddah, Saudi Arabia
xiangliang.zhang@kaust.edu.sa

ABSTRACT

Generating novel molecules with desired properties is a fundamental problem in modern drug discovery. This is a challenging problem because it requires the optimization of the given objectives while obeying the rules of chemical valence. An effective approach is to incorporate the molecular graph with deep generative models. However, recent generative models with high-performance are still computationally expensive. In this paper, we propose GF-VAE, a flow-based variational autoencoder (VAE) model for molecular graph generation. Specifically, the model equips VAE a lightweight flow model as its decoder, in which, the encoder aims to accelerate the training process of the decoder, while the decoder in turns to optimize the performance of the encoder. Thanks to the invertibility of flow model, the generation process is easily accomplished by reversing the decoder. Additionally, the final generated molecules are processed by validity correction. Therefore, our GF-VAE inherits the advantages of both VAE and flow-based methods. We validate our model on molecule generation and reconstruction, smoothness of learned latent space, property optimization and constrained property optimization. The results show that our model achieves state-of-the-arts performance on these tasks. Moreover, the time performance of GF-VAE on two classical datasets can achieve 31.3% and 62.9% improvements separately than the state-of-the-art model.

CCS CONCEPTS

• **Theory of computation** → *Graph algorithms analysis*; • **Computing methodologies** → *Machine learning*; *Maximum likelihood modeling*; *Latent variable models*; *Unsupervised learning*;

KEYWORDS

Graph Generative Model, Variational AutoEncoder, Normalizing Flow, Molecular Generation, Molecular property optimization

ACM Reference Format:

Changsheng Ma and Xiangliang Zhang. 2021. GF-VAE: A Flow-based Variational Autoencoder for Molecule Generation. In *Proceedings of the 30th ACM Int'l Conf. on Information and Knowledge Management (CIKM '21)*.

*Xiangliang Zhang is the corresponding author

This work is licensed under a Creative Commons Attribution International 4.0 License.

CIKM '21, November 1–5, 2021, Virtual Event, Australia.

© 2021 Copyright is held by the owner/author(s).

ACM ISBN 978-1-4503-8446-9/21/11.

<https://doi.org/10.1145/3459637.3482260>

November 1–5, 2021, Virtual Event, Australia. ACM, New York, NY, USA, 10 pages. <https://doi.org/10.1145/3459637.3482260>

1 INTRODUCTION

Finding new molecules with desired properties is a fundamental problem in chemical biology and drug discovery. The discovery paradigm involves investments both in organic synthesis and in biological testing of large compound collections. According to the estimation of chemists, there are 10^{60} drug-like compounds that could be made [24]. Moreover, for each drug in market, there can be millions of compounds that are chemically almost identical to it, some of which may work better than the approved drug by just slightly changing the compound structure with an extra hydrogen or double bond [24]. Computer algorithms with scripted instructions have been used to screen millions of compounds for finding the top-ranked alternative candidates [22]. Recently, machine learning based algorithms are designed to study the less-explored regions of chemical space and generate novel compounds with desirable property profiles from scratch [4].

The basic idea of molecule generation based on machine learning is to approximate one data distribution, which could capture as much as possible the patterns that are presented in the given data set. In recent years, different generative models have been developed to learn data distribution, such as generative adversarial networks (GANs) [11], Variational autoencoders (VAEs) [17] and flow-based models [7, 8]. They have been applied to molecule generation resulting in, for example, MolGAN [6] and GCPN [32] based on GAN, CVAE [10] and JT-VAE [14] based on VAE, GraphNVP [21] and MoFlow [33] based on flow-based models.

Current VAE-based molecule generation models, e.g., GVAE [18] and GraphVAE [30], first use the encoder f to map a molecular from its source space to a latent space vector z , and then use the decoder g hoping to map it back. The mapping functions f and g can be deep networks learned through optimizing a variational lower bound on the log-likelihood of the data, such that the generator g approximately inverts the encoder f . Although enjoying the flexibility of the forms of f and g , the optimization of a lower bound criterion might yield a suboptimal solution with respect to the true log-likelihood [7]. The failure to optimize the decoder variance in typical implementations of VAE is a major cause of poor quality of generated samples [5]. In contrast, the flow-based models seek for a perfect auto-encoder pair, i.e., $g = f^{-1}$. The invertible mapping allows exact and tractable likelihood estimation for training, ensures to reconstruct all the training samples, and supports for various applications such as interpolations between samples and meaningful modifications of existing samples [15]. The flow-based models thus have potential capability of deeper exploration of the

huge chemical space, which could produce more valid and novel molecules. However, in order to capture a better invertible mapping function for flow models, a large number of coupling layers are required, since the mapping function is often implemented by composing a series of easily invertible modules (e.g., additive coupling layers). For example, Glow [15] has 192 layers. Therefore, the flow models are often heavyweight and require a long training process. This shortcoming of flow models has been investigated in image domain [23], not yet for molecular domain. Unlike image generation, molecular generation suffers from very low fault tolerance and is very sensitive to noise. Even one wrong generation of valence will lead to failure of molecule generation. Therefore, molecular generation naturally benefits from the attribute of exact density evaluation of flow models. It is thus essential to design light-weight flow-based molecular generator.

In this paper, we propose GF-VAE which is short for *Graph Flow - Variational AutoEncoder* to generate molecular graphs. The key idea behind it is to attach a VAE-encoder with a flow-generator, which plays the role of decoder. During the training process, the encoder learns a mapping from original molecular space to a Gaussian space, and the decoder will also learn this mapping as well as optimize it, guiding the encoder to learn more meaningful high level molecule representation (shown in experiment 5.3). This is a perfect marriage between VAE model and normalization flow model, which combines the advantages of both, while alleviating or even eliminating their shortcomings. On the side of flow models, the model weight is lightened because its mapping target $p(\mathbf{z})$ is a Gaussian distribution from VAE-encoder rather than a typical target which is a standard Gaussian distribution without any original data distribution information. i.e., f is not necessarily implemented by a large number of additive coupling layers any more. On the other side of VAE-based models, since the poor performance of typical VAE is due to the failure to optimize the decoder variance [26], and [16] pointed out that the posterior be made more flexible via the use of normalizing flows. Therefore, attaching the flow model to VAE-encoder, providing a tighter, modified variational lower bound, which enables the approximate posterior of VAE can achieve a better fit to the true posterior.

