MoFlow: An Invertible Flow Model for Generating Molecular Graphs

Chengxi Zang
Department of Population Health Sciences, Weill Cornell Medicine
chz4001@med.cornell.edu

Fei Wang
Department of Population Health Sciences, Weill Cornell Medicine
few2001@med.cornell.edu

ABSTRACT

Generating molecular graphs with desired chemical properties driven by deep graph generative models provides a very promising way to accelerate drug discovery process. Such graph generative models usually consist of two steps: learning latent representations and generation of molecular graphs. However, to generate novel and chemically-valid molecular graphs from latent representations is very challenging because of the chemical constraints and combinatorial complexity of molecular graphs. In this paper, we propose MoFlow, a flow-based graph generative model to learn invertible mappings between molecular graphs and their latent representations. To generate molecular graphs, our MoFlow first generates bonds (edges) through a Glow based model, then generates atoms (nodes) given bonds by a novel graph conditional flow, and finally assembles them into a chemically valid molecular graph with a posthoc validity correction. Our MoFlow has merits including exact and tractable likelihood training, efficient one-pass embedding and generation, chemical validity guarantees, 100% reconstruction of training data, and good generalization ability. We validate our model by four tasks: molecular graph generation and reconstruction, visualization of the continuous latent space, property optimization, and constrained property optimization. Our MoFlow achieves state-of-the-art performance, which implies its potential efficiency and effectiveness to explore large chemical space for drug discovery.

CCS CONCEPTS

• **Mathematics of computing** → *Graph algorithms*; • **Theory of computation** → *Generating random combinatorial structures*; • **Computing methodologies** → *Unsupervised learning*; *Neural networks*; *Maximum likelihood modeling*;

KEYWORDS

Graph Generative Model; Graph Normalizing Flow; Graph Conditional Flow; Deep Generative Model; De novo Drug Design; Molecular Graph Generation; Molecular Graph Optimization;

ACM Reference Format:
Chengxi Zang and Fei Wang. 2020. MoFlow: An Invertible Flow Model for Generating Molecular Graphs. In *Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '20), August 23–27, 2020, Virtual Event, CA, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3394486.3403104

KDD '20, August 23–27, 2020, Virtual Event, CA, USA

ACM ISBN 978-1-4503-7998-4/20/08…$15.00
https://doi.org/10.1145/3394486.3403104

1 INTRODUCTION

Drug discovery aims at finding candidate molecules with desired chemical properties for clinical trials, which is a long (10-20 years) and costly ($0.5-$2.6 billion) process with a high failure rate [1, 25]. Recently, deep graph generative models have demonstrated their big potential to accelerate the drug discovery process by exploring large chemical space in a data-driven manner [12, 35]. These models usually first learn a continuous latent space by encoding[1] the training molecular graphs and then generate novel and optimized ones through decoding from the learned latent space guided by targeted properties [9, 12]. However, it is still very challenging to generate novel and chemically-valid molecular graphs with desired properties since: a) the scale of the chemical space of drug-like compounds is 10^{60} [22] but the scale of possibly generated molecular graphs by existing methods are much smaller, and b) generating molecular graphs that have both multi-type nodes and edges and follow bond-valence constraints is a hard combinatorial task.

Prior works leverage different deep generative frameworks for generating molecular SMILES codes [33] or molecular graphs, including variational autoencoder (VAE)-based models [4, 5, 12, 16, 19, 20, 31], generative adversarial networks (GAN)-based models [6, 34], and autoregressive (AR)-based models [26, 34]. In this paper, we explore a different deep generative framework, namely the normalizing flow [7, 14, 21] to generate molecular graphs. Compared with above three frameworks, the flow-based models are the only one which can memorize and exactly reconstruct all the input data, and at the same time have the potential to generate more novel, unique and valid molecules, which implies its potential capability of deeper exploration of the huge chemical space. To our best knowledge, there have been three flow-based models proposed for molecular graph generation. The GraphAF [30] model is an autoregressive flow-based model that achieves state-of-the-art performance in molecular graph generation. GraphAF generates molecules in a sequential manner by adding each new atom or bond followed by a validity check. GraphNVP [21] and GRF [10] are proposed for molecular graph generation in a one-shot manner. However, they cannot guarantee chemical validity and thus show poor performance in generating valid and novel molecules.

In this paper, we propose a novel deep graph generative model named MoFlow to generate molecular graphs. Our MoFlow is the

[1] In this paper, we use inference, embedding or encoding interchangeably to refer to the transformation from molecular graphs to the learned latent space, and we use decoding or generation for the reverse transformation.

first of its kind which not only generates molecular graphs efficiently by invertible mapping at one shot, but also has a chemical validity guarantee. More specifically, to capture the combinatorial atom-and-bond structures of molecular graphs, we propose a variant of the Glow model [14] to generate bonds (multi-type edges, e.g., single, double and triple bonds), a novel *graph conditional flow* to generate atoms (multi-type nodes, e.g. C, N etc.) given bonds by leveraging graph convolutions, and finally assemble atoms and bonds into a valid molecular graph which follows bond-valence constraints. We illustrate our modelling framework in Figure 1. Our MoFlow is trained by exact and tractable likelihood estimation, and one-pass inference and generation can be efficiently utilized for molecular graph optimization.

We validate our MoFlow through a wide range of experiments from molecular graph generation, reconstruction, visualization to optimization. As baselines, we compare the state-of-the-art VAE-based model [12], autoregressive-based models [26, 34], and all three flow-based models [10, 21, 30]. As for memorizing input data, MoFlow achieves 100% reconstruction rate. As for exploring the unknown chemical space, MoFlow outperforms above models by generating more novel, unique and valid molecules (as demonstrated by the N.U.V. scores in Table 2 and 3). MoFlow generates 100% chemically-valid molecules when sampling from prior distributions. Furthermore, if without validity correction, MoFlow still generates much more valid molecules than existing models (validity-without-check scores in Table 2 and 3). For example, the state-of-the-art autoregressive-flow-based model GraphAF [30] achieves 67% and 68% validity-without-check scores for two datasets while MoFlow achieves 96% and 82% respectively, thanks to its capability of capturing the chemical structures in a holistic way. As for chemical property optimization, MoFlow can find much more novel molecules with top drug-likeness scores than existing models (Table 4 and Figure 5). As for constrained property optimization, MoFlow finds novel and optimized molecules with the best similarity scores and second best property improvement (Table 5).

It is worthwhile to highlight our contributions as follows:

- **Novel MoFlow model:** our MoFlow is one of the first flow-based graph generative models which not only generates molecular graphs at one shot by invertible mapping but also has a validity guarantee. To capture the combinatorial atom-and-bond structures of molecular graphs, we propose a variant of Glow model for bonds (edges) and a novel graph conditional flow for atoms (nodes) given bonds, and then assemble them into valid molecular graphs.
- **State-of-the-art performance:** our MoFlow achieves many state-of-the-art results w.r.t. molecular graph generation, reconstruction, optimization, etc., and at the same time our one-shot inference and generation are very efficient, which implies its potentials in deep exploration of huge chemical space for drug discovery.

The outline of this paper is: survey (Sec. 2), proposed method (Sec. 3 and 4), experiments (Sec. 5), and conclusions (Sec. 6). In order to promote reproducibility, our codes and datasets are open-sourced at https://github.com/calvin-zcx/moflow.

