DEMO: Disentangled Molecular Graph Generation via an Invertible Flow Model

Changsheng Ma
KAUST Computational Bioscience Research Center,
King Abdullah University of Science and Technology
changsheng.ma@kaust.edu.sa

Qiang Yang
KAUST Computational Bioscience Research Center,
King Abdullah University of Science and Technology
qiang.yang@kaust.edu.sa

Xin Gao
KAUST Computational Bioscience Research Center,
King Abdullah University of Science and Technology
xin.gao@kaust.edu.sa

Xiangliang Zhang*
University of Notre Dame, Indiana, United States
King Abdullah University of Science and Technology
xzhang33@nd.edu

ABSTRACT

Molecular graph generation via deep generative models has attracted increasing attention. This is a challenging problem because it requires optimizing a given objective under a huge search space while obeying the chemical valence rules. Although recently developed molecular generation models have achieved promising results on generating novel, valid and unique molecules, few efforts have been made toward interpretable molecular graph generation. In this work, we propose DEMO, a flow-based model for DisEntangled Molecular graph generatiOn in a completely unsupervised manner, which is able to generate molecular graphs w.r.t. the learned disentangled latent factors that are relevant to molecular semantic features and interpretable structural patterns. Specifically, DEMO is composed of a VAE-encoder and a flow-generator. The VAE-encoder focuses on extracting global features of molecular graphs, and the flow-generator aims at disentangling these features to be corresponding to certain types of understandable molecular structure features while learning data distributions. To generate molecular graphs, DEMO simply runs the flow-generator in the reverse order due to the reversibility of the flow-based models. Extensive experimental results on two benchmark datasets demonstrate that DEMO outperforms the state-of-the-art methods in molecular generation, and takes the first step in interpretable molecular graph generation.

CCS CONCEPTS

• **Theory of computation** → Graph algorithms analysis; • **Computing methodologies** → Learning latent representations.

KEYWORDS

Molecular Graph Generation, Interpretable Molecular Generation, Flow-based Generative Model, Disentangled Learning

*Xiangliang Zhang is the corresponding author

ACM Reference Format:
Changsheng Ma, Qiang Yang, Xin Gao, and Xiangliang Zhang. 2022. DEMO: Disentangled Molecular Graph Generation via an Invertible Flow Model. In *Proceedings of the 31st ACM Int'l Conference on Information and Knowledge Management (CIKM '22), Oct. 17–21, 2022, Atlanta, GA, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3511808.3557217

1 INTRODUCTION

Finding new molecules with desired properties is a crucial problem in drug discovery. This is a very challenging and urgent problem because the scale of the drug-like compounds is as large as 10^{60} [28], but the compounds that have been discovered are only the tip of the iceberg. Recently, deep learning (DL) methods have shown a promising approach to producing satisfactory candidate molecules, avoiding costly chemical space searches [10, 39]. As a particular class of DL methods, deep generative models that model the underlying probability distributions of structures or properties of datasets, have achieved notable results in molecular graph generation [38].

Despite promising results, the existing molecular generation models have a major limitation: the generation process is difficult to interpret. Because existing graph generative models map the graph structural information into continuous latent representations while neglecting the entanglement of the latent factors, rendering the learned representations hardly explainable. Disentangled representation learning enables the learned latent representations to separate the distinct informative factors of the data, which has caused a lot of attention in the computer vision domain. One popular stream of work is to modify the objective function of the Variational autoencoder (VAE) [19]. As for the graph domain, disentangled graph representation learning was studied for improving graph/node classification tasks [21, 25]. However, disentangled enhancement has rarely been explored for graph generation.

To make molecular generation understandable, we propose a model named DEMO, a flow-based model for DisEntangled Molecular graph generatiOn. DEMO is designed to learn disentangled latent factors relevant to molecular features, and do a favor in characterizing the correspondence between the molecule graph structure and latent variables in generative space with better transparency. Figure 1 shows examples in a 3D space, where generated molecules in each direction have similar semantic-level features (structures), accompanied by subtle differences in syntactic-level features (substructures). This in our case is realized by manipulating one dimension of the learned disentangled factors during the generation process.

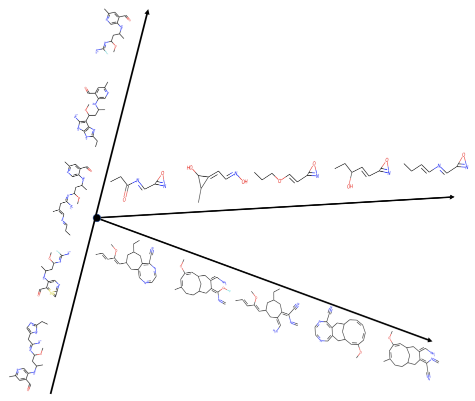


Figure 1: Demonstration of molecular graph generation in disentangled 3D space. Each dimension corresponds to a certain type of molecular structure feature.

Technically, our proposed DEMO consists of a VAE-encoder and a flow-generator. The VAE encoder is responsible for learning global features of the molecular graphs, which are then fed into the flow-generator for disentangling. The target of the flow-generator is to map molecular graphs to their latent representations to approximate the data distribution under the restriction of the learned disentangled factors, which enhance the interpretability of the generated results. The cooperation between the VAE-encoder and the flow-generator can be explained from both perspectives.

From the VAE-encoder perspective, its encoding process needs an extra loss function to make the learned global features disentangled. Since the target distribution of the flow-generator follows one isomorphic Gaussian distribution, this makes each dimension of the latent representation independent of each other. Sending the learned global features of the VAE-encoder to the flow-generator for training, the loss function of the flow-generator actually acts as an implicit *total correlation* [36] penalty working on them, making them independent in the dimensions, i.e., being disentangled. Furthermore, the features learned by the VAE-encoder bring semantic-level features to the flow-generator, alleviating the defect that flow-based models tend to capture local dependencies among features.

From the flow-generator prospective, in each coupling layer, a flow-based model only randomly processes half the amount of information to learn syntactic-level features. In one iteration, a flow-based model stacks multiple coupling layers, which work for combining multiple different syntactic-level features. Since our flow-generator learns an invertible function mapping input data to an isomorphic Gaussian distribution, by including VAE encoding in its training process, it illuminates partial features learned in the VAE-encoder and enforces them to be independent of each other. To this end, over multiple iterations, every dimension of the features learned in the VAE-encoder is disentangled.

Compared to the previous disentangled methods [22] that proceed from the information theory and explicitly impose a loss function to achieve disentanglement, our model jumps out of this paradigm. It applies a flow-based model to map the latent representations to an isomorphic Gaussian distribution, making all latent factors be independent of each other to achieve disentanglement.

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

- We move the first step toward interpretable molecular graph generation. The generated molecular graphs are interpretable w.r.t. structural patterns and molecular semantic features.
- Our designed DEMO model offers a new perspective for dealing with disentangled representation learning.
- Extensive experiments show that DEMO outperforms the state-of-the-art models in the molecular graph generation task. Meanwhile, DEMO is able to obtain robust latent representations and explainable disentangled factors.

