GraphNVP: An Invertible Flow-based Model for Generating Molecular Graphs

Anonymous authors
Paper under double-blind review

Abstract

We propose GraphNVP, an invertible flow-based molecular graph generation model. Existing flow-based models only handle node attributes of a graph with invertible maps. In contrast, our model is the first invertible model for the *whole graph components*: both dequantized node attributes and adjacency tensor are converted into latent vectors through two novel invertible flows. This decomposition yields the exact likelihood maximization on graph-structured data. We decompose the generation of a graph into two steps: generation of (i) an adjacency tensor and (ii) node attributes. We empirically demonstrate that our model and the two-step generation efficiently generates valid molecular graphs with almost no duplicated molecules, although there are no domain-specific heuristics ingrained in the model. We also confirm that the sampling (generation) of graphs is faster in order of magnitude than other models in our implementation. In addition, we observe that the learned latent space can be used to generate molecules with desired chemical properties.

1 Introduction

Generation of molecules with certain desirable properties is a crucial problem in computational drug discovery. Recently, deep learning approaches are being actively studied for generating promising candidate molecules quickly. Earlier models (Kusner et al., 2017; Gómez-Bombarelli et al., 2018) depend on a string-based representation of molecules. However, recent models (Jin et al., 2018; You et al., 2018a; De Cao & Kipf, 2018) directly work on molecular graph representations and record impressive experimental results. In these studies, either variational autoencoder (VAE) (Kingma & Welling, 2014) or generative adversarial network (GAN) (Goodfellow et al., 2014; Radford et al., 2015) are used mainly to learn mappings between the graphs and their latent vector representations.

In this paper, we propose *GraphNVP*, yet another framework for molecular graph generation based on the invertible normalizing flow, which was mainly adopted for image generation tasks (Dinh et al., 2017; Kingma & Dhariwal, 2018). To capture distributions of irregular graph structure of molecules into a latent representation, we propose a novel two-step generation scheme. Specifically, GraphNVP is equipped with two latent representations for a molecular graph: first for the graph structure represented by an adjacency tensor, and second for node (atom) attributes. We introduce two types of reversible flows that work for the aforementioned two latent representations of graphs.

Recent work by Liu et al. (2019) proposes a flow-based invertible model for transforming the node attribute matrix. However, they use a non-invertible encoder for transforming the adjacency tensor making the complete model non-invertible. Our model is the first fully invertible model for the *whole graph components*: both adjacency tensor and node attributes are converted into latent vectors through two novel invertible flows.

To sample a graph, we develop a novel two-step generation process. During the generation process, GraphNVP first generates the graph structure. Then node attributes are generated according to this structure. This two-step generation enables efficient generation of valid molecular graphs. The full reversibility of our model on graphs contributes to two major benefits: a simple architecture and precise log-likelihood maximization. A major advantage of invertible models is that we do not need to design a separate decoder for sample generation: new graph samples can be generated by simply feeding a latent vector into the same model but in the reverse order.

In contrast, VAE models require an encoder and a separated decoder. Decoding processes of several VAE graph generators are often quite complicated to assure valid generations (Kusner et al., 2017; Jin et al., 2018; Ma et al., 2018), and computing a graph reconstruction loss may require expensive graph matching (Simonovsky & Komodakis, 2018). The lack of an encoder in GAN models makes it challenging to manipulate the sample generation. For example, it is not straightforward to use a GAN model to generate graph samples that are similar to a query graph (e.g., lead optimization for drug discovery), while it is easy for flow-based models.

Unlike VAEs and GANs, invertible models are capable of precise log-likelihood evaluation. We believe precise optimization is crucial in molecule generation for drugs, which are highly sensitive to a minor replacement of a single atom (node).

In the experiments, we compare the proposed flow model with several existing graph generation models using two popular molecular datasets. The proposed flow model generates molecular graphs with almost 100% uniqueness ratio: namely, the results contain almost no duplicated molecular graphs without ingrained domain expert knowledge and extra validity checks. The proposed model enjoys fast graph samplings; faster in orders of magnitude than other graph generation models in our implementation. Additionally, we show that the learned latent space can be utilized to generate molecular graphs with desired chemical properties, even though we do not encode domain expert knowledge into the model.

2 Related Work

2.1 Molecular Graph Generation

We can classify the existing molecular graph generation models based on how the data distribution is learned. Most current models belong to two categories. First, VAE-based models assume a simple variational distribution for latent representation vectors (Jin et al., 2018; Liu et al., 2018; Ma et al., 2018). Second, some models implicitly learn the empirical distribution, especially based on the GAN architecture (e.g., (De Cao & Kipf, 2018; You et al., 2018a; Guimaraes et al., 2017)). Some may resort to reinforcement learning (You et al., 2018a) to alleviate the difficulty of direct optimization of the objective function. We also observe an application of autoregressive recurrent neural networks (RNN) for graphs (You et al., 2018b). In this paper, we will add a new category to this list: namely, the invertible flow.

Additionally, we can classify the existing models based on the process they use for generating a graph. There are mainly two choices in the generation process. One is a sequential *iterative* process, which generates a molecule in a step-by-step fashion by adding nodes and edges one by one (Jin et al., 2018; You et al., 2018a). The alternative is *one-shot* generation of molecular graphs, when the graph is generated in a single step. This process resembles commonly used image generation models (e.g., (Kingma & Dhariwal, 2018)). The former process is advantageous in (i) dealing with large molecules and (ii) forcing validity constraints on the graph (e.g., a valency condition of molecule atoms). The latter approach has a major advantage: the model is simple to formulate and implement. This is because the one-shot approach does not have to consider arbitrary permutations of the sequential steps, which can grow exponentially with the number of nodes in the graph.