2 RELATED WORK

Molecular Generation. Different deep generative frameworks are proposed for generating molecular SMILES or molecular graphs. Among the variational autoencoder (VAE)-based models [4, 5, 12, 16, 19, 20, 31], the JT-VAE [12] generates valid tree-structured molecules by first generating a tree-structured scaffold of chemical substructures and then assembling substructures according to the generated scaffold. The MolGAN [6] is a generative adversarial networks (GAN)-based model but shows very limited performance in generating valid and unique molecules. The autoregressive-based models generate molecules in a sequential manner with validity check at each generation step. For example, the MolecularRNN [26] sequentially generates each character of SMILES and the GCPN [34] sequentially generates each atom/bond in a molecular graphs. In this paper, we explore a different deep generative framework, namely the normalizing flow models [7, 14, 21], for molecular graph generation, which have the potential to memorize and reconstruct all the training data and generalize to generating more valid, novel and unique molecules.

Flow-based Models. The (normalizing) flow-based models try to learn mappings between complex distributions and simple prior distributions through invertible neural networks and such a framework has good merits of exact and tractable likelihood estimation for training, efficient one-pass inference and sampling, invertible mapping and thus reconstructing all the training data etc. Examples include NICE[7], RealNVP[8], Glow[14] and GNF [18] which show promising results in generating images or even graphs [18]. See latest reviews in [15, 23] and more technical details in Section 3.

To our best knowledge, there are three flow-based models for molecular graph generation. The GraphAF [30] is an autoregressive flow-based model which achieves state-of-the-art performance in molecular graph generation. The GraphAF generates molecular graphs in a sequential manner with validity check when adding any new atom or bond. The GraphNVP [21] and GRF [10] are proposed for molecular graph generation in a one-shot manner. However, they have no guarantee for chemical validity and thus show very limited performance in generating valid and novel molecular graphs. Our MoFlow is the first of its kind which not only generates molecular graphs efficiently by invertible mapping at one shot but also has a validity guarantee. In order to capture the atom-and-bond composition of molecules, we propose a variant of Glow[14] model for bonds and a novel graph conditional flow for atoms given bonds, and then combining them with a post-hoc validity correction. Our MoFlow achieves many state-of-the-art results thanks to capturing the chemical structures in a holistic way, and our one-shot inference and generation are more efficient than sequential models.

3 MODEL PRELIMINARY

The flow framework. The flow-based models aim to learn a sequence of invertible transformations $f_\Theta = f_L \circ ... \circ f_1$ between complex high-dimensional data $X \sim P_\mathcal{X}(X)$ and $Z \sim P_\mathcal{Z}(Z)$ in a latent space with the same number of dimensions where the latent distribution $P_\mathcal{Z}(Z)$ is easy to model (e.g., strong independence assumptions hold in such a latent space). The potentially complex data in the original space can be modelled by the **change of variable**

formula where $Z = f_\Theta(X)$ and:

$$P_\mathcal{X}(X) = P_\mathcal{Z}(Z) \mid \det(\frac{\partial Z}{\partial X}) \mid . \tag{1}$$

To sample $\tilde{X} \sim P_\mathcal{X}(X)$ is achieved by sampling $\tilde{Z} \sim P_\mathcal{Z}(Z)$ and then to transform $\tilde{X} = f_\Theta^{-1}(\tilde{Z})$ by the reverse mapping of f_Θ.

Let $Z = f_\Theta(X) = f_L \circ ... \circ f_1(X)$, $H_l = f_l(H_{l-1})$ where f_l ($l = 1, ...L \in \mathbb{N}^+$) are invertible mappings, $H_0 = X$, $H_L = Z$ and $P_\mathcal{Z}(Z)$ follows a standard isotropic Gaussian with independent dimensions. Then we get the log-likelihood of X by the change of variable formula as follows:

$$\begin{aligned} \log P_\mathcal{X}(X) &= \log P_\mathcal{Z}(Z) + \log \mid \det(\frac{\partial Z}{\partial X}) \mid \\ &= \sum_i \log P_{\mathcal{Z}_i}(Z_i) + \sum_{l=1}^{L} \log \mid \det(\frac{\partial f_l}{\partial H_{l-1}}) \mid \end{aligned} \tag{2}$$

where $P_{\mathcal{Z}_i}(Z_i)$ is the probability of the i^{th} dimension of Z and $f_\Theta = f_L \circ ... \circ f_1$ is an invertible deep neural network to be learnt. Thus, the exact-likelihood-based training is tractable.

Invertible affine coupling layers. However, how to design a.) an invertible function f_Θ with b.) expressive structures and c.) efficient computation of the Jacobian determinant are nontrivial. The NICE[7] and RealNVP [8] design an affine coupling transformation $Z = f_\Theta(X) : \mathbb{R}^n \mapsto \mathbb{R}^n$:

$$\begin{aligned} Z_{1:d} &= X_{1:d} \\ Z_{d+1:n} &= X_{d+1:n} \odot e^{S_\Theta(X_{1:d})} + T_\Theta(X_{1:d}), \end{aligned} \tag{3}$$

by splitting X into two partitions $X = (X_{1:d}, X_{d+1:n})$. Thus, a.) the invertibility is guaranteed by:

$$\begin{aligned} X_{1:d} &= Z_{1:d} \\ X_{d+1:n} &= (Z_{d+1:n} - T_\Theta(Z_{1:d}))/e^{S_\Theta(Z_{1:d})}, \end{aligned} \tag{4}$$

b.) the expressive power depends on arbitrary neural structures of the **Scale function** $S_\Theta : \mathbb{R}^d \mapsto \mathbb{R}^{n-d}$ and the **Transformation function** $T_\Theta : \mathbb{R}^d \mapsto \mathbb{R}^{n-d}$ in the affine transformation of $X_{d+1:n}$, and c.) the Jacobian determiant can be computed efficiently by $\det(\frac{\partial Z}{\partial X}) = \exp(\sum_j S_\Theta(X_{1:d})_j)$.

Splitting Dimensions. The flow-based models, e.g., RealNVP [8] and Glow [14], adopt **squeeze operation** which compresses the spatial dimension $X^{c\times n\times n}$ into $X^{(ch^2)\times \frac{n}{h}\times \frac{n}{h}}$ to make more channels and then split channels into two halves for the coupling layer. A deep flow model at a specific layer transforms unchanged dimensions in the previous layer to keep all the dimensions transformed. In order to learn an optimal partition of X, Glow [14] model introduces an **invertible** 1×1 **convolution** : $\mathbb{R}^{c\times n\times n} \times \mathbb{R}^{c\times c} \mapsto \mathbb{R}^{c\times n\times n}$ with learnable convolution kernel $W \in \mathbb{R}^{c\times c}$ which is initialized as a random rotation matrix. After the transformation Y = invertible 1×1 convolution(X, W), a fixed partition $Y = (Y_{1:\frac{c}{2},:,:}, Y_{\frac{c}{2}+1:n,:,:})$ over the channel c is used for the affine coupling layers.

Numerical stability by actnorm. In order to ensure the numerical stability of the flow-based models, **actnorm layer** is introduced in Glow [14] which normalizes dimensions in each channel over a batch by an affine transformation with learnable scale and bias. The scale and the bias are initialized as the mean and the inverse of the standard variation of the dimensions in each channel over the batch.