2 RELATED WORK

2.1 Molecular Graph Generation

A variety of deep graph generative models have been proposed for *de-novo* molecule generation. According to the generation mode, these methods can be categorized into one-shot generation and sequential generation. In the family of one-shot generation, GraphVAE [33] is a pioneering work applying VAE to realize graph-to-graph autoencoding. RVAE [26] presents a regularization framework as a step toward semantic validity, transforming a constrained optimization problem into a regularized unconstrained one. In the group of sequential generation, GCPN [41] generates atoms and bonds of molecules one by one. JT-VAE [16] generates scaffold of chemical substructure components step by step. Our model adopts one-shot generation for two reasons. First, the sequential way is more time-consuming compared to the one-shot way. Second, sequential generation requires a predefined sequence to specify the generation process (usually in accordance with the SMILES [37] sequence), which may limit the search capabilities of the model.

2.2 Flow-based Models

Flow-based models [6, 7, 18] learn mappings between complex distributions and simple prior distributions through invertible neural networks. Allowing the exact and tractable likelihood estimation for training is their key feature, enabling the reconstruction of all original data. This property is naturally suitable for molecular graph generation due to the high sensitivity to noise in the generative process, where even one wrong generation of a valence will lead to the failure of the whole process. GraphNVP [27] is the first work that generates molecular graphs by a flow-based model, GRF [13] improves it by using residual flows. GraphAF [32] is an autoregressive flow-based model which generates molecular graphs in a sequential iterative manner. GraphDF [23] is an autoregressive RL-finetuned flow model that maps discrete latent variables to graph nodes and edges and eliminates the negative effect of dequantization [15]. MoFlow [42] is a recent model achieving the state-of-the-art performance in molecular graph generation based on flow models. It not only generates molecular graphs in a one-shot way, but also has a post-hoc validity correction. GF-VAE [24] optimizes the time cost of MoFlow by combining VAE with the flow-based models and achieves comparable performance. We build DEMO on flow-based models to take advantage of its natural advantages in molecular graph generation. More importantly, the novelty of DEMO is its capability on interpretation of the generated molecular graphs.

2.3 Disentangled Representation Learning

Learning a disentangled representation is favored in the generative model as one latent unit can be sensitive to the change of one

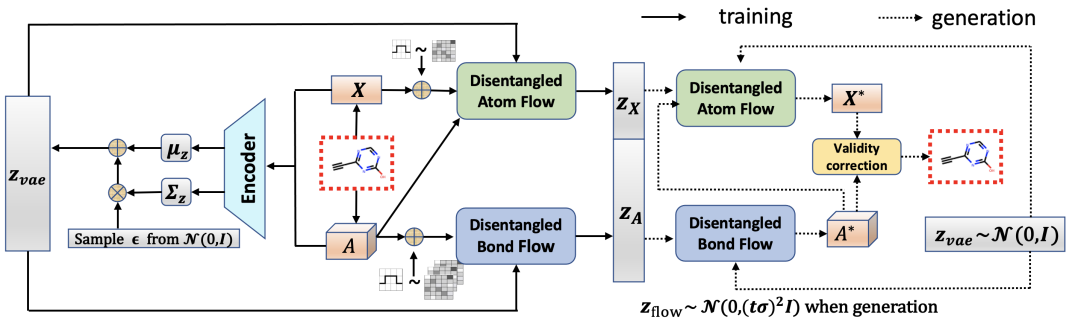


Figure 2: The overall architecture of DEMO

generative factor while being relatively invariant to changes in other factors [2]. Such representations are demonstrated to be more resilient to the complex variants, and able to bring enhanced generalization ability as well as improved robustness facing downstream tasks. Moreover, the disentangled representations are inherently more explainable, and thus can potentially facilitate interpretable generative models. Current disentangled learning mainly focuses on image applications, with a number of approaches that modify the VAE objective function by adding more restrictions to make latent factors more disentangled [4, 5, 11, 17, 20]. In contrast, only a few works consider learning disentangled factors for graphs. To our best knowledge, [25] is the first work that proposes a novel neighborhood routing mechanism to learn disentangled node representations for graphs. [21] then employs the Hilbert-Schmidt Independence Criterion (HSIC) to strengthen the independence performance. These models are designed for solving graph/node classification tasks. However, the problem of disentangled graph generation remains largely unexplored.

Our proposed DEMO model is a new design of disentangled representation learning for the graph generation problem. It differs from other VAE-based disentangled representation learning models, as it introduces flow-based models to implicitly place the restriction on disentangled factors.

3 THE PROPOSED DEMO METHOD

3.1 Problem Formulation

Let $G = (A, X)$ represent a molecular graph G consisting of an adjacency tensor A and a feature matrix X. Let N be the number of atom nodes, M be the number of atom types and R be the number of bond types. Then we have $A \in \{0,1\}^{N\times N\times R}$ and $X \in \{0,1\}^{N\times M}$. The goal is to learn a disentangled generative model $p_{\mathcal{G}}(G)$ from a given set of graphs $\mathcal{G}$, such that a sample drawn from the distribution $p_{\mathcal{G}}$ is a valid molecular graph. Meanwhile, the latent variables in distribution $p_{\mathcal{G}}$ correspond to a set of disentangled factors that can manipulate the structured semantic features of $\mathcal{G}$.

The overall architecture of our proposed DEMO model is shown in Figure 2. The training process consists of two parts. The VAE-encoder aims at extracting the global feature of a molecular graph G. The flow-generator is responsible for approximating the data distribution $p_{\mathcal{G}}$. Meanwhile, it eliminates the dependencies among global features $\mathbf{z}_{\mathbf{vae}}$ learned from the VAE-encoder by the newly designed *Disentangled Atom Flow* module and *Disentangled Bond Flow* module. Due to the reversibility of the flow-based models, the generation process is performed by running the *Disentangled Atom Flow* module and the *Disentangled Bond Flow* module in the reverse order, followed by a validity correction step.

3.2 VAE-encoder

The VAE-encoder maps a molecular graph $G = (A, X)$ to a latent vector $\mathbf{z}_{\mathbf{vae}} \in \mathbb{R}^D$, which follows a Gaussian distribution $\mathcal{N}(\mu_z, \Sigma_z)$, where μ_z and Σ_z are mean and diagonal covariance matrix of $\mathbf{z}_{\mathbf{vae}}$ respectively. In it, D is the output dimension size of the VAE-encoder.