Combining these two types of classification, we summarize the current status of molecular graph generation in Table 1. In this paper, we propose the first graph generation model based on the invertible flow, with one-shot generation strategy.

2.2 Invertible Flow Models

To the best of our knowledge, the invertible flow was first introduced to the machine learning community by (Tabak & Vanden-Eijnden, 2010; Tabak & Turner, 2013). Later, Rezende & Mohamed (2015) and Dinh et al. (2015) leveraged deep neural networks in defining tractable invertible flows. Dinh et al. (2015) introduced reversible transformations for which the log-determinant calculation is tractable. These transformations, known as *coupling layers*, serve as the basis of recent flow-based image generation models (Dinh et al., 2017; Kingma & Dhariwal, 2018; Grathwohl et al., 2019)

Readers are referred to the latest survey (Kobyzev et al., 2019) for the general flow methodologies.

Name	Distribution Model					Generation Process	
	VAE	Adversarial	RL	RNN	Inv.Flow	Iterative	OneShot
RVAE (Ma et al., 2018)	✓	-	-	-	-	-	✓
CGVAE (Liu et al., 2018)	✓	-	-	-	-	✓	-
JT-VAE (Jin et al., 2018)	✓	-	-	-	-	✓	-
MolGAN (De Cao & Kipf, 2018)	-	✓	-	-	-	-	✓
GCPN (You et al., 2018a)	-	✓	✓	-	-	✓	-
GraphRNN (You et al., 2018b)	-	-	-	✓	-	✓	-
GraphNVP	-	-	-	-	✓	-	✓

Table 1: Existing models of molecular graph generation. We propose the first invertible flow-based graph generation model in the literature..

So far, the application of flow-based models is mostly limited to the image domain. As a few exceptions, Kumar et al. (2018) proposed flow-based invertible transformations on graphs. However, their model is only capable of modeling the node assignments and cannot learn a latent representation of the adjacency tensor; therefore, it cannot generate a graph structure. Liu et al. (2019) proposed to plug a non-invertible decoder for the adjacency tensor to this flow model afterwards, giving up training the entire graph generator in a single unified estimator. We overcome this issue by introducing two latent representations, one for node assignments and another for the adjacency tensor, to capture the unknown distributions of the graph structure and its node assignments. Thus, we consider our proposed model to be the first invertible flow-based model that can generate attributed graphs including the adjacency structure.

3 GraphNVP: Flow-based graph generation model

3.1 Formulation

We use the notation $G = (A, X)$ to represent a graph G consisting of an adjacency tensor A and a feature matrix X. Let there be N nodes in the graph. Let M be the number of types of nodes and R be the number of types of edges. Then $A \in \{0,1\}^{N \times N \times R}$ and $X \in \{0,1\}^{N \times M}$. In the case of molecular graphs, $G = (A, X)$ represents a molecule with R types of bonds (single, double, etc.) and M types of atoms (e.g., oxygen, carbon, etc.). Our objective is to learn an invertible model f_θ with parameters θ that maps G into a latent point $z = f_\theta(G) \in \mathbb{R}^{D=(N \times N \times R)+(N \times M)}$. We describe f_θ as a normalizing flow composed of multiple invertible functions.

Let z be a latent vector drawn from a known prior distribution $p_z(z)$ (e.g., Gaussian): $z \sim p_z(z)$. With the change of variable formula, the log probability of a given graph G can be calculated as:

$$\log\left(p_G(G)\right) = \log\left(p_z(z)\right) + \log\left(\left|\det\left(\frac{\partial z}{\partial G}\right)\right|\right), \tag{1}$$

where $\frac{\partial z}{\partial G}$ is the Jacobian of f_θ at G.

3.2 Graph Representation

Directly applying a continuous density model on discrete components may result in degenerate probability distributions. Therefore, we cannot directly employ the change of variable formula (Eq. 1) for these components. The same issue, especially modeling the discrete structure of the adjacency A, has been a problem in existing one-shot generators based on GAN (De Cao & Kipf, 2018) and VAE (Ma et al., 2018). They resort to an ad-hoc workaround; treating the adjacency tensor as a real-valued continuous tensor. In this paper we take another approach, *dequantization* (Theis et al., 2016), following the flow-based image generation models (Dinh et al., 2017; Kingma & Dhariwal, 2018). The dequantization process adds uniform noises to A and X and yield the dequantized graph component $G' = (A', X')$. Specifically, $A' = A + cu$; $u \sim U[0,1)^{N \times N \times R}$ and $X' = X + cu$; $u \sim U[0,1)^{N \times M}$, where $0 < c < 1$ is a scaling hyperparameter ($c = 0.9$ is adopted for our experiment). This G' is used as the input in Eq. 1. Note that the original discrete inputs A and X can be recovered by *quantization*: simply applying floor operation on each continuous value in A' and X'.