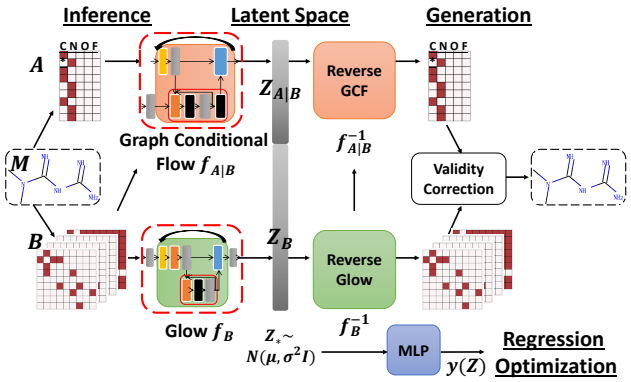

Figure 1: The outline of our MoFlow. A molecular graph M (e.g. Metformin) is represented by a feature matrix A for atoms and adjacency tensors B for bonds. Inference: the graph conditional flow (GCF) $f_{\mathcal{A}|\mathcal{B}}$ for atoms (Sec. 4.2) transforms A given B into conditional latent vector $Z_{A|B}$, and the Glow $f_\mathcal{B}$ for bonds (Sec. 4.3) transform B into latent vector Z_B. The latent space follows a spherical Gaussian distribution. Generation: the generation process is the reverse transformations of previous operations, followed by a validity correction (Sec. 4.4) procedure which ensures the chemical validity. We summarize MoFlow in Sec. 4.5. Regression and optimization: the mapping $y(Z)$ between latent space and molecular properties are used for molecular graph optimization and property prediction (Sec. 5.3, Sec. 5.4).

4 PROPOSED MOFLOW MODEL

In this section, we first define the problem and then introduce our Molecular Flow (MoFlow) model in detail. We show the outline of our MoFlow in Figure 1 as a roadmap for this section.

4.1 Problem Definition: Learning a Probability Model of Molecular Graphs

Let $\mathcal{M} = \mathcal{A} \times \mathcal{B} \subset \mathbb{R}^{n\times k} \times \mathbb{R}^{c\times n\times n}$ denote the set of $\mathcal{M}$olecules which is the Cartesian product of the $\mathcal{A}$tom set $\mathcal{A}$ with at most $n \in \mathbb{N}^+$ atoms belonging to $k \in \mathbb{N}^+$ atom types and the $\mathcal{B}$ond set $\mathcal{B}$ with $c \in \mathbb{N}^+$ bond types. A molecule $M = (A, B) \in \mathcal{A} \times \mathcal{B}$ is a pair of an atom matrix $A \in \mathbb{R}^{n\times k}$ and a bond tensor $B \in \mathbb{R}^{c\times n\times n}$. We use one-hot encoding for the empirical molecule data where $A(i, k) = 1$ represents the atom i has atom type k, and $B(c, i, j) = B(c, j, i) = 1$ represents a type c bond between atom i and atom j. Thus, a molecule M can be viewed as an undirected graph with multi-type nodes and multi-type edges.

Our primary goal is to learn a molecule generative model $P_\mathcal{M}(M)$ which is the probability of sampling any molecule M from $P_\mathcal{M}$. In order to capture the combinatorial atom-and-bond structures of molecular graphs, we decompose the $P_\mathcal{M}(M)$ into two parts:

$$P_\mathcal{M}(M) = P_\mathcal{M}((A, B)) \approx P_{\mathcal{A}|\mathcal{B}}(A|B; \theta_{\mathcal{A}|\mathcal{B}}) P_\mathcal{B}(B; \theta_\mathcal{B}) \tag{5}$$

where $P_\mathcal{M}$ is the distribution of molecular graphs, $P_\mathcal{B}$ is the distribution of bonds (edges) in analogy to modelling multi-channel images , and $P_{\mathcal{A}|\mathcal{B}}$ is the conditional distribution of atoms (nodes) given the bonds by leveraging graph convolution operations. The $\theta_\mathcal{B}$ and $\theta_{\mathcal{A}|\mathcal{B}}$ are learnable modelling parameters. In contrast with VAE

or GAN based frameworks, we can learn the parameters by exact maximum likelihood estimation (MLE) framework by maximizing:

$$\arg\max_{\theta_{\mathcal{B}}, \theta_{\mathcal{A}|\mathcal{B}}} \mathbb{E}_{M=(A,B)\sim P_{\mathcal{M}-data}}[\log P_{\mathcal{A}|\mathcal{B}}(A|B;\theta_{\mathcal{A}|\mathcal{B}}) + \log P_{\mathcal{B}}(B;\theta_{\mathcal{B}})] \quad (6)$$

Our model thus consists of two parts, namely a *graph conditional flow for atoms* to learn the atom matrix conditional on the bond tensors and a *flow for bonds* to learn bond tensors. We further learn a mapping between the learned latent vectors and molecular properties to regress the graph-based molecular properties, and to guide the generation of optimized molecular graphs.

4.2 Graph Conditional Flow for Atoms

Given a bond tensor $B \in \mathcal{B} \subset \mathbb{R}^{c\times n\times n}$, our goal of the atom flow is to generate the right atom-type matrix $A \in \mathcal{A} \subset \mathbb{R}^{n\times k}$ to assemble valid molecules $M = (A, B) \in \mathcal{M} \subset \mathbb{R}^{n\times k + c\times n\times n}$. We first define **$\mathcal{B}$-conditional flow** and **graph conditional flow** $f_{\mathcal{A}|\mathcal{B}}$ to transform A given B into conditional latent variable $Z_{A|B} = f_{\mathcal{A}|\mathcal{B}}(A|B)$ which follows isotropic Gaussian $P_{\mathcal{Z}_{\mathcal{A}|\mathcal{B}}}$. We can get the conditional probability of atom features given the bond graphs $P_{\mathcal{A}|\mathcal{B}}$ by a conditional version of the change of variable formula.

4.2.1 *$\mathcal{B}$-Conditional Flow and Graph Conditional Flow.*

Definition 4.1. $\mathcal{B}$-conditional flow: A **$\mathcal{B}$-conditional flow** $Z_{A|B}|B = f_{\mathcal{A}|\mathcal{B}}(A|B)$ is an invertible and dimension-kept mapping and there exists reverse transformation $f^{-1}_{\mathcal{A}|\mathcal{B}}(Z_{A|B}|B) = A|B$ where $f_{\mathcal{A}|\mathcal{B}}$ and $f^{-1}_{\mathcal{A}|\mathcal{B}} : \mathcal{A}\times\mathcal{B} \mapsto \mathcal{A}\times\mathcal{B}$.

The condition $B \in \mathcal{B}$ keeps fixed during the transformation. Under the independent assumption of $\mathcal{A}$ and $\mathcal{B}$, the Jacobian of $f_{\mathcal{A}|\mathcal{B}}$ is:

$$\frac{\partial f_{\mathcal{A}|\mathcal{B}}}{\partial(A,B)} = \begin{bmatrix} \frac{\partial f_{\mathcal{A}|\mathcal{B}}}{\partial A} & \frac{\partial f_{\mathcal{A}|\mathcal{B}}}{\partial B} \\ 0 & \mathbb{1}_B \end{bmatrix}, \quad (7)$$

the determiant of this Jacobian is $\det \frac{\partial f_{\mathcal{A}|\mathcal{B}}}{\partial(A,B)} = \det \frac{\partial f_{\mathcal{A}|\mathcal{B}}}{\partial A}$, and thus the *conditional version of the change of variable formula* in the form of log-likelihood is:

$$\log P_{\mathcal{A}|\mathcal{B}}(A|B) = \log P_{\mathcal{Z}_{\mathcal{A}|\mathcal{B}}}(Z_{A|B}) + \log \mid \det \frac{\partial f_{\mathcal{A}|\mathcal{B}}}{\partial A} \mid . \quad (8)$$

Definition 4.2. Graph conditional flow: A graph conditional flow is a $\mathcal{B}$-conditional flow $Z_{A|B}|B = f_{\mathcal{A}|\mathcal{B}}(A|B)$ where $B \in \mathcal{B} \subset \mathbb{R}^{c\times n\times n}$ is the adjacency tenor for edges with c types and $A \in \mathcal{A} \subset \mathbb{R}^{n\times k}$ is the feature matrix of the corresponding n nodes.