The flow-generator in GF-VAE is designed with the cooperation of two sub-flow modules followed by a validity check module. The generator learns the molecular graph structure (bond) and molecular atom assignments separately based on the global feature from VAE-encoder. This design allows the generator to generate molecules in one-shot. GF-VAE is thus more efficient than the sequential generation models that generate atoms and bonds of a molecule one by one [29]. It's also not restricted by the limitation that the sequential model requires a predefined sequence to specify the generation process (usually in accordance with the SMILES sequence). In addition, due to the reversibility of flow-generator, GF-VAE bypasses the reconstruction loss that usually involves in computationally demanding graph matching process [30].

In summary, our contributions in this work are as follows:

- We propose GF-VAE, a seamless combination of VAE and normalizing flow model, for one-shot molecule graph generation. GF-VAE uses a flow-generator to guide a VAE-encoder to learn more meaningful high-level molecule representations,

while it adopts a VAE-encoder to make the flow-generator lightened with less weights.

- Experimental results on two benchmark datasets show that GF-VAE outperforms the state-of-the-art models on all evaluation metrics, such as validity, novelty, uniqueness and reconstruction. The flow-generator effectively guides the learning of the encoder. The meaningfulness of the molecule embedding and the smoothness of the learned latent space are demonstrated.
- The lightweight GF-VAE is around two times faster than the state-of-the-art MoFlow model in training. Our extensive evaluation also shows that a lightweight GF-VAE can have better performance than heavier-weight MoFlow.

2 RELATED WORK

A variety of deep generative models have been proposed for *de-novo* molecule generation. Some models take molecular SMILES codes [31] as input, and others directly deal with the molecular graph structure. Models working with SMILES strings address molecule generation as a text generation problem, which has been extensively studied in natural language processing field. However, the linear string description of molecular structures in SMILES causes two difficulties in molecule generation. First, the SMILES representation cannot well capture the structural information of molecules, e.g., two structurally similar molecules may be encoded into notable different SMILES strings, making the distribution complicated to learn [14]. Second, the models tend to learn the specific grammar rules of SMILES strings, rather than the molecular features [3]. We thus focus on learning from molecular graphs, and discuss next the related work using VAE and flow-based models.

In the group of VAE-based models, from our best knowledge, CVAE [10] is the first work using VAE to encode and generate molecular SMILES. GVAE [18] parses SMILES as a tree from a context-free grammar and employs VAE model for encoding and decoding these parsed trees, ensuring the generated outputs are always valid. Unlike these SMILES-oriented models, GraphVAE [30] is a pioneering work using VAE for molecular graph generation, applying VAE to realize graph-to-graph autoencoding. By transforming a constrained optimization problem to a regularized unconstrained one, RVAE [20] presents a regularization framework for VAE-based molecule generation as a step toward semantic validity. In stead of generating atoms one by one, JT-VAE [14] generates scaffold of chemical substructure components step by step.

In the group of flow-based models, GraphNVP [21] is the first work, generating molecular graphs by invertible flow model in a one-shot manner. GRF [12] improves GraphNVP by using residual flows. GraphAF [29] is an autoregressive flow-based model which generates molecular graphs in a sequential manner with validity check when adding any new atom or bond. MoFlow [33] is a recent model achieving state-of-the-art performance in molecular graph generation. The model consists of a variant of Glow model [15] for generating bonds, a novel graph conditional flow for generating atoms given bonds, and then combining them with validity corrections. To enable the invertible mapping function to map complicated molecular structures to a simple standard Gaussian distribution, one-shot flow generative models (such as GraphNVP and MoFlow) stack around 50 coupling layers (more than 100 basic

Figure 1: Architecture of GF-VAE (training stage)

neural network layers, such as linear layer, convolutional layer) in total, which require a long training process. We thus propose to combine VAEs and flow-based models in molecular generation to take the advantages of both, while alleviating the shortcomings of both.

3 THE PROPOSED GF-VAE MODEL

In this section, we first give the problem definition, and then describe each component of our model in detail. The architecture of our model at the training stage is shown as Figure 1.

3.1 Problem Definition

From the perspective of graph, a molecule can be viewed as an undirected graph with multi-type nodes and multi-type edges, each node of a molecular graph pictures an atom while each edge represents a chemical bond. Giving R types of bonds (e.g., single, double, etc.) and M types of atoms (e.g., oxygen, carbon, etc.), a molecular graph can then be denoted by $G = (A, X)$, where $A \in \{0, 1\}^{N \times N \times R}$ is an adjacency tensor of N atoms, and $X \in \{0, 1\}^{N \times M}$ is a feature matrix showing the node types of N atoms.

Our main goal is to learn a generative model $p_G(G)$ from a given set of graphs $\mathcal{G}$, such that a sample drawn from the distribution p_G is a valid molecular graph.

Our proposed GF-VAE model attaches a VAE-encoder with a flow-generator, as shown in Figure 1.

3.2 VAE-encoder

The VAE-encoder maps a molecular graph $G = (A, X)$ to a latent vector $\mathbf{z} \in \mathbb{R}^{N \times N \times R + N \times M}$, which belongs to a Gaussian distribution $\mathcal{N}_z(\mu_z, \Sigma_z)$, where μ_z and Σ_z are mean and diagonal covariance matrix of $\mathbf{z}$ separately. We implement the encoder by Relational-GCN [28], which is a classical and effective model for graph embedding with multiple types of edges. From the layer l , the feature vector of node (atom) i at layer $l + 1$ is updated as

$$\begin{aligned} \mathbf{h}_i^{(\ell+1)} &= f_s^{(\ell)}(\mathbf{h}_i^{(\ell)}, \mathbf{x}_i) + \sum_{j=1}^N \sum_{r=1}^R \frac{A_{ijr}}{|\mathcal{S}_i|} f_r^{(\ell)}(\mathbf{h}_j^{(\ell)}, \mathbf{x}_i) \\ \mathbf{h}_i^{(\ell+1)} &= \tanh(\mathbf{h}_i^{(\ell+1)}) \end{aligned} \quad (1)$$

Figure 2: Atom Flow & Bond Flow in GF-VAE

where $\mathbf{x}_i \in \{0, 1\}^M$ is the atom feature from X , and $\mathbf{h}_i^{(\ell)}$ is the feature vector of the node i at layer ℓ (at the initial layer $\mathbf{h}_i^{(0)} = \mathbf{x}_i$). The function $f_s^{(\ell)}$ acts as a self-connection of node i between layers, to merge $\mathbf{h}_i^{(\ell)}$ and $\mathbf{x}_i$ for propagation to next layer. The $f_r^{(\ell)}$ is an edge type-specific function, which takes all one-hop neighbors of node i to update its feature vector. The set $\mathcal{S}_i$ denotes the neighbors for node i .