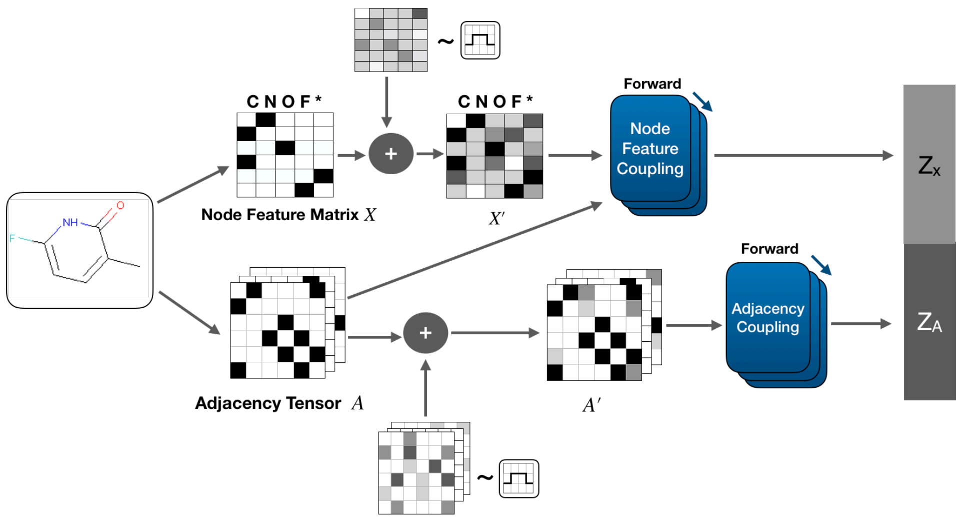

Figure 1: Forward transformation of the proposed GraphNVP. The original discrete A and X are first dequantized into A' and X'. Layers of invertible couplings are applied to these dequantized tensors.

Hereafter, all the transformations consisting f_θ are performed on dequantized inputs A' and X', not on A and X. It means f_θ is a bijective function that maps $G' \to z$: thus $f^{-1}(z)$ returns the dequantized G', not the original G. However, our generative model can recover the original discrete G by performing the postprocessing quantization to inverted G'.

There are a few works related to discrete invertible flows such as (Hoogeboom et al., 2019; Tran et al., 2019). The former maps discrete data x to a discrete latent space. However, we prefer a smoothly distributed continuous latent space for molecule decoration and optimization applications (see Sec. 4.3). The latter can map discrete data x to a continuous z, but computation includes approximation. Approximated likelihood evaluations decreases the significance of the invertible flows against VAEs. So we do not adopt these options in this paper.

3.3 Coupling Layers

Based on real-valued non-volume preserving (real NVP) transformations introduced in (Dinh et al., 2017), we propose two types of reversible affine coupling layers; *adjacency coupling* layers and *node feature coupling* layers that transform the adjacency tensor A' and the feature matrix X' into latent representations, $z_A \in \mathbb{R}^{N\times N\times R}$ and $z_X \in \mathbb{R}^{N\times M}$, respectively.

We apply L_X layers of node feature coupling layers to a feature matrix X' to obtain z_X. We denote an intermediate representation of the feature matrix after applying the ℓ^{th} node feature coupling layer as $z_X^{(\ell)}$. Starting from $z_X^{(0)} = X'$, we repeat updating rows of z_X over L_X layers. Each row of $z_X^{(\ell)}$ corresponds to a feature vector of a node in the graph. Finally, we obtain $z_X = z_X^{(L_X)}$ as the final latent representation of the feature matrix. The ℓ^{th} node feature coupling layer updates a single row ℓ of the feature matrix while keeping the rest of the input intact:

$$z_X^{(\ell)}[\ell,:] \leftarrow z_X^{(\ell-1)}[\ell,:] \odot \exp\left(s(z_X^{(\ell-1)}[\ell^-,:],A)\right) + t(z_X^{(\ell-1)}[\ell^-,:],A), \tag{2}$$

where functions s and t stand for scale and translation operations, and $\odot$ denotes element-wise multiplication. We use $z_X[\ell^-,:]$ to denote a latent representation matrix of X' excluding the ℓ^{th} row (node). Rest of the rows of the feature matrix will stay the same as

$$z_X^{(\ell)}[\ell^-,:] \leftarrow z_X^{(\ell-1)}[\ell^-,:]. \tag{3}$$

Both s and t can be formulated with arbitrary nonlinear functions, as the reverse step of the model does not require inverting these functions. Therefore, we use the graph adjacency tensor A when computing invertible transformations of the node feature matrix X'. So as functions s and t in a node feature coupling layer, we use a sequence of generic graph neural networks. It should be noted that

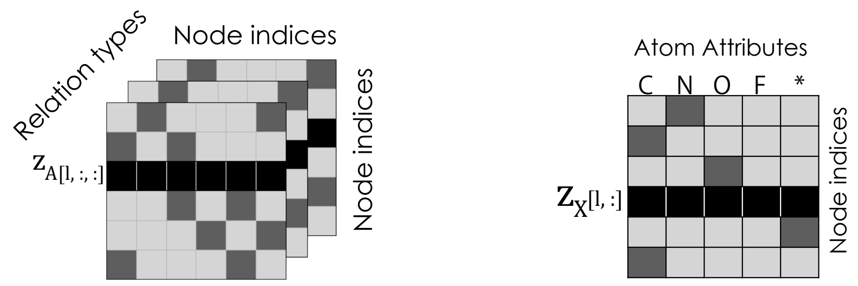

Figure 2: Masking schemes used in proposed affine coupling layers. Left: Adjacency coupling layer: A single row of adjacency tensor is masked. Right: Node feature coupling layer: All channels belonging to a single node are masked.

we use the discrete adjacency tensor A, as only the node feature matrix is updated in this step. In this paper, we use a variant of Relational GCN (Schlichtkrull et al., 2018) architecture.