4.2.2 *Graph coupling layer.*

We construct aforementioned invertible mapping $f_{\mathcal{A}|\mathcal{B}}$ and $f^{-1}_{\mathcal{A}|\mathcal{B}}$ by the scheme of the affine coupling layer. Different from traditional affine coupling layer, our coupling transformation relies on graph convolution [32] and thus we name such a coupling transformation as a **graph coupling layer**.

For each graph coupling layer, we split input $A \in \mathbb{R}^{n\times k}$ into two parts $A = (A_1, A_2)$ along the n row dimension, and we get the output $Z_{A|B} = (Z_{A_1|B}, Z_{A_2|B}) = f_{\mathcal{A}|\mathcal{B}}(A|B)$ as follows:

$$\begin{aligned} Z_{A_1|B} &= A_1 \\ Z_{A_2|B} &= A_2 \odot \text{Sigmoid}(S_\Theta(A_1|B)) + T_\Theta(A_1|B) \end{aligned} \quad (9)$$

where $\odot$ is the element-wise product. We deign the scale function S_Θ and the transformation function T_Θ in each graph coupling layer by incorporating **graph convolution** structures. The bond tensor $B \in \mathbb{R}^{c\times n\times n}$ keeps a fixed value during transforming the atom matrix A. We also apply the masked convolution idea in [8] to the graph convolution in the graph coupling layer. Here, we adopt Relational Graph Convolutional Networks (R-GCN) [29] to build graph convolution layer **graphconv** as follows:

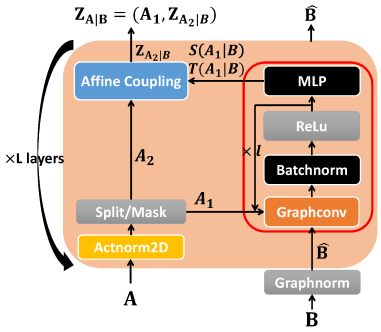

Figure 2: Graph conditional flow $f_{\mathcal{A}|\mathcal{B}}$ for the atom matrix. We show the details of one invertible graph coupling layer and a multiscale structure consists of a cascade of L layers of such graph coupling layer. The graphnorm is computed only once.

$$\text{graphconv}(A_1) = \sum_{i=1}^{c} \hat{B}_i(M \odot A)W_i + (M \odot A)W_0 \quad (10)$$

where $\hat{B}_i = D^{-1}B_i$ is the normalized adjacency matrix at channel i, $D = \sum_{c,i} B_{c,i,j}$ is the sum of the in-degree over all the channels for each node, and $M \in \{0,1\}^{n\times k}$ is a binary mask to select a partition A_1 from A. Because the bond graph is fixed during graph coupling layer and thus the graph normalization, denoted as **graphnorm**, is computed only once.

We use multiple stacked graphconv->BatchNorm1d->ReLu layers with a multi-layer perceptron (MLP) output layer to build the graph scale function S_Θ and the graph transformation function T_Θ. What's more, instead of using exponential function for the S_Θ as discussed in Sec. 3, we adopt Sigmoid function for the sake of the numerical stability of cascading multiple flow layers. The reverse mapping of the graph coupling layer $f^{-1}_{\mathcal{A}|\mathcal{B}}$ is:

$$\begin{aligned} A_1 &= Z_{A_1|B} \\ A_2 &= (Z_{A_2|B} - T_\Theta(Z_{A_1|B}|B))/\text{Sigmoid}(S_\Theta(Z_{A_1|B}|B)). \end{aligned} \quad (11)$$

The logarithm of the Jacobian determiant of each graph coupling layer can be efficiently computed by:

$$\log \mid \det(\frac{\partial f_{\mathcal{A}|\mathcal{B}}}{\partial A}) \mid = \sum_j \log \text{Sigmoid}(S_\Theta(A_1|B))_j \quad (12)$$

where j iterates each element. In principle, we can use arbitrary complex graph convolution structures for S_Θ and T_Θ since the computing of above Jacobian determinant of $f_{\mathcal{A}|\mathcal{B}}$ does not involve in computing the Jacobian of S_Θ or T_Θ.

4.2.3 *Actnorm for 2-dimensional matrix.*

For the sake of numerical stability, we design a variant of invertible actnorm layer [14] for the 2-dimensional atom matrix, denoted as **actnorm2D** (activation normalization for 2D matrix), to normalize each row, namely the feature dimension for each node, over a batch of 2-dimensional atom matrices. Given the mean $\mu \in \mathbb{R}^{n\times 1}$ and the standard deviation

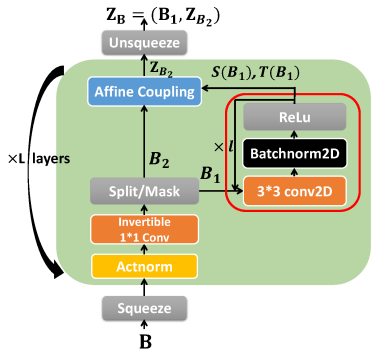

Figure 3: A variant of Glow $f_{\mathcal{B}}$ for bonds' adjacency tensors.

$\sigma^2 \in \mathbb{R}^{n\times 1}$ for each row dimension, the normalized input follows $\hat{A} = \frac{A-\mu}{\sqrt{\sigma^2+\epsilon}}$ where ϵ is a small constant, the reverse transformation is $A = \hat{A} * \sqrt{\sigma^2+\epsilon} + \mu$, and the logarithmic Jacobian determiant is:

$$\log \mid \det \frac{\partial \textbf{actnorm2D}}{\partial X} \mid = \frac{k}{2} \sum_{i}^{n} \mid \log(\sigma_i^2 + \epsilon) \mid \tag{13}$$

4.2.4 Deep architectures. We summarize our deep graph conditional flow in Figure 2. We stack multiple graph coupling layers to form graph conditional flow. We alternate different partition of $A = (A_1, A_2)$ in each layer to transform the unchanged part of the previous layer.

4.3 Glow for Bonds

The bond flow aims to learn an invertible mapping $f_{\mathcal{B}} : \mathcal{B} \subset \mathbb{R}^{c\times n\times n} \mapsto \mathcal{B} \subset \mathbb{R}^{c\times n\times n}$ where the transformed latent variable $Z_B = f_{\mathcal{B}}(B)$ follows isotropic Gaussian. According to the change of variable formula, we can get the logarithmic probability of bonds by $\log P_{\mathcal{B}}(B) = \log P_{\mathcal{Z}_{\mathcal{B}}}(Z_B) + \log \mid \det(\frac{\partial f_{\mathcal{B}}}{\partial B}) \mid$ and generating bond tensor by reversing the mapping $\tilde{B} = f_{\mathcal{B}}^{-1}(\tilde{Z})$ where $\tilde{Z} \sim P_{\mathcal{Z}}(Z)$. We can use arbitrary flow model for the bond tensor and we build our bond flow $f_{\mathcal{B}}$ based on a variant of Glow [14] framework.