After several layers of propagation, following [19], we aggregate nodes embedding into a graph level representation vector as

$$\begin{aligned} \mathbf{h}'_G &= \sum_{v \in \mathcal{V}} \sigma\left(\text{nn}\left(\mathbf{h}_v^{(L)}, \mathbf{x}_v\right)\right) \odot \tanh\left(\text{nn}\left(\mathbf{h}_v^{(L)}, \mathbf{x}_v\right)\right) \\ \mathbf{h}_G &= \tanh \mathbf{h}'_G \end{aligned} \quad (2)$$

where $\mathcal{V}$ includes all nodes in the graph, $\sigma(\cdot)$ is the sigmoid function, nn is a linear neural network and $\odot$ denotes element-wise multiplication.

To force the encoding vector $\mathbf{z}$ to roughly conform to a normal distribution, we separately apply linear neural network to $\mathbf{h}_G$ to get a mean vector μ_z and a diagonal covariance matrix Σ_z ,

$$\mu_z = \text{nn}(\mathbf{h}_G); \quad \Sigma_z = \text{nn}(\mathbf{h}_G). \quad (3)$$

Then, based on the reparameterization trick, the latent vector $\mathbf{z}$ is obtained by

$$\mathbf{z} = \mu_z + \epsilon \odot \Sigma_z \quad (4)$$

where ϵ is a random Gaussian variable. The capability of the encoder is measured by the Kullback–Leibler divergence between the posterior distribution and a simple prior, e.g., a standard Gaussian distribution:

$$\mathcal{L}_{\text{KL}} = \mathcal{KL}[\mathcal{N}_z(\mu_z, \Sigma_z) \parallel \mathcal{N}(\mathbf{0}, \mathbf{I})] \quad (5)$$

where $\mathcal{KL}$ denotes the Kullback–Leibler divergence.

3.3 Flow-generator

Our flow-generator is composed of two cooperative parts: the *atom* flow aims to approximate the distribution of atoms, and the *bond* flow is used to capture the distribution of bonds. Since the process of probability calculation by flow models refers to Jacobian matrix,

directly applying a continuous density model on discrete components may result in degenerate probability distributions. Therefore, we dequantize G by adding a uniform random noise $U[0, 1]$ to A and X ,

$$\begin{aligned} A' &= A + c * u; \quad u \sim U[0, 1)^{N \times N \times R} \\ X' &= X + c * v; \quad v \sim U[0, 1)^{N \times M} \end{aligned} \quad (6)$$

where $0 < c < 1$ is a scaling hyperparameter.

The obtained $G' = (A', X')$ is then sent to the flow generator to learn a bijection from the original space to the latent space such that the mapped graph vectors in the latent space follow a Gaussian distribution. Since we have obtained VAE encoding vectors $\mathbf{z}$ in a Gaussian distribution, the atom flow and bond flow only need to map X' and A' close to $\mathbf{z}$, instead of conducting as much work as a fully marginal normalization flow model to map to a standard Gaussian distribution.

For the bond flow, we base it on the Glow model. It learns an invertible mapping for $\mathbf{z}_A = f_{\theta}^{\text{bond}}(A')$, which maps $A' \in \mathbb{R}^{R \times N \times N}$ to $\mathbf{z}_A \in \mathbb{R}^{R \times N \times N}$, using all process shown in bond flow of Figure 2 from bottom to top, such as *squeeze+actnorm*, *invertible 1*1 convolution*, and *masked convolution*, which will be introduced later. For the *bond coupling layer*, similar to that used in image processing, we build it by $3 \times 3 \text{conv}2d \rightarrow \text{BatchNorm}2d \rightarrow \text{ReLU}$ layers and stack K_B times. Moreover, the whole module is stacked L_B times to enhance the mapping performance.

For the atom flow, we use the same module presented in [33]. An invertible mapping function f_{ω}^{atom} is learned to get $\mathbf{z}_X \in \mathbb{R}^{N \times M}$ by $f_{\omega}^{\text{atom}}(X', A)$. As shown in atom flow of Figure 2, the atom affine coupling layer consists of K_A times stacked *graphconv* $\rightarrow \text{BatchNorm}1d \rightarrow \text{ReLU}$ layers with a Multilayer Perceptron (MLP) output layer; here *graphconv* is also implemented by Relational-GCN. Since there is no channel in the 2-d atom matrix, we need to normalize each row instead of each channel, similarly for the masked convolution process. Moreover, the whole module is stacked L_A times to achieve a better mapping performance.

Since both f_{ω}^{atom} and f_{θ}^{bond} are invertible, given latent vectors $\mathbf{z}_X$ and $\mathbf{z}_A$ drawn from a known prior distribution, the probability of a given graph $G = (A, X)$ can be calculated as

$$\begin{aligned} p(X'|\mathbf{z}_X, A)p(A'|\mathbf{z}_A) &= p(\mathbf{z}_X) \left(\left| \det \left(\frac{\partial f_{\omega}^{\text{atom}}(X', A)}{\partial X'} \right) \right| \right) \\ p(\mathbf{z}_A) &\left(\left| \det \left(\frac{\partial f_{\theta}^{\text{bond}}(A')}{\partial A'} \right) \right| \right) \end{aligned} \quad (7)$$

where $p(\mathbf{z}_X)$ and $p(\mathbf{z}_A)$ are $\mathcal{N}(\mu_z, \Sigma_z \mathbf{I})$, determined by the VAE-encoder. The loss for the flow module is thus defined as the negative log-likelihood,

$$\mathcal{L}_{\text{flow}} = -(\log p(X'|\mathbf{z}_X, A) + \log p(A'|\mathbf{z}_A)). \quad (8)$$

We next discuss the details of the bond/atom flow structure.

Squeeze & unsqueeze. *Squeeze* aims to increase the size of the channel axis, meanwhile, retaining the local correlation. Given the squeeze fold h , the input dimension $R \times N \times N$ is squeezed to $(Rh^2) \times \frac{N}{h} \times \frac{N}{h}$. The number of channels is increased to Rh^2 , and channel sizes are reduced to $\frac{N}{h}$. *Unsqueeze* is the inverse process of squeeze.