Since our target is to learn the disentangled factors that can manipulate the semantic level features rather than syntactic level features, we employ a spatial-based graph convolutional network (GCN) to obtain the global structure information, rather than a spectral-based GCN which focuses more on mining information of neighbors. Specifically, we adopt GIN [40] to learn the feature vector of node i at layer l as

$$\boldsymbol{h}_i^{(\ell)} = MLP^{(\ell)}((1+\gamma^{(\ell)})\boldsymbol{h}_i^{\ell-1} + \sum_{j=1}^{N}\sum_{r=1}^{R} A_{ijr}\boldsymbol{h}_j^{(\ell-1)}), \quad (1)$$

where at the initial layer $\boldsymbol{h}_i^{(0)} = \boldsymbol{x}_i \in \{0,1\}^M$, which is the atom feature from X. $MLP^\ell(*)$ is a multi-layer perceptron, and γ^ℓ is a learnable parameter to adjust the weight of the central node at layer ℓ. After L layers of propagation, we aggregate nodes embedding into a graph level representation vector by taking sum as the readout function as

$$\boldsymbol{h}_G = \text{concat}(\text{readout}(\{\boldsymbol{h}_i^{(\ell)} \mid i \in G\}) \mid \ell = 0, ..., L). \quad (2)$$

To force the encoding vector $\mathbf{z}_{\mathbf{vae}}$ to roughly conform to a normal distribution, we separately apply MLP to $\boldsymbol{h}_G$ to get the mean vector μ_z and the diagonal covariance matrix Σ_z as

$$\mu_z = MLP(\boldsymbol{h}_G), \qquad \Sigma_z = MLP(\boldsymbol{h}_G). \quad (3)$$

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

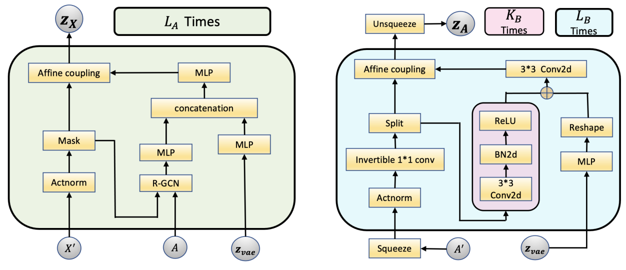
$$\mathbf{z}_{\text{vae}} = \mu_z + \epsilon \odot \Sigma_z, \quad (4)$$

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:

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

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

To disentangle the latent representation $\mathbf{z}_{\mathbf{vae}}$, we connect the VAE-encoder with the flow-generator, and let $\mathbf{z}_{\mathbf{vae}}$ be a part of input to the flow-generator.



(a) (b)

Figure 3: (a) Disentangled Atom Flow, and (b) Disentangled Bond Flow.

3.3 Flow-generator

Our flow-generator is composed of two cooperative modules: the *disentangled atom flow* aims to approximate the distribution of atoms, and the *disentangled bond flow* is used to capture the distribution of bonds. Moreover, these two modules collaborate with the VAE-encoder to disentangle $\mathbf{z_{vae}}$.

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 [15]. We hence dequantize G by adding a uniform random noise $U[0,1)$ to A and X as

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

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 an isomorphic Gaussian distribution. We base our two modules on Glow [18] and many basic components are applied in ours. Such as *Actnorm* and *invertible 1∗1 convolution*, *squeeze* and *unsqueeze*. We will give a brief introduction of them later.

3.3.1 Disentangled Atom Flow. For the *disentangled atom flow* module, an invertible mapping function f_ω^{atom} is learned to get $\mathbf{z}_X \in \mathbb{R}^{N\times M}$ by $f_\omega^{atom}(X', A, \mathbf{z_{vae}})$, shown in Figure 3 (a). Since the flow-based models tend to capture the dependencies among local features, we want to amplify this trend as much as possible in order to impose more precise penalties to achieve better disentanglement. Therefore, we apply R-GCN [31] to extract 1-hop information of atoms to build the atom affine coupling function as

$$\begin{aligned} x_a, x_b &= \text{Mask}(\text{Actnorm}(X')), \\ x_b &= \text{MLP}(\text{R-GCN}(x_b, A)), \\ \log s_\omega, t_\omega &= \text{MLP}(\text{concat}(x_b, \text{MLP}(\mathbf{z_{vae}}))), \\ \mathbf{z}_X &= \text{concat}\left(x_a, \text{sigmod}(\log s_\omega) \odot x_a + t_\omega\right). \end{aligned} \tag{7}$$

The operation *Mask* is commonly used in the image domain [7] which randomly separates tensors into two parts. Here we conduct this operation on the atom node channel. For the scale function s_ω and the transformation function t_ω of atoms, we use *MLP* to build them, not only based on the knowledge explored by R-GCN, but also taking into account the knowledge $\mathbf{z_{vae}}$ learned from the VAE-encoder. Due to the output $\mathbf{z}_X$ following an isomorphic Gaussian distribution, this will force the representation of $\mathbf{z_{vae}}$ to be mapped onto mutually orthogonal dimensions during training, enabling $\mathbf{z_{vae}} = \prod_{j=1}^{D} \mathbf{z_{vae}}(j)$, i.e., disentangled.

In order to better approximate the real distribution of atoms, the whole process (Eq (7)) needs to be stacked L_A times to enhance the mapping performance. We thus adopt the Sigmoid function for the scale function s_ω rather than the exponential function for the sake of the numerical stability of cascading multiple layers.

Since f_ω^{atom} is invertible, sampling a latent vector $\mathbf{z}_X \in \mathbb{R}^{N\times M}$ from a spherical multivariate Gaussian distribution $\mathcal{N}\left(0, (t\sigma)^2\mathbf{I}\right)$, where σ is the learned standard deviation and the hyper-parameter t is the temperature for the reduced-temperature generative model [27], based on the **change of variable formula**, the log-likelihood of the reconstructed feature matrix X^* is calculated as

$$\log p_{X^*} = \log p(\mathbf{z}_X) + \log\det\left|\frac{\partial f_\omega^{atom}(X', A, \mathbf{z_{vae}})}{\partial X'}\right|. \tag{8}$$

3.3.2 Disentangled Bond Flow. The *disentangled bond flow* module shown in Figure 3 (b) learns an invertible mapping for $\mathbf{z}_A \in \mathbb{R}^{R\times N\times N}$ from $f_\theta^{bond}(A', \mathbf{z_{vae}})$. Since the flow-based models have strong advantages in generating high-resolution images, we treat the dequantized adjacency matrix A' as a 3d image and adopt convolution neural network (CNN) to mine the latent knowledge of it. However, similar to images, the amount of the molecular graph channel R (bond types) is small, in order to guarantee a competitive performance, we need to increase the size of the channel axis without destroying the local correlation. So we apply *squeeze* on the dequantized adjacency matrix A' to realize this purpose and employ *unsqueeze* to recover the processed tensor $\mathbf{z}_A$ as,

$$A' = \text{Squeeze}(A'), \quad \mathbf{z}_A = \text{Unsqueeze}(\mathbf{z}_A), \tag{9}$$

and build the bond affine coupling function as

$$\begin{aligned} x_a, x_b &= \text{Split}(\text{Invertible } 1 * 1 \text{ conv}(\text{Actnorm}(A'))), \\ x_b &= \text{ReLU}(\text{BatchNorm2d}(3 * 3 \text{ Conv2d}(x_b))), \\ z &= \text{Reshape}(\text{MLP}(\mathbf{z_{vae}})), \\ \log s_\theta, t_\theta &= 3 * 3 \text{ Conv2d}(\text{ElementWiseAddition}(x_b, z)), \\ \mathbf{z}_A &= \text{concat}\left(x_a, \text{sigmod}(\log s_\theta) \odot x_a + t_\theta\right). \end{aligned} \tag{10}$$