Likewise, we apply L_A layers of transformations for the adjacency tensor A' to obtain the latent representation z_A. We denote an intermediate representation of the adjacency tensor after applying the ℓ^{th} adjacency coupling as $z_A^{(\ell)}$. The ℓ^{th} adjacency coupling layer updates only a single slice of z_A^{ℓ} with dimensions $N \times R$ as:

$$z_A^{(\ell)}[\ell,:,:] \leftarrow z_A^{(\ell-1)}[\ell,:,:] \odot \exp\left(s(z_A^{(\ell-1)}[\ell^-,:,:])\right) + t(z_A^{(\ell-1)}[\ell^-,:,:]). \tag{4}$$

The rest of the rows will stay as it is:

$$z_A^{(\ell)}[\ell^-,:,:] \leftarrow z_A^{(\ell-1)}[\ell^-,:,:]. \tag{5}$$

For the adjacency coupling layer, we adopt multi-layer perceptrons (MLPs) for s and t functions. Starting from $z_A^{(0)} = A'$, we repeat updating the first axis slices of z_A over L_A layers. Finally, we obtain $z_A = z_A^{(L_A)}$ as the final latent representation of the adjacency tensor.

3.3.1 Masking Patterns and Permutation over Nodes

Eqs. (2, 4) are implemented with masking patterns shown in Figure 2. Based on experimental evidence, we observe that masking $z_A(A')$ and $z_X(X')$ w.r.t. the node axis performs the best. Because a single coupling layer updates one single slice of z_A and z_X, we need a sequence of N coupling layers at the minimum, each masking a different node, for each of the adjacency coupling and the node feature coupling layers.

We acknowledge that this choice of masking axis over z_X and z_A makes the transformations not invariant to permutations of the nodes. We can easily formulate permutation-invariant couplings by changing the slice indexing based on the non-node axes (the 3rd axis of the adjacency tensor, and the 2nd axis of the feature matrix). However, using such masking patterns results in dramatically worse performance due to the sparsity of molecular graphs. For example, organic compounds are mostly made of carbon atoms. Thus, masking the carbon column in X' (and z_X) results in feeding a nearly-empty matrix to the scale and the translation networks, which is almost non-informative to update the carbon column entries of X' and z_X. We consider this permutation dependency as a limitation of the current model, and we intend to work on this issue as future work.

3.4 Training

During the training, we perform the forward computations shown in Figure 1 over minibatches of training data ($G = (A, X)$) and obtain latent representations $z = \text{concat}(z_A, z_X)$. Our objective is maximizing the log likelihood (Eq. 1) of z over minibatches of training data. This is implemented as minimization of the negative log likelihood using the Adam optimizer (Kingma & Ba, 2015).

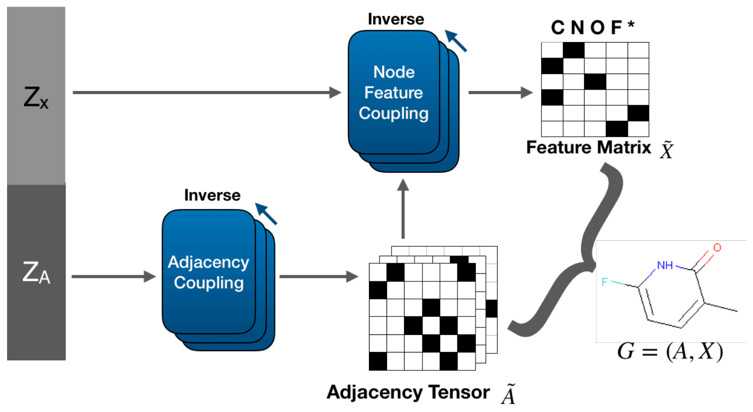

Figure 3: Generative process of the proposed GraphNVP. We apply the inverse of the coupling layers in the reverse order, so that the dequantized inputs $\tilde{A}$ and $\tilde{X}$ are recovered. Additional quantization reconstructs the original discrete graph, $G = (A, X)$.

3.5 Two-step Molecular Graph Generation

Because our proposed model is invertible, graph generation is simply executing the process shown in Figure 1 in reverse. During the training, node feature coupling and adjacency coupling can be performed in either order, as the output of one coupling module does not depend on the output of the other coupling module. However, because the node feature coupling module requires a valid adjacency tensor as an input, we also need an adjacency tensor to perform the reverse step of node feature coupling. Therefore, we apply the reverse step of adjacency coupling module first, so we get an adjacency tensor as the output. Next, the adjacency tensor is fed into the reverse step of the node feature coupling. The generation process is shown in Figure 3. In section 4, we show that this *2-step generation process* can efficiently generate chemically valid molecular graphs.

1st step: We draw a random sample $z = \text{concat}(z_A, z_X)$ from prior p_z and split sampled z into z_A and z_X. Next, we apply a sequence of *inverted* adjacency coupling layers on z_A. As a result, we obtain a probabilistic adjacency tensor $\tilde{A}'$, from which we construct a discrete adjacency tensor $\tilde{A} \in \{0,1\}^{N \times N \times R}$ by taking node-wise and edge-wise argmax.

2nd step: We generate a feature matrix given the sampled z_X and the generated adjacency tensor $\tilde{A}$. We input $\tilde{A}$ along with z_X into a sequence of *inverted* node feature coupling layers to attain $\tilde{X}'$. Likewise, we take node-wise argmax of $\tilde{X}'$ to get discrete feature matrix $\tilde{X} \in \{0,1\}^{N \times M}$.