We also follow the scheme of affine coupling layer to build invertible mappings. For each affine coupling layer, We split input $B \in \mathbb{R}^{c\times n\times n}$ into two parts $B = (B_1, B_2)$ along the channel c dimension, and we get the output $Z_B = (Z_{B_1}, Z_{B_2})$ as follows:

$$\begin{aligned} Z_{B_1} &= B_1 \\ Z_{B_2} &= B_2 \odot \text{Sigmoid}(S_\Theta(B_1)) + T_\Theta(B_1). \end{aligned} \tag{14}$$

And thus the reverse mapping $f_{\mathcal{B}}^{-1}$ is:

$$\begin{aligned} B_1 &= Z_{B_1} \\ B_2 &= (Z_{B_2} - T_\Theta(Z_{B_1}))/\text{Sigmoid}(S_\Theta(Z_{B_1})). \end{aligned} \tag{15}$$

Instead of using exponential function as scale function, we use the Sigmoid function with range $(0, 1)$ to ensure the numerical stability when stacking many layers. We find that exponential scale function leads to a large reconstruction error when the number of affine coupling layers increases. The scale function S_Θ and the transformation function T_Θ in each affine coupling layer can have arbitrary structures. We use multiple 3×3 conv2d->BatchNorm2d->ReLu layers to build them. The logarithm of the Jacobian determiant of each affine coupling is

$$\log \mid \det(\frac{\partial Z_B}{\partial B}) \mid = \sum_j \log \text{Sigmoid}(S_\Theta(B_1))_j. \tag{16}$$

In order to learn optimal partition and ensure model's stability and learning rate, we also use the invertible 1×1 convolution layer and actnorm layer adopted in the Glow. In order to get more channels for masking and transformation, we **squeeze** the spatial size of B from $\mathbb{R}^{c\times n\times n}$ to $\mathbb{R}^{(c*h*h)\times \frac{n}{h}\times \frac{n}{h}}$ by a factor h and apply the affine coupling transformation to the squeezed data. The reverse **unsqueeze** operation is adopted to the output. We summarize our bond flow in Figure 3.

4.4 Validity Correction

Molecules must follow the valency constraints for each atom, but assembling a molecule from generated bond tensor and atom matrix may lead to chemically invalid ones. Here we define the valency constraint for the i^{th} atom as:

$$\sum_{c,j} c \times B(c, i, j) \leq \text{Valency}(\text{Atom}_i) + Ch \tag{17}$$

where $B \in \{0, 1\}^{c\times n\times n}$ is the one-hot bond tensor over $c \in \{1, 2, 3\}$ order of chemical bonds (single, double, triple) and $Ch \in \mathbb{N}$ represents the formal charge. Different from existing valency constraints defined in [26, 34], we consider the effect of formal charge which may introduce extra bonds for the charged atoms. For example, ammonium $[NH4]^+$ may have 4 bonds for N instead of 3. Similarly, S^+ and O^+ may have 3 bonds instead of 2. Here we only consider $Ch = 1$ for N^+, S^+ and O^+ and make $Ch = 0$ for other atoms.

In contrast with the existing reject-sampling-based validity check adopted in the autoregressive models [26, 34], we introduce a new post-hoc validity correction procedure after generating a molecule M at once: 1) check the valency constraints of M; 2) if all the atoms of M follows valecny constraints, we return the largest connected component of the molecule M and end the procedure; 3) if there exists an invalid atom i, namely $\sum_{c,j} c \times B(c, i, j) > \text{Valency}(\text{Atom}_i) + Ch$, we sort the bonds of i by their order and delete 1 order for the bond with the largest order; 4) go to step 1). Our validity correction procedure tries to make a minimum change to the existing molecule and to keep the largest connected component as large as possible.

4.5 Inference and Generation

We summarize the inference (encoding) and generation (decoding) of molecular graphs by our MoFlow in Algorithm 1 and Algorithm 2 respectively. We visualize the overall framework in Figure 1. As shown in the algorithms, our MoFlow have merits of exact likelihood estimation/training, one-pass inference, invertible and one-pass generation, and chemical validity guarantee.

5 EXPERIMENTS

Following previous works [12, 30], we validate our MoFlow by answering following questions:

- **Molecular graph generation and reconstruction (Sec. 5.1):** Can our MoFlow memorize and reconstruct all the training molecule datasets? Can our MoFlow generalize to generate novel, unique and valid molecules as many as possible?
- **Visualizing continuous latent space (Sec. 5.2):** Can our MoFlow embed molecular graphs into continuous latent space with reasonable chemical similarity?
- **Property optimization (Sec. 5.3):** Can our MoFlow generate novel molecular graphs with optimized properties?

Algorithm 1: Exact Likelihood Inference (Encoding) of Molecular Graphs by MoFlow

Input: $f_{\mathcal{A}|\mathcal{B}}$: graph conditional flow for atoms, $f_{\mathcal{B}}$: glow for bonds, A: atom matrix, B: bond tensor, $P_{\mathcal{Z}_*}$: isotropic Gaussian distributions.
Output: Z_M:latent representation for atom M, $\log P_{\mathcal{M}}(M)$: logarithmic likelihood of molecule M.

$Z_B = f_{\mathcal{B}}(B)$
$\log P_{\mathcal{B}}(B) = \log P_{\mathcal{Z}_{\mathcal{B}}}(Z_B) + \log \mid \det(\frac{\partial f_{\mathcal{B}}}{\partial B}) \mid$
$\hat{B}$ = graphnorm(B)
$Z_{A|B} = f_{\mathcal{A}|\mathcal{B}}(A|\hat{B})$
$\log P_{\mathcal{A}|\mathcal{B}}(A|B) = \log P_{\mathcal{Z}_{\mathcal{A}|\mathcal{B}}}(Z_{A|B}) + \log \mid \det(\frac{\partial f_{\mathcal{A}|\mathcal{B}}}{\partial A}) \mid$
$Z_M = (Z_{A|B}, Z_B)$
$\log P_{\mathcal{M}}(M) = \log P_{\mathcal{B}}(B) + \log P_{\mathcal{A}|\mathcal{B}}(A|B)$
Return: $Z_M, \log P_{\mathcal{M}}(M)$

Algorithm 2: Molecular Graph Generation (Decoding) by the Reverse Transformation of MoFlow

Input: $f_{\mathcal{A}|\mathcal{B}}$: graph conditional flow for atoms, $f_{\mathcal{B}}$: glow for bonds, Z_M:latent representation of molecule M or sampling from a prior Gaussian, validity-correction: validity correction rules.
Output: M: a molecule

$(Z_{A|B}, Z_B) = Z_M$
$B = f_{\mathcal{B}}^{-1}(Z_B)$
$\hat{B}$ = graphnorm(B)
$A = f_{\mathcal{A}|\mathcal{B}}^{-1}(Z_{A|B}|\hat{B})$
M = validity-correction(A, B)
Return: M

- **Constrained property optimization (Sec. 5.4):** Can our MoFlow generate novel molecular graphs with the optimized properties and at the same time keep the chemical similarity as much as possible?

Baselines. We compare our MoFlow with: a) the state-of-the-art VAE-based method JT-VAE [12] which captures the chemical validity by encoding and decoding a tree-structured scaffold of molecular graphs; b) the state-of-the-art autoregressive models GCPN [34] and MolecularRNN (MRNN)[26] with reinforcement learning for property optimization, which generate molecules in a sequential manner; c) flow-based methods GraphNVP [21] and GRF [10] which generate molecules at one shot and the state-of-the-art autoregressive-flow-based model GraphAF [30] which generates molecules in a sequential way.

Datasets. We use two datasets QM9 [27] and ZINC250K [11] for our experiments and summarize them in Table 1. The QM9 contains 133, 885 molecules with maximum 9 atoms in 4 different types, and the ZINC250K has 249, 455 drug-like molecules with maximum 38 atoms in 9 different types. The molecules are kekulized by the chemical software RDKit [17] and the hydrogen atoms are removed. There are three types of edges, namely single, double, and triple bonds, for all molecules. Following the pre-processing procedure in [21], we encode each atom and bond by one-hot encoding, pad the molecules which have less than the maximum number of atoms with an virtual atom, augment the adjacency tensor of each molecule by a virtual edge channel representing no bonds between atoms, and dequantize [8, 21] the discrete one-hot-encoded data by adding uniform random noise $U[0, 0.6]$ for each dimension, leading to atom matrix $A \in \mathbb{R}^{9\times5}$ and bond tensor $B \in \mathbb{R}^{4\times9\times9}$ for QM9, and $A \in \mathbb{R}^{38\times10}$ and $B \in \mathbb{R}^{4\times38\times38}$ for ZINC250k.