The operation *Split* operates on the bond type channel and cuts tensors into two equal parts. In order to fully mine the information of the adjacency matrix, we build a CNN block which is composed of a K_B times stacked $3 * 3$ Conv2d->BatchNorm2d->ReLU layers. Similarly, for the scale function s_θ and the transformation function t_θ of bonds, we employ $3 * 3$ Conv2d to construct them, additionally considering the knowledge $\mathbf{z_{vae}}$ learned from the VAE-encoder on top of the knowledge explored by the CNN block. Since the output $\mathbf{z}_A$ also follows an isomorphic Gaussian distribution, this forces $\mathbf{z_{vae}}$ to be disentangled and enhances the overall disentanglement performance. The whole process (Eq (10)) needs to be stacked L_B times to improve the mapping performance so as to approximate the real distribution of bonds.

Since f_θ^{bond} is invertible, sampling a latent vector $\mathbf{z}_A \in \mathbb{R}^{R\times N\times N}$ from the same distribution as in the disentangled atom flow module, based on the **change of variable formula**, the log-likelihood of the reconstructed adjacency matrix A^* is calculated as

$$\log p_{A^*} = \log p(\mathbf{z}_A) + \log\det\left|\frac{\partial f_\theta^{bond}(A', \mathbf{z_{vae}})}{\partial A'}\right|. \tag{11}$$

3.3.3 *Objective Function & Basic Components of Flow-generator.*

For the reconstructed graph $G^* = (A^*, X^*)$, as we know the log-likelihood of the disentangled bond flow $\log p_{A^*}$ for the reconstructed adjacency matrix A^* and the log-likelihood of the disentangled atom flow $\log p_{X^*}$ for the reconstructed feature matrix X^*, the parameters of the flow-generator can be trained by minimizing the overall negative log-likelihood as

$$\mathcal{L}_{\text{flow}} = -\log p_{G^*=(A^*,X^*)} = -(\log p_{A^*} + \log p_{X^*}). \tag{12}$$

We next give a short introduction of the basic components of Glow model applied in our work.

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 its inverse process.

Actnorm. This process performs an affine transformation of the activations using a scale and bias parameter per channel. These parameters are initialized such that the post-actnorm activations per-channel have zero mean and unit variance given an initial minibatch of data. As for atom matrix which is a 2D tensor, we normalize each row to realize it.

Invertible 1$*$1 convolution. This 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.

3.4 Training & Generation

The training process follows the solid line of Figure 2 as forward calculation on training graphs by minimizing the loss function defined as

$$\mathcal{L}_{\text{DEMO}} = \mathcal{L}_{\text{KL}} + \mathcal{L}_{\text{flow}}. \tag{13}$$

The model parameters in the VAE-encoder, the disentangled atom flow and the disentangled bond flow are jointly optimized.

The generation process follows the dotted line of Figure 2. Since the disentangled atom flow f_{ω}^{atom} and the disentangled bond flow f_{θ}^{bond} are invertible, the generation process can be realized by sending random sampled vectors into them but in the reverse order. Specifically, we first draw a random sample $\mathbf{z_{flow}} \in \mathbb{R}^{R\times N\times N+N\times M}$ from a spherical multivariate Gaussian distribution $\mathcal{N}(0, (t\sigma)^2\mathbf{I})$ and split it into $\mathbf{z_A} \in \mathbb{R}^{R\times N\times N}$ and $\mathbf{z_X} \in \mathbb{R}^{N\times M}$. Than we draw another random sample $\mathbf{z_{vae}} \in \mathbb{R}^D$ from a standard Gaussian distribution as disentangled factors. After that, we simply put $\mathbf{z_A}$ and $\mathbf{z_{vae}}$ into f_{θ}^{bond} to generate adjacency matrix, and feed it with $\mathbf{z_X}$ and $\mathbf{z_{vae}}$ together into f_{ω}^{atom} to generate feature matrix. Last, we put the generated adjacency matrix and generated feature matrix into the validity correction module [42] to get the final generated molecule.

3.5 Model Discussion

3.5.1 *Disentanglement Analysis.*

In this part, we focus on strengthening the understanding of the disentanglement process. Eq (12) shows the objective function of the flow-generator, which consists of two parts, the log-likelihood of the adjacency matrix and the log-likelihood of the feature matrix. We transform it into another two parts based on the operation type: the log value of the latent distribution of adjacency matrix and feature matrix, as well as the log value of the Jacobian determiant of adjacency matrix and feature matrix. Then the overall loss function Eq (13) becomes

$$\underbrace{\underbrace{\mathcal{L}_{\text{KL}}}_{\text{(i) Feature Extraction}} - \underbrace{(\log p(\mathbf{z}_A) + \log p(\mathbf{z}_X))}_{\text{(ii) Total Correlation}}}_{(iv) Disentangled} - \underbrace{\log\det|\cdot|}_{\text{(iii) Reconstruction}}. \tag{14}$$

In it, (i) is referred to as the *Feature Extraction*, which comes from the VAE-encoder to extract the global features $\mathbf{z_{vae}}$ of the molecular graph G. (ii) plays the role of the *Total Correlation* (TC) [36], because the target distribution of the adjacency matrix $\mathbf{z}_A$ and the feature matrix $\mathbf{z}_X$ follow an isomorphic Gaussian distribution. This enables the mapped variables dimensionally independent. Feeding $\mathbf{z_{vae}}$ into the flow-generator, the TC penalty will force it to be statistically independent in the data distribution, i.e., (iv) *Disentangled*, which returns the disentangled factors of G.

Additionally, (iii) plays the role of the *Reconstruction*, based on the **change of variable formula**. This part bridges the connection between the complex original data distribution and the simple latent distribution. Obviously, this requires the original space and the target latent space share the same size of dimensions. Because of this major limitation that strictly restricts the degrees of freedom for disentanglement factors, we cannot realize a satisfactory disentanglement via a pure normalizing flow model [7]. In contrast, DEMO can determine the degrees of freedom by setting different D (the output dimension of the VAE-encoder) to enjoy flexibility.

3.5.2 *Time Complexity.*

The computational complexity of our model depends on the affine coupling function of atoms and bonds in the flow-generator. We apply $\|A\|_1$ to denote the number of nonzeros in the adjacency tensor A, the number of features is equal to the number of atom types since we adopt one-hot encoding. So the time complexity of the disentangled atom flow T_a is $O\left(L_A R(\|A\|_1 M + NM^2)\right)$. Let F be the size of the feature map, K be the kernel size, C be the channel size, and l be the number of layers. Then the time complexity of the disentangled bond flow T_b is $O(L_B \sum_{l=1}^{K_B} F_l^2 \cdot K_l^2 \cdot C_{l-1} \cdot C_l)$. And the time complexity of our model is the sum of T_a and T_b.