4 Experiments

4.1 Procedure

We use two popular chemical molecular datasets, QM9 (Ramakrishnan et al., 2014) and ZINC-250k (Irwin et al., 2012). QM9 dataset contains 134k molecules, and ZINC-250k is made of 250k drug-like molecules randomly selected from the ZINC database. The maximum number of atoms in a molecule are 9 for the QM9 and 38 for the ZINC, respectively (excluding hydrogen). Following a standard procedure, we first kekulize molecules and then remove hydrogen atoms from them. The resulting molecules contain only single, double, and triple bonds.

We convert each molecule to an adjacency tensor $A \in \{0,1\}^{N \times N \times R}$ and a feature matrix $X \in \{0,1\}^{N \times M}$. N is the maximum number of atoms a molecule in a certain dataset can have. If a molecule has less than N atoms, we insert virtual nodes as padding to keep the dimensions of A and X the same for all the molecules. Because the original adjacency tensors can be sparse, we add a virtual bond edge between the atoms that do not have a bond in the molecule. Thus, an adjacency tensor consists of $R = 4$ adjacency matrices stacked together, each corresponding to the existence

Method	QM9				ZINC			
	% V	% N	% U	% R	% V	% N	% U	% R
GraphNVP	83.1 ($\pm$ 0.5)	58.2 ($\pm$ 1.9)	99.2 ($\pm$ 0.3)	100.0	42.6 ($\pm$ 1.6)	100.0 ($\pm$ 0.0)	94.8 ($\pm$ 0.6)	100.0
RVAE	96.6	97.5	-	61.8	34.9	100.0	-	54.7
MolGAN	98.1	94.2	10.4	-	-	-	-	-
GVAE	60.2	80.9	9.3	96.0	7.2	100.0	9.0	53.7
CVAE	10.3	90.0	67.5	3.6	0.7	100.0	67.5	44.6
JT-VAE	-	-	-	-	100.0	100.0	100.0	76.7
CG-VAE	100.0	94.4	98.6	-	100.0	100.0	99.8	-

Table 2: Performance of generative models with respect to quality metrics. Baseline scores are borrowed from the original papers. Zinc results of JT-VAE are reproduced based on the settings written in the original paper. Scores of GraphNVP are averages over 5 runs. Standard deviations are presented below the average scores.

of a certain type of bond (single, double, triple, and virtual bonds) between the atoms. The feature matrix is used to represent the type of each atom (e.g., oxygen, fluorine, etc.).

We use a multivariate Gaussian distribution $\mathcal{N}(\mathbf{0}, \sigma^2 \boldsymbol{I})$ as prior distribution $p_z(z)$, where standard deviation σ is learned simultaneously during the training. More details are presented in the appendix.

4.2 Numerical Evaluation

Following (Kingma & Dhariwal, 2018), we sample 1,000 latent vectors from a temperature-truncated normal distribution $p_{z,T}(z)$ (see the appendix for details) and transform them into molecular graphs by performing the reverse step of our model. We compare the performance of the proposed model with baseline models in Table 2 using following metrics. **Validity (V)** is the percentage of generated graphs corresponding to valid molecules. **Novelty (N)** is the percentage of generated valid molecules not present in the training set. **Uniqueness (U)** is the percentage of unique valid molecules out of all generated molecules. **Reconstruction accuracy (R)** is the percentage of molecules that can be reconstructed perfectly by the model: namely, the ratio of molecules G s.t. $G = f_\theta^{-1}(f_\theta(G))$.

We choose Regularizing-VAE (RVAE) (Ma et al., 2018) and MolGAN (De Cao & Kipf, 2018) as baseline one-shot generation models. We compare with two additional models: grammar VAE(GVAE) (Kusner et al., 2017) and character VAE (CVAE)(Gómez-Bombarelli et al., 2018), which learn to generate string representations of molecules. Finally, JT-VAE (Jin et al., 2018) and CG-VAE (Ma et al., 2018) as the state-of-the-art iterative generation models with complicated decoders with validity checkers.

Notably, proposed GraphNVP guarantees 100% reconstruction accuracy, attributed to the invertible function construction of normalizing flows. Also, it is notable that GraphNVP enjoys a significantly high uniqueness ratio. Although some baselines exhibit a higher validity on QM9 dataset, the set of generated molecules contains many duplicates. Additionally, we want to emphasize that our model generates a substantial number of valid molecules without explicitly incorporating the chemical knowledge as done in some baselines (e.g., valency checks for chemical graphs in RVAE, MolGAN, JT-VAE, and CG-VAE. This is preferable because additional validity checks consume computational time (see Sec.4.2.1), and may result in a low reconstruction accuracy (e.g., RVAE and JT-VAE). As GraphNVP does not incorporate domain-specific procedures during learning, it can be easily used for learning generative models on general graph structures. Two iterative generation models, JT-VAE (Jin et al., 2018) and CG-VAE (Liu et al., 2018), show great results in the table. However, decoders of these models are quite complicated to properly implement and reproduce the same performance. In contrast, the proposed GraphNVP enjoys a simple network architecture and its decoder is immediately available by just inverting the trained coupling layers.

Considering the simplicity of the model, proposed GraphNVP achieves good performance among latest graph generation models. We guess the generation scheme of the GraphNVP may affect these performances in part. The proposed generation scheme is in the midst of the one-shot and the iterative graph generation. From a higher perspective, our generation is one-shot: once we sample the latent vector $z = [z_X, z_A]$ then the final output graph is determined. In a detailed observation, the inversion process is iterative: for each (inverting) ℓ-th layer of two couplings, the network recovers a adjacency matrix or a feature vector of a ℓ-th node, given representations of all the nodes except ℓ-th node. One layer of partition-based affine coupling is not a mapping of super-flexible, but may be flexible enough to warp a single node's representation.