Actnorm. This process performs an affine transformation of the activations using a scale and bias parameter per channel. These

Figure 3: Generation process of GF-VAE

parameters are initialized such that the post-actnorm activations per-channel have zero mean and unit variance given an initial minibatch of data.

Invertible 1*1 convolution. This process is a trainable permutation matrix used to rearrange the dimensions to learn an optimal partition of input, using LU decomposition to reduce the time complexity.

Affine coupling layer. This process is a powerful reversible transformation where the forward function, the reverse function and the log determinant are computationally efficient. We follow the formula of RealNVP [8].

$$\begin{aligned} \mathbf{y}_{1:d} &= \mathbf{x}_{1:d} \\ \mathbf{y}_{d+1:D} &= \mathbf{x}_{d+1:D} \odot \exp(s(\mathbf{x}_{1:d})) + t(\mathbf{x}_{1:d}) \end{aligned} \quad (9)$$

where $\mathbf{x}$ is the input and $\mathbf{y}$ is the output, s and t are functions from $\mathbb{R}^d \mapsto \mathbb{R}^{D-d}$, and $\odot$ is the element-wise product. In terms of implementation, we adopt Sigmoid function instead of exponential function for the sake of the numerical stability of cascading multiple flow layers.

Masked convolution. This module is hoping to cover some channels to increase the randomness of the training process to improve the generalization performance of the model. Partitioning can be implemented by using a binary mask $\mathbf{b}$, for input $\mathbf{x}$, the output $\mathbf{y}$ will be

$$\mathbf{y} = \mathbf{b} \odot \mathbf{x} + (1 - \mathbf{b}) \odot (\mathbf{x} \odot \exp(s(\mathbf{b} \odot \mathbf{x})) + t(\mathbf{b} \odot \mathbf{x})). \quad (10)$$

3.3.1 Loss Function. As shown in Figure 1, in addition to $\mathcal{L}_{\text{KL}}$ defined in Eq (5) and $\mathcal{L}_{\text{flow}}$ defined in Eq (8), we have $\mathcal{L}_{\text{recon}}$ to consider the reconstruction loss in the latent space, thanks to the reversible properties of flow models. The loss is defined as

$$\mathcal{L}_{\text{recon}} = \text{rmse}(\mathbf{z}, \text{concat}(\mathbf{z}_X, \mathbf{z}_A)) \quad (11)$$

where *rmse* is short for root-mean-square error function. Comparing with the reconstruction loss in original VAE, this loss (11) in latent space avoids complicated graph matching.

In the end, the overall loss function of GF-VAE model is

$$\mathcal{L}_{\text{GF-VAE}} = \mathcal{L}_{\text{KL}} + \mathcal{L}_{\text{flow}} + \mathcal{L}_{\text{recon}}. \quad (12)$$

3.4 Training, Generation & Validity Correction

As for the training process, we perform the whole process shown in Figure 1 as forward calculation on training graphs by minimizing the overall loss function in Eq (12). The model parameters in VAE-encoder, atom and bond flow models are jointly optimized.

As for the generation process, shown in Figure 3, we first draw a random sample $\mathbf{z}$ from a standard Gaussian distribution and split it into $\mathbf{z}_A$ and $\mathbf{z}_X$. Since the generator of our flow model (f_{ω}^{atom} and f_{θ}^{bond}) is invertible, we simply put $\mathbf{z}_A$ into f_{ω}^{atom} and $\mathbf{z}_X$ into f_{θ}^{bond} separately but in reverse order.

It is worth noting that the atom flow and bond flow can be performed simultaneously during training, because their update are independent of each other. However, the atom flow requires valid bond tensor as an input in the generation process. We thus need to first apply the reverse step of bond flow on $\mathbf{z}_A$ to get bond tensor A . Then we fed it and $\mathbf{z}_X$ together into the reverse step of atom flow to obtain atom tensor X . Last, we put atom tensor and bond tensor into validity correction module and get the final generated molecular.

There are many constraints in chemistry to determine whether a molecule is valid or not. We use valency constraints to check the validity of our generated molecular. Let $|A_{ij}|$ denote the order of the chemical bond, we check the following valency constraint for the i^{th} and j^{th} atom as

$$\sum_j |A_{ij}| \leq \text{Valency}(X_i); \sum_i |A_{ij}| \leq \text{Valency}(X_j). \quad (13)$$

Different from the traditional reject-sampling category, we use a post-hoc validity correction method from [33]. The core idea is to always delete one order for the bond with the largest order until invalid molecular becomes valid and return its largest connected component. In detail, for an entered molecular graph G , we first check whether its largest connected component violates the valency constraints or not. If it is valid, then we output it as final molecular and terminate the algorithm. Otherwise we choose the bond with the largest order. If the bond is a triple bond or double bond, we delete one order and change it to a double bond or a single bond. If it is single bond, we drop it. Then we get a new graph G' . We repeat the above process for G' until we get a valid molecular or an atom since all bonds are dropped.

4 EXPERIMENTS

In this section, we conduct extensive experiments that evaluate the performance and time cost of molecular generation, the optimization of decoder to encoder, the smoothness of the learned latent space, and the constrained property optimization to show the effectiveness of GF-VAE.

4.1 Experiment Setting

Benchmark Datasets. We choose two commonly used chemical molecular datasets, QM9 [25] and ZINC-250K[13] for evaluation. QM9 contains 134k molecules with maximum 9 atoms in 4 different types, and ZINC-250K is made of 250k molecules with maximum 38 atoms in 9 different types. We apply the chemical libraries rdkit (<https://github.com/rdkit/rdkit>) to kekulize the molecules and remove hydrogen atoms from them. The resulting molecules contain

only single, double, and triple bonds. Then we encode each atom and bond by one-hot encoding, pad the molecule tensors which have less than the maximum number of atoms with a virtual atom, and add a virtual edge channel for adjacency tensor, representing no bonds between atoms. Thus, an adjacency tensor consists of $R = 4$ adjacency matrices stacked together, each corresponding to the existence of a bond type (single, double, triple, and virtual bonds) between the atoms. The feature matrix represents the type of each atom (e.g., oxygen, fluorine, etc). For the input of VAE-encoder part, we use adjacency tensor directly, for the input of flow-generator part, we dequantize the discrete one-hot-encoded data by adding uniform random noise $U[0, 0.6]$ for each dimension.