4 EXPERIMENTAL EVALUATION

In this section, we verify the performance of our model in terms of the generation performance and the disentanglement performance, and answer the following questions:

For the generation performance, we conduct

- **Molecular graph generation quality (Sec. 4.2)**: Can DEMO reconstruct all trained molecules and generate as many valid, novel and unique molecules as possible?
- **Molecule optimization (Sec. 4.3)**: Can DEMO generate novel molecular graphs with the optimized properties? Can DEMO generate novel molecular graphs with the optimized properties while keeping the chemical similarity as much as possible?

For the disentanglement performance, we conduct

- **Impact on downstream tasks (Sec. 4.4.1)**: Can DEMO obtain good representations to improve the performance for downstream tasks?
- **Disentanglement evaluation (Sec. 4.4.2)**: Can DEMO achieve competitive results on classical evaluation metrics for disentanglement learning?

- **Interpretation by the disentangled factors (Sec. 4.4.3)**: Can the generation results be interpreted by the learned disentangled factors?

4.1 Experiment Setting

4.1.1 Benchmark Datasets. We choose two commonly used chemical molecular datasets, QM9 [29] and ZINC-250K[14] 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 matrix 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 the VAE-encoder, we use adjacency tensor directly. For the input of the flow-generator, We dequantize the discrete one-hot-encoded data by setting $c = 0.6$ as [27], making uniform random noise $U[0, 0.6)$ for each dimension.

4.1.2 Baselines. We choose the following two groups of baselines: **a) VAE-based models**: **RVAE** [26] and **JT-VAE** [16] are the recent state-of-the-art VAE-based 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 **GraphVAE** [33], which is the first VAE model to learn from molecular graphs.
b) Flow-based models: GraphAF [32] and **GraphDF** [23] are the two state-of-the-art flow-based models which generates molecules in a sequential way. **MoFlow** [42] is the state-of-the-art flow-based model which generates molecules in one-shot way. We also choose **GraphNVP** [27] and **GRF** [13], the former is the first work to apply the normalizing flow model to generate molecular graphs, the latter improves the former by using residual flows.

For the evaluation of the generative performance, we choose all baselines. For the evaluation of the disentanglement performance, we compared with the most relevant models of our DEMO, **GraphVAE** and **MoFlow**, since there is no model yet that involves generating molecules from a disentanglement perspective.

4.1.3 Evaluation Metrics. We adopt the widely-used metrics to evaluate the generation performance: ***Validity*** - the percentage of chemically valid molecules among all generated molecules; ***Novelty*** - the percentage of generated valid molecules not presenting in the training set; ***Uniqueness*** - the percentage of unique valid molecules out of all generated molecules; and ***Reconstruction*** - the percentage of the molecules that can be reconstructed from their own latent vectors. However, it is difficult 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). Therefor, we adopt another metric ***Score***, which is the product of *Validity*, *Novelty* and *Uniqueness* in order to compromise these basic metrics. We also report ***Validity w/o check*** as the ablation models that not use the validity correction module.

We evaluate the disentanglement performance of models by the following metrics: $\boldsymbol{\beta} - M$ [11] and $\boldsymbol{F} - \mathrm{M}$ [17] that measure disentanglement by examining the accuracy of a classifier that predicts the index of a fixed factor of variation, and **DCI** [8] that computes the entropy of the distribution obtained by normalizing the importance of each dimension of the learned representation for predicting the value of a factor of variation. The implementation details of these metrics follow the settings in [22].

4.1.4 Implementation Details. For the QM9, we adopt $L_B = 10, K_B = 1$ for the disentangled bond flow and $L_A = 27$ for the disentangled atom flow. We set all dimensions of $3 * 3$ convolution layer be 128, R-GCN layer with 64 dimensions and all MLPs are single layer with 128 dimensions. As for the VAE-encoder, we set two layers with 128 dimensions and output layer with $D = 32$ for GIN. For the ZINC-250K, we adopt $L_B = 10, K_B = 1$ and $L_A = 38$ for the disentangled bond flow and the disentangled atom flow respectively. We set all dimensions of $3 * 3$ convolution layer be 512, R-GCN layer with 64 dimensions and all MLPs are single layer with 128 dimensions. As for the VAE-encoder, we set two layers with 128 dimensions and output layer with $D = 256$ for GIN. We implement our DEMO on a workstation with 1 GeForce RTX 2080 Ti GPU and trained the model by Adam optimizer with learning rate 0.001, batch size 256 and 200 epochs for two datasets.

4.2 Molecular Graph Generation Quality

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. The results are shown in Table 1 and Table 2. The key observations can be summarized as follows.
1) Our model achieves the best validity w/o check accuracy and score accuracy on QM9, and the second-best validity w/o check accuracy and the best score accuracy on ZINC-250K, which demonstrates the excellent generative ability of our model.
2) Compared to the VAE-based models, DEMO as well as all other flow-based models achieve 100% reconstruction accuracy. At the same time, the flow-based models share significantly high uniqueness ratio. Such advanced performance is attributed to the invertible characteristic of normalizing flows.
3) Compared with the sequential flow-based model GraphAF, DEMO outperforms it in terms of the validity w/o check accuracy by a large margin, with around 16% and 0.9% improvements in the score accuracy for QM9 and ZINC-250K, respectively, these improvements are 2% and 0.8% respectively compared to GraphDF. The results imply the superiority of capturing the molecular graph structures in a one-shot way by our model over autoregressive ones in a sequential way. As for the slight performance deficit w.r.t the validity w/o check accuracy compared to GraphDF, this may be because it eliminates the negative effects of dequantization by sequentially mapping discrete latent variables to graph nodes and edges.
4) Compared with the one-shot flow-based models GraphNVP and GRF, Our model outperforms them by more than two times w.r.t. the score accuracy, since they are based on pure normalizing flow

Table 1: Generation & Reconstruction performance on QM9

	% **Validity**	% **Validity w/o check**	% **Uniqueness**	% **Novelty**	% **Score**	% **Reconstruct**
GraphVAE [33]	55.7	n/a	76.0	61.6	26.1	n/a
GraphNVP [27]	83.1 ± 0.5	n/a	99.2 ± 0.3	58.2 ± 1.9	47.97	100
GRF [13]	84.5 ± 0.70	n/a	66.0 ± 1.15	58.6 ± 0.82	32.68	100
GraphAF [32]	100	67	94.51	88.83	83.95	100
GraphDF [23]	100	82.67	97.62	98.1	95.77	100
MoFlow [42]	100.00 ± 0.00	96.17 ± 0.18	99.20 ± 0.12	98.03 ± 0.14	97.24 ± 0.21	100.00 ± 0.00
DEMO	100.00 ± 0.00	**96.74 ± 0.12**	**99.47 ± 0.09**	**98.20 ± 0.09**	**97.68 ± 0.01**	100.00 ± 0.00