Table 1: Statistics of the datasets.

	#Mol. Graphs	Max. #Nodes	#Node Types	#Edge Types
QM9	133,885	9	4+1	3+1
ZINC250K	249,455	38	9+1	3+1

MoFlow Setup. To be comparable with one-shot-flow baseline GraphNVP [21], for the ZINC250K, we adopt 10 coupling layers and 38 graph coupling layers for the bonds' Glow and the atoms' graph conditional flow respectively. We use two $3 * 3$ convolution layers with 512, 512 hidden dimensions in each coupling layer. For each graph coupling layer, we set one relational graph convolution layer with 256 dimensions followed by a two-layer multilayer perceptron with 512, 64 hidden dimensions. As for the QM9, we adopt 10 coupling layers and 27 graph coupling layers for the bonds' Glow and the atoms' graph conditional flow respectively. There are two 3*3 convolution layers with 128, 128 hidden dimensions in each coupling layer, and one graph convolution layer with 64 dimensions followed by a two-layer multilayer perceptron with 128, 64 hidden dimensions in each graph coupling layer. As for the optimization experiments, we further train a regression model to map the latent embeddings to different property scalars (discussed in Sec. 5.3 and 5.4) by a multi-layer perceptron with 18-dim linear layer -> ReLu -> 1-dim linear layer structures. For each dataset, we use the same trained model for all the following experiments.

Empirical Running Time. Following above setup, we implemented our MoFlow by Pytorch-1.3.1 and trained it by Adam optimizer [13] with learning rate 0.001, batch size 256, and 200 epochs for both datasets on 1 GeForce RTX 2080 Ti GPU and 16 CPU cores. Our MoFlow finished 200-epoch training within 22 hours (6.6 minutes/epoch) for ZINC250K and 3.3 hours (0.99 minutes/epoch) for QM9. Thanks to efficient one-pass inference/embedding, our MoFlow takes negligible 7 minutes to learn an additional regression layer trained in 3 epochs for optimization experiments on ZINC250K. In comparison, as for the ZINC250K dataset, GraphNVP [21] costs 38.4 hours (11.5 minutes/epoch) by our Pytorch implementation for training on ZINC250K with the same configurations, and the estimated total running time of GraphAF [30] is 124 hours (24 minutes/epoch) which consists of the reported 4 hours for a generation model trained by 10 epochs and estimated 120 hours for another optimization model trained by 300 epochs. The reported running time of JT-VAE [12] is roughly 24 hours in [34].

5.1 Generation and Reconstruction

Setup. In this task, we evaluate our MoFlow 's capability of generating novel, unique and valid molecular graphs, and if our MoFlow can reconstruct input molecular graphs from their latent representations. We adopted the widely-used metrics, including: **Validity** which is the percentage of chemically valid molecules in all the generated molecules, **Uniqueness** which is the percentage of unique valid molecules in all the generated molecules, **Novelty** which is the percentage of generated valid molecules which are not in the training dataset, and **Reconstruction** rate which is the percentage of molecules in the input dataset which can be reconstructed from their latent representations. Besides, because the novelty score also accounts for the potentially duplicated novel molecules, we propose

Table 2: Generation and reconstruction performance on QM9 dataset.

	% Validity	% Validity w/o check	% Uniqueness	% Novelty	% N.U.V.	% Reconstruct
GraphNVP [21]	83.1 ± 0.5	n/a	99.2 ± 0.3	58.2 ± 1.9	47.97	100
GRF [10]	84.5 ± 0.70	n/a	66.0 ± 1.15	58.6 ± 0.82	32.68	100
GraphAF [30]	100	67	94.51	88.83	83.95	100
MoFlow	**100.00 ± 0.00**	**96.17 ± 0.18**	**99.20 ± 0.12**	**98.03 ± 0.14**	**97.24 ± 0.21**	**100.00 ± 0.00**

Table 3: Generation and reconstruction performance on ZINC250K dataset.

	% Validity	% Validity w/o check	% Uniqueness	% Novelty	% N.U.V.	% Reconstruct
JT-VAE [12]	100	n/a	100	100	100	76.7
GCPN [34]	100	20	99.97	100	99.97	n/a
MRNN [26]	100	65	99.89	100	99.89	n/a
GraphNVP [21]	42.6 ± 1.6	n/a	94.8 ± 0.6	100	40.38	100
GRF [10]	73.4 ± 0.62	n/a	53.7 ± 2.13	100	39.42	100
GraphAF [30]	100	68	99.10	100	99.10	100
MoFlow	**100.00 ± 0.00**	**81.76 ± 0.21**	**99.99 ± 0.01**	**100.00 ± 0.00**	**99.99 ± 0.01**	**100.00 ± 0.00**

a new metric **N.U.V.** which is the percentage of Novel, Unique, and Valid molecules in all the generated molecules. We also compare the validity of ablation models if not using validity check or validity correction, denoted as **Validity w/o check** in [30].

The prior distribution of latent space follows a spherical multivariate Gaussian distribution $\mathcal{N}(0, (t\sigma)^2\mathbf{I})$ where σ is the learned standard deviation and the hyper-parameter t is the temperature for the reduced-temperature generative model [14, 21, 24]. We use $t = 0.85$ in the generation for both QM9 and ZINC250K datasets, and $t = 0.6$ for the ablation study without validity correction. To be comparable with the state-of-the-art baseline GraphAF[30], we generate 10,000 molecules, i.e., sampling 10,000 latent vectors from the prior and then decode them by the reverse transformation of our MoFlow. We report the the mean and standard deviation of results over 5 runs. As for the reconstruction, we encode all the molecules from the training dataset into latent vectors by the encoding transformation of our MoFlow and then reconstruct input molecules from these latent vectors by the reverse transformation of MoFlow.

Results. Table 2 and Table 3 show that our MoFlow outperfoms the state-of-the-art models on all the six metrics for both QM9 and ZINC250k datasets. Thanks to the invertible characteristic of the flow-based models, our MoFlow builds an one-to-one mapping from the input molecule M to its corresponding latent vector Z, enabling 100% reconstruction rate as shown in Table 2 and Table 3. In contrast, the VAE-based method JT-VAE and the autoregressive-based method GCPN and MRNN can't reconstruct all the input molecules. Compared with the one-shot flow-based model GraphNVP and GRF, by incorporating validity correction mechanism, our MoFlow achieves 100% validity, leading to significant improvements of the validity score and N.U.V. score for both datasets. Specifically, the N.U.V. score of MoFlow are 2 and 3 times as large as the N.U.V. scores of GraphNVP and GRF respectively in Table 2. Even without validity correction, our MoFlow still outperforms the validity scores of GraphNVP and GRF by a large margin. Compared with the autoregressive flow-based model GraphAF, we find our MoFlow outperforms GraphAF by additional 16% and 0.8% with respect to N.U.V scores for QM9 and ZINC respectively, indicating that our MoFlow generates more novel, unique and valid molecules. Indeed, MoFlow achieves better uniqueness score and novelty score compared with GraphAF for both datasets. What's more, our MoFlow without validity correction still outperforms GraphAF without the validity check by a large margin w.r.t. the validity score (validity w/o check in Table 2 and Table 3) for both datasets, implying the superiority of capturing the molecular structures in a holistic way by our MoFlow over autoregressive ones in a sequential way.

In conclusion, our MoFlow not only memorizes and reconstructs all the training molecular graphs, but also generates more novel, unique and valid molecular graphs than existing models, indicating that our MoFlow learns a strict superset of the training data and explores the unknown chemical space better.