Baselines. We choose the following two groups of baselines:

a) VAE-based models: RVAE [20] and JT-VAE [14] are the recent state-of-the-art VAE-based baseline models. The RVAE model focuses on the matrix representation and formulates penalty terms that address validity constraints. The JT-VAE model captures the chemical validity by encoding and decoding a tree-structured scaffold of molecular graphs. We also choose three additional VAE models for comparison: CVAE [10], GVAE [18] and GraphVAE [30]. The first two models only learn from string (SMILES) information, while the third model is the first VAE model learning from molecular graphs.

b) Flow-based models: GraphAF [29] is the state-of-the-art flow-based model which generates molecular in a sequential way. MoFlow [33] is another state-of-the-art flow-based model which generates molecule in one-shot way. We also choose two more flow-based models: GraphNVP [21] and GRF [12]. GraphNVP is the first work to apply normalizing flow model to molecular generation task, and GRF uses residual flow to optimize GraphNVP. We do not compare GF-VAE with other sequential generative models such as GCPN [32], since our model generates molecular in one shot.

Evaluation Metrics. We evaluate the performance of our model on the generation and reconstruction of real molecules by the following widely-used metrics: **Validity (V)** - the percentage of chemically valid molecules among all generated molecules; **Novelty (N)** - the percentage of generated valid molecules not presenting in the training set; **Uniqueness (U)** - the percentage of unique valid molecules out of all generated molecules; and **Reconstruction (R)** - the percentage of the molecules that can be reconstructed from their own latent vectors. However, it is hard to compare models in two cases, one has a high *Novelty* but low *Uniqueness* (generating many new molecules, but most of them are the same) and the other has a high *Uniqueness* but low *Novelty* (generating different molecules, but most of them have already appeared in training set). Therefore, we adopt another metric **Score (S)**, which is the product of *Validity*, *Novelty* and *Uniqueness* in order to compromise these basic metrics.

Implementation Details. For the small dataset QM9, we adopt $L_B = 5, K_B = 1$ for the bond flow and $L_A = 13, K_A = 1$ for the atom flow. We use one 3*3 convolution layer with 128 hidden dimensions in bond coupling layer. For atom coupling layer, we set one relational graph convolution layer with 64 dimensions followed by a one-layer neural network with 64 hidden dimensions. For VAE-encoder, we set one layer with 64 dimensions for relational

Table 1: Generation & Reconstruction performance on QM9

	% V	% N	% U	% S	% R
CVAE	10.3	90.0	67.5	6.3	3.6
GVAE	60.2	80.9	9.3	4.5	96.0
GraphVAE	55.7	61.6	76.0	26.1	-
RVAE	96.6	97.5	-	-	61.8
GraphAF	100	88.8	94.5	83.9	100
GraphNVP	83.1 $\pm$ 0.5	58.2 $\pm$ 1.9	99.2 $\pm$ 1.3	48.0 $\pm$ 1.2	100
GRF	84.5 $\pm$ 0.7	58.6 $\pm$ 0.8	66.0 $\pm$ 1.2	32.7 $\pm$ 0.7	100
MoFlow	100	98.03 $\pm$ 0.14	99.20 $\pm$ 0.12	97.24 $\pm$ 0.21	100
GF-VAE	100	98.30$\pm$0.27	99.96$\pm$0.02	98.26$\pm$0.01	100

Table 2: Generation & Reconstruction performance on ZINC-250K

	% V	% N	% U	% S	% R
CVAE	0.7	100	67.5	0.5	44.6
GVAE	7.2	100	9.0	0.6	53.7
RVAE	34.9	100	-	-	54.7
JT-VAE	100	100	100	100	76.7
GraphAF	100	100	99.1	99.1	100
GraphNVP	42.6 $\pm$ 1.6	100	94.8 $\pm$ 0.6	40.4 $\pm$ 1.0	100
GRF	73.4 $\pm$ 0.6	100	53.7 $\pm$ 2.1	39.4 $\pm$ 1.3	100
MoFlow	100	100	99.99 $\pm$ 0.01	99.99 $\pm$ 0.01	100
GF-VAE	100	100	100	100	100

graph convolution layer and graph aggregation layer. For the bigger dataset ZINC-250K, we adopt $L_B = 5$, $K_B = 2$ and $L_A = 15$, $K_A = 2$ for the bond flow and atom flow, respectively. The bond coupling layer consists of two 3×3 convolution layers with 256 and 128 hidden dimensions respectively, and the atom coupling layer uses one relational graph convolution layer with 256 dimensions followed by a two-layer neural network with 256 and 64 hidden dimensions, respectively. For VAE-encoder, we set two layers with 256 and 128 dimensions for relational graph convolution layer and one layer with 128 dimensions for graph aggregation layer. For the compared baselines, we use their default parameter setting to reproduce their reported generation results.

4.2 Molecular Generation

Generation & Reconstruction Performance. We measure the performance of all evaluated models on 10,000 randomly generated molecules and report the mean and standard deviation of results over 5 runs in Table 1, and Table 2. Compared to VAE-based models, GF-VAE as well as all flow-based models achieve 100% reconstruction accuracy. At the same time, flow-based models share significantly high uniqueness ratio. Such advanced performance is attributed to the invertible characteristic of normalizing flows. Compared with the sequential flow-based model GraphAF, GF-VAE has improvements around 14% and 0.9% w.r.t. the scores for QM9 and ZINC-250K, respectively. In contrast with one-shot flow-based models, our model outperforms GraphNVP and GRF with a huge margin. As for MoFlow, although it shows almost equally competitive performance compared to our GF-VAE, its model parameters are more than two times of GF-VAE. We next deeply analyze the time cost of MoFlow and our GF-VAE model.

Time Cost Performance. We implemented our GF-VAE and MoFlow on a workstation with 1 GeForce RTX 2080 Ti GPU and trained

Table 3: Time cost of MoFlow & GF-VAE

	QM9	ZINC-250K
MoFlow	3.3 h 0.99 min/epoch	22 h 6.6 min/epoch
GF-VAE	2.3 h 0.68 min/epoch	8.2 h 2.45 min/epoch

Figure 4: Score (S) of GF-VAE and MoFlow at epoch 5 when varying L_A & L_B on QM9. The x-axis is the combination of different setting of (L_B, L_A)

Figure 5: Score (S) of GF-VAE and MoFlow at each epoch on ZINC-250K with different L_A & L_B (Model_ L_B L_A in legend)

the model by Adam optimizer with learning rate 0.001, batch size 256 and 200 epochs for two datasets. The results are shown in Table 3. Our GF-VAE spent 2.3 hours (0.68 minutes/epoch) for training on QM9 and 8.2 hours (2.45 minutes/epoch) on ZINC-250K. Compared to the time cost of MoFlow, which uses 3.3 hours (0.99 minutes/epoch) for QM9 and 22 hours (6.6 minutes/epoch) for ZINC-250K, our model is **1.4 and 2.7 times faster** on QM9 and ZINC-250K, respectively.