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

	% **Validity**	% **Validity w/o check**	% **Uniqueness**	% **Novelty**	% **Score**	% **Reconstruct**
GraphVAE [33]	34.9	n/a	n/a	100	n/a	54.7
JT-VAE [16]	100	n/a	100	100	100	76.7
GraphNVP [27]	42.6 ± 1.6	n/a	94.8 ± 0.6	100	40.38	100
GRF [13]	73.4 ± 0.62	n/a	53.7 ± 2.13	100	39.42	100
GraphAF [32]	100	68	99.10	100	99.10	100
GraphDF [23]	100	**89.03**	99.16	100	99.16	100
MoFlow [42]	100.00 ± 0.00	81.76 ± 0.21	99.99 ± 0.01	100.00 ± 0.00	99.99 ± 0.01	100.00 ± 0.00
DEMO	100.00 ± 0.00	**86.97 ± 0.38**	**100.00 ± 0.00**	100.00 ± 0.00	**100.00 ± 0.00**	100.00 ± 0.00

models [7], validating the effectiveness of the components borrowed from Glow [18] and the validity correction module. As for MoFlow, although it achieves competitive performance on both datasets, DEMO is still better than it. Especially in ZINC-250K, we obtain a totally satisfied result, reflecting the positive effect of the VAE-encoder on the flow-generator.

4.3 Molecule Optimization

Molecule optimization has two branches generally. One is property optimization, which generates novel molecules with the best property scores. The other is constrained property optimization, which finds molecules that are as similar as possible to a given molecule while improving chemical properties as much as possible. We solve these problems in three steps. 1) train a DEMO model and fix it; 2) train a 3-layers MLP as a regressor on the latent space of molecules with the target chemical properties; 3) search the molecules with the best property score by gradient ascend between the output and the input of the learned regressor. For constrained property optimization, we additionally add the similarity constrain δ between the original molecules and the generated molecules. The optimization process fails when the similarity score is smaller than δ or the property score is not improved within n steps.

We choose two widely used metrics as the target property: *Quantitative Estimate of Druglikeness* (QED) [3] 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.Molecular similarity is measured by Tanimoto similarity [1] of Morgan fingerprint [30].

We conduct experiments of these two problems by sampling from the 800 molecules with the lowest property scores and summarize the discovered novel molecules sorted by QED scores in Table 3 and the constrained optimized results in Table 4. Table 3 shows that DEMO can optimize more molecules to achieve the highest QED scores compared to the previous methods, and have a comparable performance with MoFlow. Table 4 shows that compared with

Table 3: Optimized molecules with the best QED scores

Method	1st	2nd	3rd	4th
ZINC-250K	0.948	0.948	0.948	0.948
JT-VAE	0.925	0.911	0.910	-
GraphAF	0.948	0.948	0.947	0.946
GraphDF	0.948	0.948	0.948	0.946
MoFlow	**0.948**	**0.948**	**0.948**	**0.948**
DEMO	**0.948**	**0.948**	**0.948**	**0.948**

Table 4: Constrained plogP optimization on ZINC-250k

	JT-VAE			GraphAF		
δ	Improvement	Similarity	Success	Improvement	Similarity	Success
0.0	1.91±2.04	0.28±0.15	97.5%	13.13±6.89	0.29±0.15	100%
0.2	1.68±1.85	0.33±0.13	97.1%	11.90±6.86	0.33±0.12	100%
0.4	0.84±1.45	0.51±0.10	83.6%	8.21±6.51	0.49±0.09	99.88%
0.6	0.21±0.71	0.69±0.06	46.4%	4.98±6.49	0.66±0.05	96.88%
	MoFlow			DEMO		
δ	Improvement	Similarity	Success	Improvement	Similarity	Success
0.0	8.61±5.44	0.30±0.20	98.88%	5.57±5.34	**0.50±0.27**	98.88%
0.2	7.06±5.04	0.43±0.20	96.75%	5.09±5.14	**0.56±0.24**	94%
0.4	4.71±4.55	0.61±0.18	85.75%	3.31±3.88	**0.69±0.20**	82.75%
0.6	2.10±2.86	0.79±0.14	58.25%	1.59±2.32	**0.82±0.15**	59.13%

the state-of-the-art VAE model JT-VAE, our DEMO 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, which means that the learned distribution can cover more potential molecules, demonstrating the smoothness of the learned space.

We additionally show the top-5 results of constrained optimized molecules while maintaining a high similarity in Figure 4. 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. All these experiments demonstrate the power of DEMO in molecular optimization problems.

Table 5: MAE of properties prediction on QM9 based on the latent representation learned from different models. Gray line is the state-of-the-art semi-supervised method, others are unsupervised methods.

	mu	alpha	Homo	Lumo	*Gap*	r^2	Zpve	U_0	U	H	G	C_v
infoGraph*	0.3168	0.5444	0.1605	0.1659	0.2421	4.92	0.0004	0.1410	0.1702	0.1552	0.1592	0.1965
GraphVAE	0.7993	1.6698	0.0128	0.0129	0.0310	137.1390	0.0166	0.8440	0.8258	0.7794	0.7879	1.3306
moflow	0.5755	0.4517	0.0105	0.0136	0.0145	21.9235	0.0018	1.0289	0.9604	3.0895	0.9016	0.2368
DGI + DMI	0.6807	1.1179	**0.0071**	**0.0068**	**0.0107**	49.6120	0.0017	0.9914	0.7247	0.9805	1.0113	0.4744
DEMO	**0.5714**	**0.4177**	0.0085	0.0092	0.0120	**21.198**	**0.0010**	**0.7367**	**0.4230**	**0.7243**	**0.7641**	**0.1803**

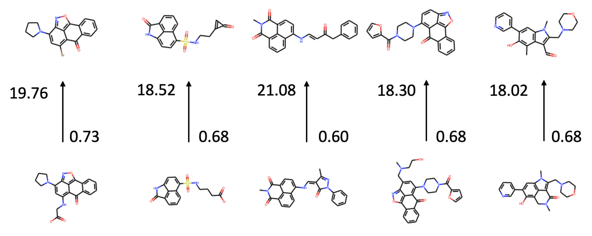


Figure 4: Constrained Property Optimization. The arrow points from original molecular to optimized molecular, numbers on the left and right side of the arrow denote the property improvement and similarity of the given molecule pairs respectively.

4.4 Disentanglement Performance

4.4.1 Impact on Downstream Tasks. One of the key motivations behind disentangled representations is that they can help improve performance on downstream tasks. In order to verify this point, we conduct molecule property prediction on QM9. We take the latent representations from the trained model and train a simple 3-layers MLP to predict molecular property. Besides GraphVAE and MoFlow, we combine DGI [35] on graph and DMI [12] on adjacency matrix to extract latent representation as the SOTA unsupervised method, and InfoGraph [34] as the SOTA semi-supervised method. We use 10,000 molecules for testing and others for training. The Mean-Absolute-Error (MAE) values are reported in Table 5. The results of InfoGraph we take from [34].