4.2.1 Computational Time for Graph Generation

One practically important aspect of graph generation is computational time. Training and sampling a generative model is much faster than wet-lab experiments, but the computational time is still an issue for tasks involving huge search spaces: e.g. drug search. We compare the computational time (wall-clock time) for sampling $1,000$ graphs for ZINC dataset experiment runs. The average wall-clock time (excluding preprocessing time) of GraphNVP for sampling is only 4.6 [sec] (implemented in Chainer Tokui et al. (2015)). This is faster in order of magnitude than several baselines (in our test runs): 193.5 [sec] for CVAE (Tensorflow), 460 [sec] for GVAE (Tensorflow), and 124 [sec] for JT-VAE (pytorch).

These computational time may depend on choices of frameworks and skills of implementations. However we think it is safe to say that the GraphNVP is significantly faster than other models in sampling for several reasons: the GraphNVP decoder does not involve additional chemical validity check (Jin et al., 2018), or grammatical validity-assurance for sampling (Kusner et al., 2017). Deterministic decoding of graphNVP further reduces generation time in practical scenarios since a latent vector is not needed to be decoded multiple times as done for JT-VAE.

4.3 Smoothness of the Learned Latent Space

Next, we qualitatively examine the learned latent space z by visualizing the latent points space. In this experiment, we randomly select a molecule from the training set and encode it into a latent vector z_0 using our proposed model. Then we choose two random axes which are orthogonal to each other. We decode latent points lying on a 2-dimensional grid spanned by those two axes and with z_0 as the origin. Figure 4 shows that the latent spaces learned from both QM9 (panel (a)) and ZINC dataset (panel (b)) vary smoothly such that neighboring latent points correspond to molecules with minor variations. This visualization indicates the smoothness of the learned latent space, similar to the results of existing VAE-based models (e.g., (Liu et al., 2018; Ma et al., 2018)). However, it should be noted that we decode each latent point only once unlike VAE-based models. For example, GVAE (Kusner et al., 2017) decodes each latent point 1000 times and selects the most common molecule as the representative molecule for that point. Because our decoding step is deterministic such a time-consuming measure is not needed. In practice, smoothness of the latent space is crucial for *decorating* a molecule: generating a slightly-modified graph by perturbing the latent representation of the source molecular graph.

4.4 Property-Targeted Molecule Optimization

Our last task is to find molecules similar to a given molecule, but possessing a better chemical property. This task is known as *molecular optimization* in the field of chemo-informatics. We train a linear regressor on the latent space of molecules with quantitative estimate of drug-likeness (QED) of each molecule as the target chemical property. QED score quantifies how likely a molecule is to be a potential drug. We interpolate the latent vector of a randomly selected molecule along the direction of increasing QED score as learned by linear regression. Figure 5 demonstrates the learned latent space and a simple linear regression yields successful molecular optimization. Here, we select a molecule with a low QED score and visualize its neighborhood. However, we note that the number of valid molecules that can be generated along a given direction varies depending on the query molecule. We show another property optimization example on QM9 dataset in the appendix.

Although we could perform molecular optimization with linear regression, we believe an extensive Bayesian optimization (e.g., (Jin et al., 2018; Kusner et al., 2017)) on the latent space may provide better results.

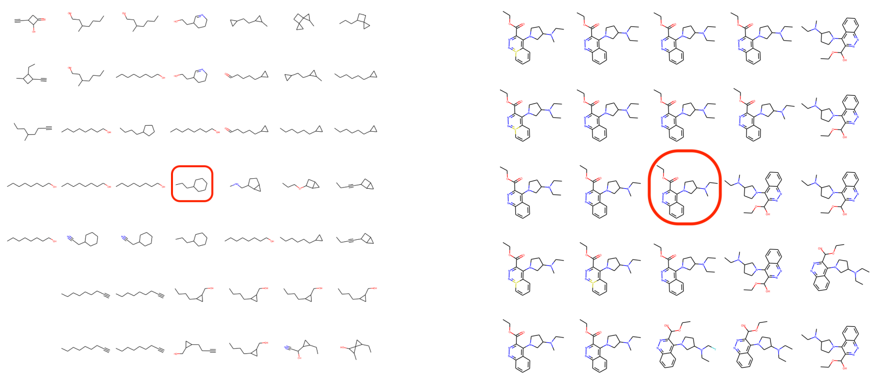

Figure 4: Visualization of the learned latent spaces along two randomly selected orthogonal axes. The red circled molecules are centers of the visualizations (not the origin of the latent spaces). An empty space in the grid indicates that an invalid molecule is generated. Left: Learned latent space for QM9. Right: Learned latent space for ZINC.

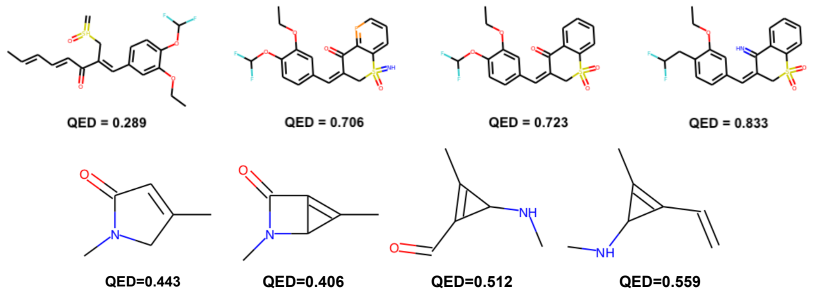

Figure 5: Chemical property optimization. Given the left-most molecule, we interpolate its latent vector along the direction which maximizes its QED property. Upper: Molecule optimization for ZINC. Lower: Molecule optimization for QM9.