In order to better compare the time performance between MoFlow and our model, we conduct two additional experiments to make in-depth analysis.

Model Size vs Performance. GF-VAE is designed to reduce the model size in the flow-generator, by attaching a VAE-encoder. To evaluate this point, we fix K_A and K_B , and vary L_A and L_B to test the performance of lightweight/heavyweight MoFlow and GF-VAE.

The first experiment was conducted on QM9. We set K_B and K_A as 1, and choose L_A between one to two fold of L_B , which

Figure 6: Comparison of the encoder in p_T (optimized with a typical neural network decoder) and the same encoder in GF-VAE (optimized with a flow-based decoder). (a) and (b) are two randomly sampled molecules from QM9. (c) and (d) show the heatmap of atom-atom cosine similarity in graph (a) and (b), which are calculated based on the embedding vectors from p_T encoder. (e) and (f) show the similarity calculated based on the atom embedding vectors from the encoder of GF-VAE.

may capture a better result for small datasets from empirically prospective. Since this is a time-consuming process, we set the dimension of each layer to 16 and train 5 epochs. The result is shown in Figure 4. The horizontal axis represents the combination of L_B and L_A , and the vertical axis is the value of *Score*. The result shows that GF-VAE is always better than MoFlow when they have the same model size. If taking *Score* = 90% as an anchor value, our model accesses this score at the combination of $(L_B, L_A) = (6, 6)$, which costs 35.9 seconds/epoch. On contrast, MoFlow accesses this value at $(L_B, L_A) = (8, 14)$, which costs 56.1 seconds/epoch. That is to say, our model can achieve the same performance of MoFlow, but with a lighter set of weights and less time cost. This verifies the effectiveness of the design of GF-VAE on reducing the model size in flow-generator.

We apply ZINC-250K for the second experiment, setting K_B and K_A as 1, and conducting 3 different combinations of (L_B, L_A) , which are (1, 3), (2, 6) and (3, 9) separately. We train 100 epochs for each combination and take random sample 1,000 times at each epoch to calculate the *Score* at current model parameter. The result is shown in Figure 5, the horizontal axis represents training epoch(s) and the vertical axis is the value of *Score*. We can see at $(L_B, L_A) = (1, 3)$, we are much better than MoFlow since we can achieve 84.4% in

Score, but the maximum *Score* of MoFlow is 56.4%. Similarly for the left two combinations, we have a better performance especially at the first few epochs. Moreover, lightweight GF-VAE ($(L_B, L_A) = (2, 6)$) has better performance than heavier-weight MoFlow with $(L_B, L_A) = (3, 9)$. That is to say, our model can achieve better performance when we share the same parameter size with MoFlow. To a certain extent, this also shows the powerful ability of our model in exploring the latent space of molecules.

In summary, our GF-VAE not only memorizes and reconstructs all the source input molecular, but also achieves the highest generation score in a short time than existing models, which means we can generate more valid, novelty and unique molecular graphs very efficiently.

4.3 Optimization of Decoder to Encoder

The typical VAE-based models apply linear neural network as decoder, while our GF-VAE uses a flow-generator as a decoder. To evaluate the encoding capability learned with different decoders, we compare the embedding results of two example molecular graphs.

We implement a simple graph-VAE generative model p_T which uses the exactly same encoder of GF-VAE, but a three-layer linear neural network as its decoder. Both p_T and GF-VAE model are

trained with the same input by 100 epochs on QM9 dataset. Then we randomly sample two molecules to check their atom embedding vectors (output of Eq 1) learned by the two models. The pair-wise cosine similarity of two atoms in one graph is calculated and visualised in the heatmap of Figure 6. The results of $p\mathcal{T}$ are shown in Figure 6(c) and (d), while the results of GF-VAE are given in Figure 6(e) and (f) for the example molecular graphs in Figure 6(a) and (b), respectively.

For molecular (a), $p\mathcal{T}$ encodes {atom0, atom1, atom4, atom5} closely (cosine similarity=0.94~1), while encoding the other atoms closely to the same position (cosine similarity~1). Although {atom0, atom1, atom4, atom5} are connected on the graph, they should not be encoded compactly as a cluster. The other atoms in molecular (a) are not connected, and are different in types. It is not appropriate to encode them compactly. In contrast, GF-VAE encodes the atoms by preserving both their closeness on graph and their intrinsic feature. In Figure 6(e), we can see atom4-8 are encoded closely, while the other atoms are placed appropriately. For example, atom0 and atom7 are the furthest to the other atoms.

For molecular (b) including a ring, $p\mathcal{T}$ encodes atom2-8 compactly close (all similarities are about 1), while atom0 and atom1 are encoded close. In fact, the ring includes only atom1-6. The atom7-8 should not be encoded closely to the ring atoms. In addition, the learned similarities of atom2-5 are 1, which shows that $p\mathcal{T}$ cannot distinguish these different atoms. By contrast, GF-VAE encodes well the ring atoms, and maps atom7-8 as a compact cluster, as shown in Figure 6(f). However, the small differences between them are well recognized. Although they form a carbon-carbon triple bond, they have different neighbors.

To conclude, the encoder of GF-VAE learns more reasonable atom embedding than the encoder of $p\mathcal{T}$. Although the two encoders are trained under the exact same setting, they were optimized with different decoders. The flow-generator can optimize better the encoder than a typical neural network.

4.4 Smoothness of the Learned Latent Space

Guaranteeing the smoothness of the learned latent space is crucial in biological application scenarios. For instance, generating property desired molecular for instance could be realized by perturbing the latent representation of an original molecular graph. We qualitatively examine the smoothness of learned latent space in two ways. One is using grid search method in two random orthogonal directions to find neighborhoods of a randomly selected molecule in latent space and visualize them. The other is to interpolate between two latent points of molecular graphs. We choose Tanimoto index [1] as the chemical similarity metrics and indicate their similarity values by a heatmap.