Table 5 shows that the learned latent representations from our model has a better performance in downstream property prediction task than GraphVAE and MoFlow. Most of the results exceeded the SOTA unsupervised method DGI+DMI. Moreover, we even perform better than SOTA semi-supervised method in some properties such as alpha and C_v, which demonstrate the superiority of our model.

4.4.2 Disentanglement Evaluation. We conduct experiments on disentanglement metrics and compare them to GraphVAE and MoFlow, which can be roughly considered as our ablation models. Table 6 shows that our model achieves the best overall disentanglement scores on both datasets.

In it, DEMO and MoFlow achieve 100% on the $\beta - \mathrm{M}$ score on both datasets. For $\mathrm{F} - \mathrm{M}$ score on both datasets, DEMO performs the best, followed by GraphVAE and MoFlow in this order. For the DCI score, MoFlow performs better than GraphVAE, while DEMO performing better than both. Performing better than GraphVAE indicates the flow-generator indeed has a positive effect on disentanglement, demonstrating the effectiveness of our model. Outperforming MoFlow shows that DEMO can map latent factors into a more powerful disentangled space compared to employing pure flow-based models. Besides, it is worth noting that MoFlow is not friendly to obtaining DCI scores for ZINC-250k, as flow-based models require the input and output dimensions of the model to be the same. In our case, we have to face complex calculations with 6156 (=38*38*4+38*10) dimensions, which spent us more than one day on it. All of these prove the superiority of our model in learning disentangled factors, while guaranteeing a competitive performance on generation task under a fully unsupervised manner.

Table 6: Disentanglement scores on QM9 and ZINC-250K

Dataset	Model	$\beta - \mathrm{M}(\%)$	$\mathrm{F} - \mathrm{M}(\%)$	DCI
QM9	GraphVAE	12	34	0.32
	MoFlow	100	16	0.39
	DEMO	**100**	**34**	**0.47**
ZINC-250K	GraphVAE	16	20	0.32
	MoFlow	100	14	0.37
	DEMO	**100**	**36**	**0.40**

4.4.3 Interpreting Generated Molecules w.r.t. the Disentangled Factors. In order to interpret the generated molecular results w.r.t. the learned disentangled factors, we visualize the generated molecular results from both microscopic and macroscopic perspectives.

At the microscopic level, we visualize the generated results during the generation process step by step to bridge the connection between molecular substructures and their properties, while showing the differences caused by manipulating one dimension of the disentangled factors. Specifically, we first draw a sample from a standard Gaussian distribution and copy it n times. Then we randomly select one dimension and fix all others, assigning linearly varying values from the interval $\mathbb{B}$ to the selected dimension of these n vectors. As we obtained $\mathbf{z_{vae}}$, we draw a random sample from a spherical multivariate Gaussian distribution and replicate it n times as $\mathbf{z_{flow}}$, and feed them into DEMO to generate molecules. During the generation process, we first generate the adjacency matrix A^*, then output the results after every C atom affine coupling functions as the feature matrix X^*, and visualize molecules generated in all intermediate processes.

At the macroscopic level, instead of replicating one random sample, we draw different random samples from a spherical multivariate Gaussian distribution. Skipping the intermediate processes, we visualize the final results of different samples by manipulating one dimension of the disentangled factors.

Figure 5 and Figure 6 shows the generated molecules of the microscopic level and their corresponding QED scores on QM9 and

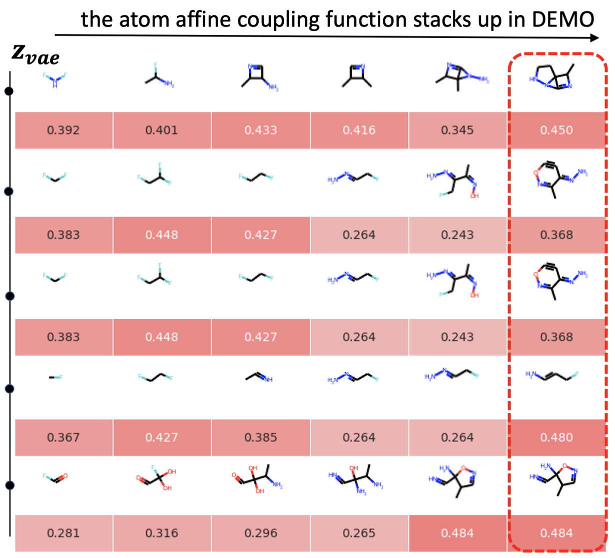


Figure 5: Generated molecules and the QED scores under the stack of the atom affine coupling function while manipulating one dimension of the disentangled factors on QM9.

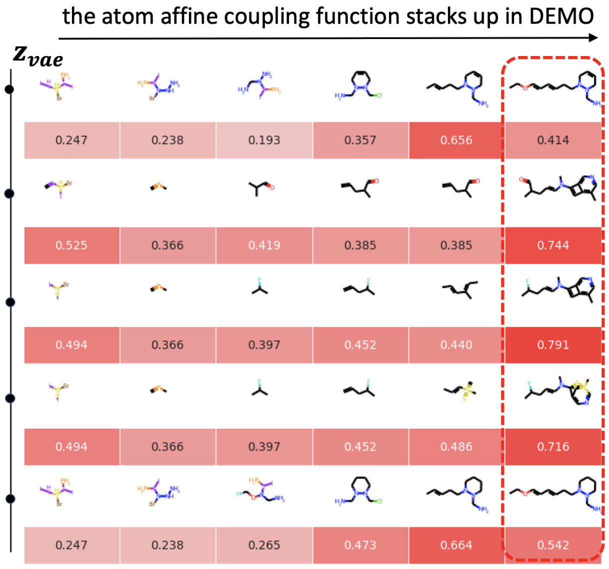


Figure 6: Generated molecules and the QED scores under the stack of the atom affine coupling function while manipulating one dimension of the disentangled factors on ZINC-250K.

ZINC-250K separately. In it, we set $n = 5$ and $\mathbb{B} \in [-30, 30)$ for both dataset. Setting $C = 4$ for QM9 and $C = 7$ for ZINC-250K for equally visualization (since L_A=27 for QM9 and L_A=38 for ZINC-250K, see Section 4.1.4).