5 Conclusion

In this paper, we proposed GraphNVP, an invertible flow-based model for generating molecular graphs. Specifically, the proposed model is the first invertible model for the whole graph components: both of node attributes and an adjacency tensor are converted into latent vectors through two novel invertible flows. Our model can generate valid molecules with a high uniqueness score and guaranteed reconstruction ability with very simple invertible coupling flow layers. The proposed model enjoys a fast graph generation; faster in order of magnitude than other graph generation models in our implementation. In addition, we demonstrate that the learned latent space can be used to search for molecules similar to a given molecule, which maximize a desired chemical property. There is an important open problem: how to improve the permutation-invariance of the proposed model. Additionally, we believe more exploration of the reasons contributing to the high uniqueness ratio of the proposed model will contribute to the understanding of graph generation models in general.

References

Nicola De Cao and Thomas Kipf. Molgan: An implicit generative model for small molecular graphs. arXiv preprint arXiv:1805.11973, 2018.

Laurent Dinh, David Krueger, and Yoshua Bengio. Nice: Non-linear independent components estimation. In Proceedings of the International Conference on Learning Representations (ICLR), 2015.

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. In Proceedings of International Conference on Learning Representations (ICLR), 2017. URL `https://arxiv.org/abs/1605.08803`.

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

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Advances in neural information processing systems, pp. 2672–2680, 2014.

Will Grathwohl, Ricky T. Q. Chen, Jesse Bettencourt, Ilya Sutskever, and David Duvenaud. FFJORD: Free-Form Continuous Dynamics for Scalable Reversible Generative Models. In Proceedings of ICLR, 2019.

Gabriel Guimaraes, Carlos Sanchez-Lengeling, Outeiral, Pedro Luis Cunha Farias, and Alan Aupuru-Guzip. Object-reinforced generative adversarial networks (organ) for seuqnce generation models. arXiv, 18:1705.18043v2 [stat.ml], 2017.

Emiel Hoogeboom, Jorn W.T. Peters, Rianne van den Berg, and Max Welling. Integer discrete flows and lossless compression. arXiv preprint arXiv:1905.07376, 2019.

John J Irwin, Teague Sterling, Michael M Mysinger, Erin S Bolstad, and Ryan G Coleman. Zinc: a free tool to discover chemistry for biology. Journal of chemical information and modeling, 52(7): 1757–1768, 2012.

Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Junction tree variational autoencoder for molecular graph generation. In Jennifer Dy and Andreas Krause (eds.), Proceedings of the 35th International Conference on Machine Learning, volume 80 of Proceedings of Machine Learning Research, pp. 2323–2332, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR. URL `http://proceedings.mlr.press/v80/jin18a.html`.

Diederik P. Kingma and Jimmy Lei Ba. Adam: a Method for Stochastic Optimization. In Proceedings of the International Conference on Learning Representations (ICLR), 2015.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In Proceedings of the 2nd International Conference on Learning Representations (ICLR), 2014. URL `https://arxiv.org/abs/1312.6114`.

Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In Advances in Neural Information Processing Systems, pp. 10236–10245. Curran Associates, Inc., 2018. URL `https://papers.nips.cc/paper/8224-glow-generative-flow-with-invertible-1x1-convolutions`.

Ivan Kobyzev, Simon Prince, and Marcus A Brubaker. Normalizing Flows: Introduction and Ideas. arXIv, pp. 1908.09257 [cs.LG], 2019.

Aviral Kumar, Jimmy Ba, Jamie Kiros, and Kevin Swersky. GRevnet: Improving Graph Neural Nets wiht Reversible Computation. In Proceedings of the Relational Representation Learning Workshop at NeurIPS 2018, 2018.

Matt J Kusner, Brooks Paige, and José Miguel Hernández-Lobato. Grammar variational autoencoder. In Proceedings of the 34th International Conference on Machine Learning-Volume 70, pp. 1945–1954. PMLR, 2017.

Jenny Liu, Aviral Kumar, Jimmy Ba, Jamle Kiros, and Kevin Swersky. Graph normalizing flows. arXiv preprint arXiv:1905.13177, 2019.

Qi Liu, Miltiadis Allamanis, Marc Brockschmidt, and Alexander Gaunt. Constrained graph variational autoencoders for molecule design. In Advances in Neural Information Processing Systems, pp. 7806–7815, 2018.

Tengfei Ma, Jie Chen, and Cao Xiao. Constrained generation of semantically valid graphs via regularizing variational autoencoders. In Advances in Neural Information Processing Systems, pp. 7113–7124, 2018.

Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv preprint arXiv:1511.06434, 2015.

Raghunathan Ramakrishnan, Pavlo O Dral, Matthias Rupp, and O Anatole Von Lilienfeld. Quantum chemistry structures and properties of 134 kilo molecules. Scientific data, 1:140022, 2014.

Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In Francis Bach and David Blei (eds.), Proceedings of the 32nd International Conference on Machine Learning, volume 37 of Proceedings of Machine Learning Research, pp. 1530–1538, Lille, France, 07–09 Jul 2015. PMLR. URL http://proceedings.mlr.press/v37/rezende15.html.

Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In European Semantic Web Conference, pp. 593–607. Springer, 2018.

Martin Simonovsky and Nikos Komodakis. Graphvae: Towards generation of small graphs using variational autoencoders. In International Conference on Artificial Neural Networks, pp. 412–422. Springer, 2018.