Figure 7 (a) shows that our learned latent space is smooth because the neighboring latent points correspond to molecules with minor variations. Figure 7 (b) also exhibits that the interpolations between two latent points only slightly change the molecular graph except at the first step. This may be because the learned latent space does not follow the uniform distribution, and similar molecules clustered closely together, while dissimilar molecules loosely scattered around. It is worth noting that, unlike other VAE-based models, our model can complete this process through a single pass inverse transformation, because our decoding is deterministic. GVAE decodes

Figure 7: Smoothness of the learned latent space

each latent point 1000 times and selects the most common molecule as the representative one. This process for JT-VAE normally need dozens of times.

4.5 Molecule Optimization

Molecule optimization has two branches generally. One is property optimization, which generates novel molecules with the best property score. The other is constrained property optimization, which means to find molecules similar to a given molecule, but possessing a better chemical property. We solve these problems in three steps. 1) train a GF-VAE model and fix it; 2) train a Multilayer Perceptron (MLP) as a regressor on the latent space of molecules with target chemical property; and 3) search the molecules with the best property score by the gradient ascend method between the output and the input of the learned regressor. For constrained property optimization, we additionally need to add similarity constrain δ between the original molecule and the generated novel molecule. The optimization process fails when the similarity score is smaller than δ or the property score is not improved within limited steps.

We choose two widely used metrics as the target property: *Quantitative Estimate of Druglikeness* (QED) [2] and *penalized logP* (plogp) [9]. QED quantifies how likely a molecule is to be a potential drug, while plogp is defined as the penalized logarithm of the ratio of the concentrations between two solvents of a solute, which is a relative complex metric. Molecular similarity is measured by Tanimoto similarity of Morgan fingerprint [27].

Table 4: Optimized molecular with top-3 plogP and QED scores separately on ZINC-250K

Method	Penalized logP			QED		
	1st	2nd	3rd	1st	2nd	3rd
(In Dataset)	4.52	4.30	4.23	0.948	0.948	0.948
JT-VAE	5.30	4.93	4.49	0.925	0.911	0.910
GraphAF	12.23	11.29	11.05	0.948	0.948	0.947
MoFlow	4.46	4.24	4.05	0.948	0.948	0.948
GF-VAE	6.47	4.42	4.42	0.948	0.948	0.948

Figure 8: Parts of optimized molecules with top-3 QED scores on ZINC-250K

We conduct experiments of these two problems by sampling from the 800 molecules with the lowest property scores and summarize the results of optimized molecule in Table 4 and the constrained optimized molecule in Table 5.

Table 4 shows that GF-VAE and MoFlow find more molecules with the best QED values than the other two models. Moreover, GF-VAE can optimize more molecules to achieve a higher value. We show parts of top-3 results of optimized QED molecule by GF-VAE in Figure 8. Regarding the improvement on plogp, GraphAF obtained higher improvements than others, since it relies on reinforcement learning, which is more suitable for optimize complex issues. Although GF-VAE reached the second best performance, our chemical property regressor is simply based on MLP. Therefore, weaker performance compared to designed sophisticated approaches. Table 5 also shows that compared with the state-of-the-art VAE model JT-VAE, our GF-VAE achieves much higher similarity scores and property improvement, indicating that our model provides a more useful latent space to optimize molecules. Compared with MoFlow and GraphAF, our model achieves the best similarity scores.

Table 5: Constrained plogP optimization on ZINC-250k

δ	JT-VAE			GraphAF		
	Improvement	Similarity	Success	Improvement	Similarity	Success
0.0	1.91 $\pm$ 2.04	0.28 $\pm$ 0.15	97.5%	13.13 $\pm$ 6.89	0.29 $\pm$ 0.15	100%
0.2	1.68 $\pm$ 1.85	0.33 $\pm$ 0.13	97.1%	11.90 $\pm$ 6.86	0.33 $\pm$ 0.12	100%
0.4	0.84 $\pm$ 1.45	0.51 $\pm$ 0.10	83.6%	8.21 $\pm$ 6.51	0.49 $\pm$ 0.09	99.88%
0.6	0.21 $\pm$ 0.71	0.69 $\pm$ 0.06	46.4%	4.98 $\pm$ 6.49	0.66 $\pm$ 0.05	96.88%

δ	MoFlow			GF-VAE		
	Improvement	Similarity	Success	Improvement	Similarity	Success
0.0	8.61 $\pm$ 5.44	0.30 $\pm$ 0.20	98.88%	9.00 $\pm$ 6.12	0.11 $\pm$ 0.19	99.88%
0.2	7.06 $\pm$ 5.04	0.43 $\pm$ 0.20	96.75%	5.35 $\pm$ 5.62	0.57$\pm$0.24	89.50%
0.4	4.71 $\pm$ 4.55	0.61 $\pm$ 0.18	85.75%	3.86 $\pm$ 4.70	0.67$\pm$0.19	80.12%
0.6	2.10 $\pm$ 2.86	0.79 $\pm$ 0.14	58.25%	1.73 $\pm$ 2.69	0.81$\pm$0.14	56.00%

(a):constrained QED optimization on QM9 **(b):constrained plogp optimization on ZINC-250K**

Figure 9: Constrained property optimization. The arrow points from the original molecule to the optimized molecule. The values on the left and right side of the arrow denote the property improvement and similarity of the given molecule pairs, respectively.

We also show the top-5 results of constrained optimized molecules while maintaining a high similarity in Figure 9. The results were sorted by the product of property improvement and molecular similarity. We can see that the molecular pairs have very similar structures but a large improvement w.r.t. the target property.

5 CONCLUSION

In this paper, we propose GF-VAE, a flow-based VAE model for molecule graph generation. It equips VAE a lightweight flow model as its decoder. The model enables the encoder to speeds up the learning process of the decoder, and the decoder in turns to optimizes the representation ability of the encoder. GF-VAE inherits the advantages of both VAE and flow-based methods. Extensive results show that our model outperforms other state-of-the-art models on molecule generation and optimization, provides a smooth latent space and guarantees 100% reconstruction rate due to the flexibility of normalizing flow. Moreover, the training of GF-VAE is very efficient, which is around two times faster than the MoFlow model.

6 ACKNOWLEDGMENTS

This work is supported by the King Abdullah University of Science and Technology (KAUST), Saudi Arabia.