5.2 Visualizing Continuous Latent Space

Setup. We examine the learned latent space of our MoFlow , denoted as f, by visualizing the decoded molecular graphs from a neighborhood of a latent vector in the latent space. Similar to [12, 16], we encode a seed molecule M into $Z = f(M)$ and then grid search two random orthogonal directions with unit vector X and Y based on Z, then we get new latent vector by $Z' = Z + \lambda_X * X + \lambda_Y * Y$ where λ_X and λ_Y are the searching steps. Different from VAE-based models, our MoFlow gets decoded molecules efficiently by the one-pass inverse transformation $M' = f^{-1}(Z')$. In contrast, the VAE-based models such as JT-VAE need to decode each latent vectors $10 - 100$ times and autoregressive-based models like GCPN, MRNN and GraphAF need to generate a molecule sequentially. Further more, we measure the chemical similarity between each neighboring molecule and the centering molecule. We choose Tanimoto index [2] as the chemical similarity metrics and indicate their similarity values by a heatmap. We further visualize a linear interpolation between two molecules to show their changing trajectory similar to the interpolation case between images [14].

Results. We show the visualization of latent space in Figure 4. We find the latent space is very smooth and the interpolations between two latent points only change a molecule graph a little bit.

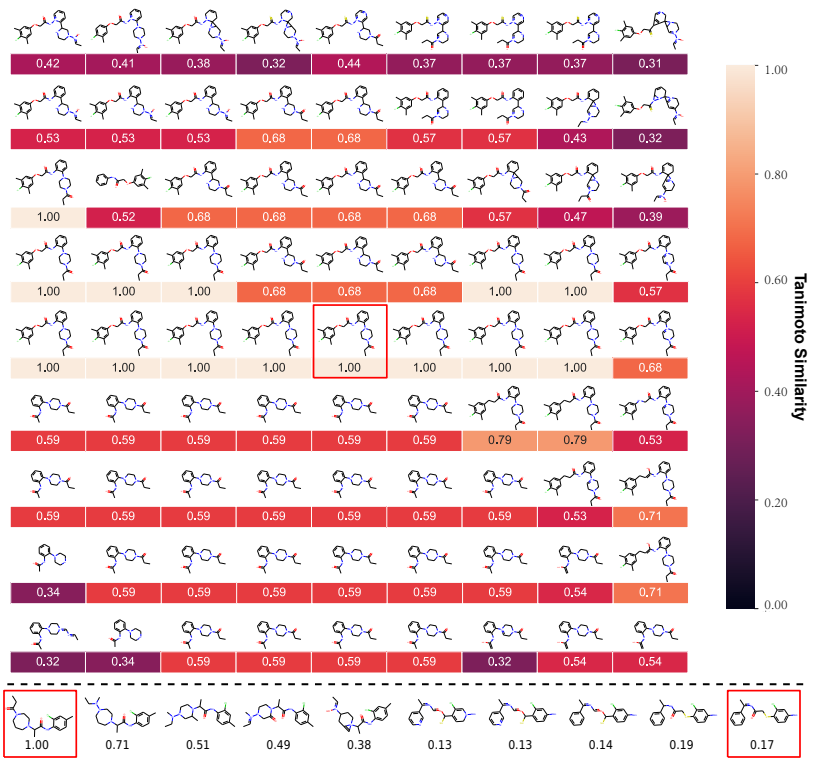

Figure 4: Visualization of learned latent space by our MoFlow. Top: Visualization of the grid neighbors of a seed molecule in the center, which serves as the baseline for measuring similarity. Bottom: Interpolation between two seed molecular graphs and the left one is the baseline molecule for measuring similarity. Seed molecules are highlighted in red boxs and they are randomly selected from ZINC250K.

Quantitatively, we find the chemical similarity between molecules majorly correspond to their Euclidean distance between their latent vectors, implying that our MoFlow embeds similar molecular graph structures into similar latent embeddings. Searching in such a continuous latent space learnt by our MoFlow is the basis for molecular property optimization and constraint optimization as discussed in the following sections.

5.3 Property Optimization

Setup. The property optimization task aims at generating novel molecules with the best Quantitative Estimate of Druglikeness (QED) scores [3] which measures the drug-likeness of generated molecules. Following the previous works [26, 34], we report the best property scores of novel molecules discovered by each method.

We use the pre-trained MoFlow, denoted as f, in the generation experiment to encode a molecule M and get the molecular embedding $Z = f(M)$, and further train a multilayer perceptron to regress the embedding Z of the molecules to their property values y. We then search the best molecules by the gradient ascend method, namely $Z' = Z + \lambda * \frac{dy}{dZ}$ where the λ is the length of the search step. We conduct above gradient ascend method by K steps. We decode the new embedding Z' in the latent space to the discovered molecule by reverse mapping $M' = f^{-1}(Z')$. The molecule M' is novel if M' doesn't exist in the training dataset.

Results. We report the discovered novel molecules sorted by their QED scores in Table 4. We find previous methods can only find very few molecules with the best QED score (= 0.948). In contrast, our MoFlow finds much more novel molecules which have the best QED values than all the baselines. We show more molecular structures with top QED values in Figure 5.

Table 4: Discovered novel molecules with the best QED scores. Our MoFlow finds more molecules with the best QED scores. More results in Figure 5.

Method	1st	2nd	3rd	4th
ZINC (Dataset)	0.948	0.948	0.948	0.948
JT-VAE	0.925	0.911	0.910	-
GCPN	0.948	0.947	0.946	-
MRNN	0.948	0.948	0.947	-
GraphAF	0.948	0.948	0.947	0.946
MoFlow	**0.948**	**0.948**	**0.948**	**0.948**

5.4 Constrained Property Optimization

Setup. The constrained property optimization aims at finding a new molecule M' with the largest similarity score $sim(M, M')$ and the largest improvement of a targeted property value $y(M') - y(M)$ given a molecule M. Following the similar experimental setup of [12, 34], we choose Tanimoto similarity of Morgan fingerprint [28] as the similarity metrics, the penalized logP (plogp) as the target property, and M from the 800 molecules with the lowest plogp scores in the training dataset of ZINC250K. We use similar gradient ascend method as discussed in the previous subsetion to search for optimized molecules. An optimization succeeds if we find a novel molecule M' which is different from M and $y(M') - y(M) \geq 0$ and

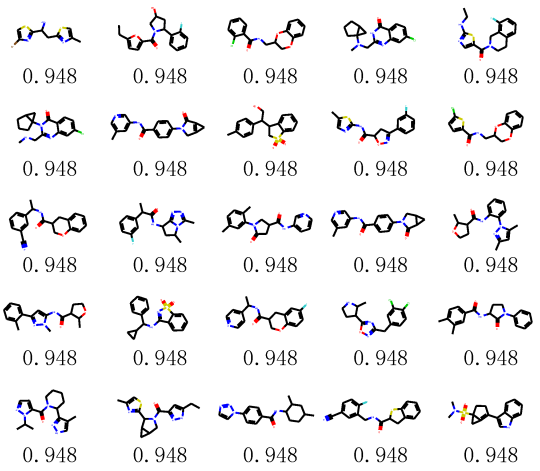

Figure 5: Illustration of discovered novel molecules with the best druglikeness QED scores.