The horizontal change from left to right generally shows molecular structures from simple with a small number of atoms to complex, as the atom affine coupling function stacks up in DEMO. Molecules in one state are likely to become substructures of molecules in the next state when stacking up by layers. The QED values of the molecules also show an increasing trend, although not strictly

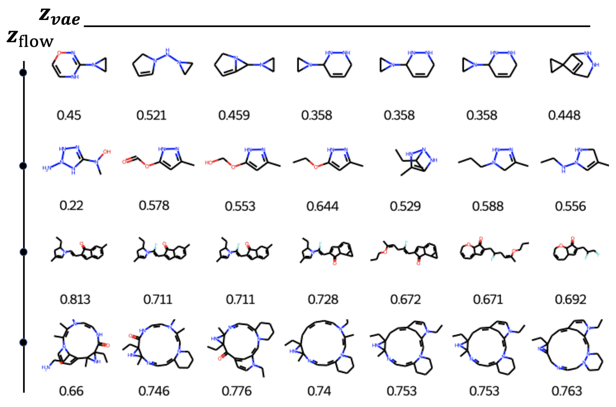


Figure 7: Generated molecular graphs and the QED scores for different $\mathbf{z}_{\text{flow}}$ under manipulating one dimension of $\mathbf{z}_{\text{vae}}$

monotonically. This process presented by the heatmap clearly reflects the rationality of our model and builds the connection between molecule substructures and their properties. Along the vertical direction when checking only the last column (the other columns are intermediate results), we find generated molecules by manipulating $\mathbf{z}_{\text{vae}}$ have similar substructures or property values. For QM9, they share similar QED values. For ZINC-250K, they share similar substructures, whose carbon chain structure and ring structure have minor changes at their junctions. The results are consistent with the general results of manipulating the disentangled factors, changing only the local information without affecting the global information, indicating that the disentangled factors learned by our model are effective.

We also visualize the results and observe the changes of the generated molecules at the macroscopic level. We set $n = 5$ and $n = 7$, and show the results in Figure 1 and Figure 7 respectively. For Figure 7, the first two rows are conducted on QM9 and the last two rows are performed on ZINC-250K. The molecules at each line share similar semantic-level features (structures), accompanied by subtle differences in syntactic-level features (substructures), while molecules from one line to another have different structural patterns. All these observations in Figure 1 and Figure 7 demonstrate the validity of our learned disentangled factors.

5 CONCLUSIONS

We have studied the problem of interpretable molecular graph generation, and proposed DEMO, a disentangled molecular graph generative model. DEMO is not only the first interpretable molecular graph generative model but also offers a new prospective for dealing with disentangled representation learning. The evaluation of generation capability and interpretability have verified the effectiveness of our model. An interesting direction for future work is to let explicit labels participate in the molecular graph generation model to give a clear chemical interpretation.

6 ACKNOWLEDGMENTS

This work is supported by the King Abdullah University of Science and Technology (KAUST), Saudi Arabia, and NSF Center for Computer-Assisted Synthesis (award no. 2202693).

REFERENCES

[1] 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.

[2] Yoshua Bengio, Aaron Courville, and Pascal Vincent. 2013. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence* 35, 8 (2013), 1798–1828.

[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), 90–98.

[4] Christopher P Burgess, Irina Higgins, Arka Pal, Loic Matthey, Nick Watters, Guillaume Desjardins, and Alexander Lerchner. 2018. Understanding disentangling in $beta$-VAE. *arXiv preprint arXiv:1804.03599* (2018).

[5] Ricky TQ Chen, Xuechen Li, Roger Grosse, and David Duvenaud. 2018. Isolating sources of disentanglement in variational autoencoders. *arXiv preprint arXiv:1802.04942* (2018).

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

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

[8] Cian Eastwood and Christopher KI Williams. 2018. A framework for the quantitative evaluation of disentangled representations. In *ICLR*.

[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] Peng Han, Peilin Zhao, Chan Lu, Junzhou Huang, Jiaxiang Wu, Shuo Shang, Bin Yao, and Xiangliang Zhang. 2022. GNN-Retro: Retrosynthetic Planning with Graph Neural Networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 36. 4014–4021.

[11] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. 2016. beta-vae: Learning basic visual concepts with a constrained variational framework. (2016).

[12] R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. 2018. Learning deep representations by mutual information estimation and maximization. *arXiv preprint arXiv:1808.06670* (2018).

[13] 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).

[14] 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.

[15] Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. 2018. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *CVPR*. 2704–2713.

[16] Wengong Jin, Regina Barzilay, and Tommi Jaakkola. 2018. Junction tree variational autoencoder for molecular graph generation. *arXiv preprint arXiv:1802.04364* (2018).

[17] Hyunjik Kim and Andriy Mnih. 2018. Disentangling by factorising. In *ICML*. PMLR, 2649–2658.

[18] Durk P Kingma and Prafulla Dhariwal. 2018. Glow: Generative flow with invertible 1x1 convolutions. In *NeurIPS*. 10215–10224.

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

[20] Abhishek Kumar, Prasanna Sattigeri, and Avinash Balakrishnan. 2017. Variational inference of disentangled latent concepts from unlabeled observations. *arXiv preprint arXiv:1711.00848* (2017).

[21] Yanbei Liu, Xiao Wang, Shu Wu, and Zhitao Xiao. 2020. Independence promoted graph disentangled networks. In *AAAI*, Vol. 34. 4916–4923.

[22] Francesco Locatello, Stefan Bauer, Mario Lucic, Gunnar Raetsch, Sylvain Gelly, Bernhard Schölkopf, and Olivier Bachem. 2019. Challenging common assumptions in the unsupervised learning of disentangled representations. In *ICML*. PMLR, 4114–4124.

[23] Youzhi Luo, Keqiang Yan, and Shuiwang Ji. 2021. GraphDF: A discrete flow model for molecular graph generation. *arXiv preprint arXiv:2102.01189* (2021).

[24] Changsheng Ma and Xiangliang Zhang. 2021. GF-VAE: A Flow-based Variational Autoencoder for Molecule Generation. In *CIKM*. 1181–1190.

[25] Jianxin Ma, Peng Cui, Kun Kuang, Xin Wang, and Wenwu Zhu. 2019. Disentangled graph convolutional networks. In *ICML*. PMLR, 4212–4221.

[26] Tengfei Ma, Jie Chen, and Cao Xiao. 2018. Constrained generation of semantically valid graphs via regularizing variational autoencoders. In *NeurIPS*. 7113–7124.

[27] 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).

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

[29] 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.

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

[31] 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.

[32] 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).

[33] 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.

[34] Fan-Yun Sun, Jordan Hoffmann, Vikas Verma, and Jian Tang. 2019. Infograph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization. *arXiv preprint arXiv:1908.01000* (2019).

[35] Petar Veličković, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. 2018. Deep graph infomax. *arXiv preprint arXiv:1809.10341* (2018).

[36] Satosi Watanabe. 1960. Information theoretical analysis of multivariate correlation. *IBM Journal of research and development* 4, 1 (1960), 66–82.

[37] 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.

[38] Xiaolin Xia, Jianxing Hu, Yanxing Wang, Liangren Zhang, and Zhenming Liu. 2020. Graph-based generative models for de Novo drug design. *Drug Discovery Today: Technologies* (2020).

[39] Shufang Xie, Rui Yan, Peng Han, Yingce Xia, Lijun Wu, Chenjuan Guo, Bin Yang, and Tao Qin. 2022. RetroGraph: Retrosynthetic Planning with Graph Search. *KDD 2022* (2022).

[40] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2018. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826* (2018).

[41] 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.

[42] Chengxi Zang and Fei Wang. 2020. MoFlow: an invertible flow model for generating molecular graphs. In *SIGKDD*. 617–626.