REFERENCES

- [1] Dávid Bajusz, Anita Rác, and Károly Héberger. 2015. Why is Tanimoto index an appropriate choice for fingerprint-based similarity calculations? *Journal of cheminformatics* 7, 1 (2015), 20.
- [2] G Richard Bickerton, Gaia V Paolini, Jérémy Besnard, Sorel Muresan, and Andrew L Hopkins. 2012. Quantifying the chemical beauty of drugs. *Nature chemistry* 4, 2 (2012), 90–98.
- [3] Esben Jannik Bjerrum and Boris Sattarov. 2018. Improving chemical autoencoder latent space and molecular de novo generation diversity with heteroencoders. *Biomolecules* 8, 4 (2018), 131.
- [4] Nathan Brown, Marco Fiscato, Marwin H.S. Segler, and Alain C. Vaucher. 2019. GuacaMol: Benchmarking Models for de Novo Molecular Design. *Journal of Chemical Information and Modeling* 59, 3 (2019), 1096–1108.
- [5] Bin Dai and David Wipf. 2018. Diagnosing and Enhancing VAE Models. In *International Conference on Learning Representations*.
- [6] Nicola De Cao and Thomas Kipf. 2018. MolGAN: An implicit generative model for small molecular graphs. *arXiv preprint arXiv:1805.11973* (2018).
- [7] Laurent Dinh, David Krueger, and Yoshua Bengio. 2014. Nice: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516* (2014).
- [8] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. 2016. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803* (2016).
- [9] Peter Ertl and Ansgar Schuffenhauer. 2009. Estimation of synthetic accessibility score of drug-like molecules based on molecular complexity and fragment contributions. *Journal of cheminformatics* 1, 1 (2009), 1–11.
- [10] 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. 2018. Automatic chemical design using a data-driven continuous representation of molecules. *ACS central science* 4, 2 (2018), 268–276.
- [11] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. *Advances in neural information processing systems* 27 (2014), 2672–2680.
- [12] Shion Honda, Hirotaka Akita, Katsuhiko Ishiguro, Toshiki Nakanishi, and Kenta Oono. 2019. Graph residual flow for molecular graph generation. *arXiv preprint arXiv:1909.13521* (2019).
- [13] John J Irwin, Teague Sterling, Michael M Mysinger, Erin S Bolstad, and Ryan G Coleman. 2012. ZINC: a free tool to discover chemistry for biology. *Journal of chemical information and modeling* 52, 7 (2012), 1757–1768.
- [14] Wengong Jin, Regina Barzilay, and Tommi Jaakkola. 2018. Junction tree variational autoencoder for molecular graph generation. *arXiv preprint arXiv:1802.04364* (2018).
- [15] Durk P Kingma and Prafulla Dhariwal. 2018. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in neural information processing systems*. 10215–10224.
- [16] Durk P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. 2016. Improved variational inference with inverse autoregressive flow. *Advances in neural information processing systems* 29 (2016), 4743–4751.
- [17] Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114* (2013).
- [18] Matt J Kusner, Brooks Paige, and José Miguel Hernández-Lobato. 2017. Grammar variational autoencoder. *arXiv preprint arXiv:1703.01925* (2017).
- [19] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. 2015. Gated graph sequence neural networks. *arXiv preprint arXiv:1511.05493* (2015).
- [20] Tengfei Ma, Jie Chen, and Cao Xiao. 2018. Constrained generation of semantically valid graphs via regularizing variational autoencoders. In *Advances in Neural Information Processing Systems*. 7113–7124.
- [21] Kaushalya Madhawa, Katsuhiko Ishiguro, Kosuke Nakago, and Motoki Abe. 2019. Graphnvp: An invertible flow model for generating molecular graphs. *arXiv preprint arXiv:1905.11600* (2019).
- [22] Aashish Manglik, Henry Lin, Dipendra K. Aryal, John D. McCorvy, Daniela Dengler, Gregory Corder, Anat Levit, Ralf C. Kling, Viachaslau Bernat, Harald Hübner, Xi-Ping Huang, Maria F. Sassano, Patrick M. Giguère, Stefan Löber, Da Duan, Grégory Scherrer, Brian K. Kobilka, Peter Gmeiner, Bryan L. Roth, and Brian K. Shoichet. 2016. Structure-based discovery of opioid analgesics with reduced side effects. *Nature* 537, 7619 (2016), 185–190.
- [23] Rogan Morrow and Wei-Chen Chiu. 2020. Variational autoencoders with normalizing flow decoders. *arXiv preprint arXiv:2004.05617* (2020).
- [24] Asher Mullard. 2017. The drug-maker’s guide to the galaxy. *Nature News* 549, 7673 (2017), 445.
- [25] Raghunathan Ramakrishnan, Pavlo O Dral, Matthias Rupp, and O Anatole Von Lilienfeld. 2014. Quantum chemistry structures and properties of 134 kilo molecules. *Scientific data* 1, 1 (2014), 1–7.
- [26] Danilo Jimenez Rezende and Shakir Mohamed. 2015. Variational inference with normalizing flows. *arXiv preprint arXiv:1505.05770* (2015).
- [27] David Rogers and Mathew Hahn. 2010. Extended-connectivity fingerprints. *Journal of chemical information and modeling* 50, 5 (2010), 742–754.
- [28] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *European Semantic Web Conference*. Springer, 593–607.
- [29] Chence Shi, Minkai Xu, Zhaocheng Zhu, Weinan Zhang, Ming Zhang, and Jian Tang. 2020. GraphAF: a flow-based autoregressive model for molecular graph generation. *arXiv preprint arXiv:2001.09382* (2020).
- [30] Martin Simonovsky and Nikos Komodakis. 2018. Graphvae: Towards generation of small graphs using variational autoencoders. In *International Conference on Artificial Neural Networks*. Springer, 412–422.
- [31] David Weininger, Arthur Weininger, and Joseph L. Weininger. 1989. SMILES. 2. Algorithm for generation of unique SMILES notation. *Journal of chemical information and computer sciences* 29, 2 (1989), 97–101.
- [32] Jiaxuan You, Bowen Liu, Zhitao Ying, Vijay Pande, and Jure Leskovec. 2018. Graph convolutional policy network for goal-directed molecular graph generation. In *NeurIPS*. 6410–6421.
- [33] Chengxi Zang and Fei Wang. 2020. MoFlow: an invertible flow model for generating molecular graphs. In *SIGKDD*. 617–626.