Table 5: Constrained optimization on Penalized-logP

	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%
0.2	1.68 ± 1.85	0.33 ± 0.13	97.1%	4.12 ± 1.19	0.34 ± 0.11	100%
0.4	0.84 ± 1.45	0.51 ± 0.10	83.6%	2.49 ± 1.30	0.48 ± 0.08	100%
0.6	0.21 ± 0.71	0.69 ± 0.06	46.4%	0.79 ± 0.63	0.68 ± 0.08	100%
	GraphAF			**MoFlow**		
δ	**Improvement**	**Similarity**	**Success**	**Improvement**	**Similarity**	**Success**
0.0	13.13 ± 6.89	0.29 ± 0.15	100%	8.61 ± 5.44	0.30 ± 0.20	98.88%
0.2	11.90 ± 6.86	0.33 ± 0.12	100%	7.06 ± 5.04	**0.43 ± 0.20**	96.75%
0.4	8.21 ± 6.51	0.49 ± 0.09	99.88%	4.71 ± 4.55	**0.61 ± 0.18**	85.75%
0.6	4.98 ± 6.49	0.66 ± 0.05	96.88%	2.10 ± 2.86	**0.79 ± 0.14**	58.25%

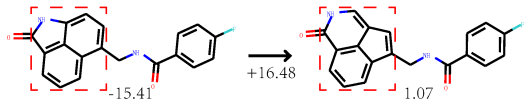

Figure 6: An illustration of the constrained optimization of a molecule leading to an improvement of +16.48 **w.r.t the penalized logP and with Tanimoto similarity** 0.624**. The modified part is highlighted.**

$sim(M, M') \geq \delta$ within K steps where δ is the smallest similarity threshold to screen the optimized molecules.

Results. Results are summarized in Table 5. We find that our MoFlow finds the most similar new molecules at the same time achieves very good plogp improvement. Compared with the state-of-the-art VAE model JT-VAE, our MoFlow achieves much higher similarity score and property improvement, implying that our model is good at interpolation and learning continuous molecular embedding. Compared with the state-of-the-art reinforcement learning based method GCPN and GraphAF which is good at generating molecules step-by-step with targeted property rewards, our model MoFlow achieves the best similarity scores and the second best property improvements. We illustrate one optimization example in Figure 6 with very similar structures but a large improvement w.r.t the penalized logP.

6 CONCLUSION

In this paper, we propose a novel deep graph generative model MoFlow for molecular graph generation. Our MoFlow is one of the first flow-based models which not only generates molecular graphs at one-shot by invertible mappings but also has a validity guarantee. Our MoFlow consists of a variant of Glow model for bonds, a novel graph conditional flow for atoms given bonds, and then combining them with post-hoc validity corrections. Our MoFlow achieves state-of-the-art performance on molecular generation, reconstruction and optimization. For future work, we try to combine the advantages of both sequential generative models and one-shot generative models to generate chemically feasible molecular graphs. Codes and datasets are open-sourced at https://github.com/calvin-zcx/moflow.

ACKNOWLEDGEMENT

This work is supported by NSF IIS 1716432, 1750326, ONR N00014-18-1-2585, Amazon Web Service (AWS) Machine Learning for Research Award and Google Faculty Research Award.

REFERENCES

[1] Jerry Avorn. 2015. The $2.6 billion pillâĂŤmethodologic and policy considerations. *New England Journal of Medicine* 372, 20 (2015), 1877–1879.
[2] Dávid Bajusz, Anita Rácz, and Károly Héberger. 2015. Why is Tanimoto index an appropriate choice for fingerprint-based similarity calculations? *Journal of cheminformatics* 7, 1 (2015), 20.
[3] 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.
[4] Xavier Bresson and Thomas Laurent. 2019. A Two-Step Graph Convolutional Decoder for Molecule Generation. *arXiv preprint arXiv:1906.03412* (2019).
[5] Hanjun Dai, Yingtao Tian, Bo Dai, Steven Skiena, and Le Song. 2018. Syntax-directed variational autoencoder for structured data. *arXiv preprint arXiv:1802.08786* (2018).
[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] 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.
[10] 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).
[11] 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.
[12] Wengong Jin, Regina Barzilay, and Tommi Jaakkola. 2018. Junction tree variational autoencoder for molecular graph generation. *arXiv preprint arXiv:1802.04364* (2018).
[13] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).
[14] Durk P Kingma and Prafulla Dhariwal. 2018. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in Neural Information Processing Systems*. 10215–10224.
[15] Ivan Kobyzev, Simon Prince, and Marcus A Brubaker. 2019. Normalizing flows: Introduction and ideas. *arXiv preprint arXiv:1908.09257* (2019).
[16] Matt J Kusner, Brooks Paige, and José Miguel Hernández-Lobato. 2017. Grammar variational autoencoder. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 1945–1954.
[17] Greg Landrum et al. 2006. RDKit: Open-source cheminformatics.
[18] Jenny Liu, Aviral Kumar, Jimmy Ba, Jamie Kiros, and Kevin Swersky. 2019. Graph normalizing flows. In *Advances in Neural Information Processing Systems*. 13556–13566.
[19] Qi Liu, Miltiadis Allamanis, Marc Brockschmidt, and Alexander Gaunt. 2018. Constrained graph variational autoencoders for molecule design. In *Advances in Neural Information Processing Systems*. 7795–7804.
[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, Katushiko Ishiguro, Kosuke Nakago, and Motoki Abe. 2019. GraphNVP: An Invertible Flow Model for Generating Molecular Graphs. *arXiv*

preprint arXiv:1905.11600 (2019).

[22] Asher Mullard. 2017. The drug-maker's guide to the galaxy. *Nature News* 549, 7673 (2017), 445.

[23] George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. 2019. Normalizing Flows for Probabilistic Modeling and Inference. *arXiv preprint arXiv:1912.02762* (2019).

[24] Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Łukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. 2018. Image transformer. *arXiv preprint arXiv:1802.05751* (2018).

[25] Steven M Paul, Daniel S Mytelka, Christopher T Dunwiddie, Charles C Persinger, Bernard H Munos, Stacy R Lindborg, and Aaron L Schacht. 2010. How to improve R&D productivity: the pharmaceutical industry's grand challenge. *Nature reviews Drug discovery* 9, 3 (2010), 203.

[26] Mariya Popova, Mykhailo Shvets, Junier Oliva, and Olexandr Isayev. 2019. MolecularRNN: Generating realistic molecular graphs with optimized properties. *arXiv preprint arXiv:1905.13372* (2019).

[27] 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 (2014), 140022.

[28] David Rogers and Mathew Hahn. 2010. Extended-connectivity fingerprints. *Journal of chemical information and modeling* 50, 5 (2010), 742–754.

[29] 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.

[30] Chence Shi, Minkai Xu, Zhaocheng Zhu, Weinan Zhang, Ming Zhang, and Jian. Tang. 2020. GraphAF: a Flow-based Autoregressive Model for Molecular Graph Generation. *ICLR 2020, Addis Ababa, Ethiopia, Apr.26-Apr. 30, 2020* (2020).

[31] 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.

[32] Mengying Sun, Sendong Zhao, Coryandar Gilvary, Olivier Elemento, Jiayu Zhou, and Fei Wang. 2019. Graph convolutional networks for computational drug development and discovery. *Briefings in bioinformatics* (2019).

[33] 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.

[34] Jiaxuan You, Bowen Liu, Zhitao Ying, Vijay Pande, and Jure Leskovec. 2018. Graph convolutional policy network for goal-directed molecular graph generation. In *Advances in Neural Information Processing Systems*. 6410–6421.

[35] Alex Zhavoronkov, Yan A Ivanenkov, Alex Aliper, Mark S Veselov, Vladimir A Aladinskiy, Anastasiya V Aladinskaya, Victor A Terentiev, Daniil A Polykovskiy, Maksim D Kuznetsov, et al. 2019. Deep learning enables rapid identification of potent DDR1 kinase inhibitors. *Nature biotechnology* 37, 9 (2019), 1038–1040.