E G Tabak and Cristina V Turner. A Family of Nonparametric Density Estimation Algorithms. Communications on Pure and Applied Mathematics, 66(2):145–164, 2013.

Esteban G. Tabak and Eric Vanden-Eijnden. Density estimation by dual ascent of the log-likelihood. Communications in Mathematical Sciences, 8(1):217–233, 3 2010. URL https://projecteuclid.org:443/euclid.cms/1266935020.

L Theis, A van den Oord, and M Bethge. A note on the evaluation of generative models. In International Conference on Learning Representations (ICLR 2016), pp. 1–10, 2016.

Seiya Tokui, Kenta Oono, Shohei Hido, and Justin Clayton. Chainer: a Next-Generation Open Source Framework for Deep Learning. In Proceedings of Workshop on Machine Learning Systems (LearningSys) in The Twenty-ninth Annual Conference on Neural Information Processing Systems (NIPS), 2015. URL https://chainer.org/.

Dustin Tran, Keyon Vafa, Kumar Krishna Agrawal, Laurent Dinh, and Ben Poole. Discrete flows: Invertible generative models of discrete data. arXiv preprint arXiv:1905.10347, 2019.

Jiaxuan You, Bowen Liu, Zhitao Ying, Vijay Pande, and Jure Leskovec. Graph convolutional policy network for goal-directed molecular graph generation. In Advances in Neural Information Processing Systems, pp. 6412–6422, 2018a.

Jiaxuan You, Rex Ying, Xiang Ren, William Hamilton, and Jure Leskovec. GraphRNN: Generating realistic graphs with deep auto-regressive models. In Jennifer Dy and Andreas Krause (eds.), Proceedings of the 35th International Conference on Machine Learning, volume 80 of Proceedings of Machine Learning Research, pp. 5708–5717, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018b. PMLR. URL http://proceedings.mlr.press/v80/you18a.html.

A NETWORK ARCHITECTURE DETAILS

For QM9 dataset, we use a total of 27 adjacency coupling and 36 node feature coupling layers. For ZINC dataset, we keep the number of coupling layers equal to the maximum number of atoms a ZINC molecule can have, 38. We model affine transformation (both scale and translation) of an adjacency coupling layer with a multi-layer perceptron (MLP). As mentioned in the main text, we utilize both node assignments and adjacency information in defining node feature coupling layers. However, we found affine transformations can become unstable when used to update the feature matrix with Relational-GCN (RelGCN). Therefore, we use only additive transformations in node feature coupling layers.

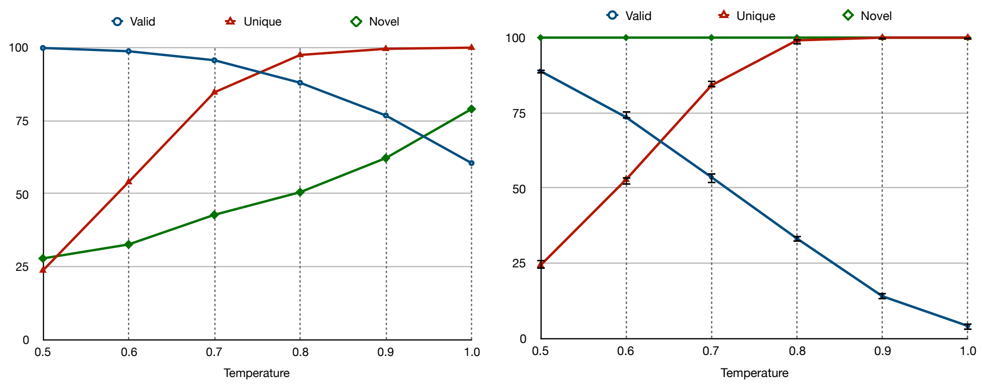

Figure 6: Impact of temperature on the quality of graph generation. Sampling with a smaller temperature yields more valid molecules but with less diversity (uniqueness) among them. Each experiment is performed five times and the average is reported in this figure. Left panel: impact of temperature on sampling from latent space of QM9. Right panel:Impact of temperature on sampling from latent space of ZINC.

We initialize the last layer of each RelGCN and MLP with zeros, such that each affine transformation initially performs an identity function.

We train the models using Adam optimizer with default parameters ($\alpha = 0.001$) and minibatch sizes 256 and 128 for QM9 and ZINC datasets. We use batch normalization in both types of coupling layers.

B Training Details

For training data splits, we used the same train/test dataset splits used in (Kusner et al., 2017). We train each model for 200 epochs. We did not employ early-stopping in the experiments. We chose the model snapshot of the last (200) epoch for evaluations and demonstrations. All models are implemented using Chainer-Chemistry[1] and RDKit[2] libraries.

C Effect of Temperature

Following previous work on likelihood-based generative models (Kingma & Dhariwal, 2018), we sampled latent vectors from a temperature-truncated normal distribution. Temperature parameter handles uniqueness and validity trade off. Sampling with a lower temperature results in higher number of valid molecules at the cost of uniqueness among them. How temperature effects validity, uniqueness, and novelty of generated molecules is shown in Figure 6. Users may tune this parameter depending on the application and its goal. In our experiments we chose 0.85 and 0.75 as the temperature values for QM9 and ZINC models respectively.

D Effect of Adjacency Tensor in GraphNVP Coupling

We performed additional experiment to quantify the effect of A introduced in the node feature coupling. We trained an ablation model, which replace the RelGCN layer with an MLP which does not use A. For QM9 dataset the validity drops to $41.8 \pm 1.26\%$, about half the validity of original GraphNVP model.

[1]https://github.com/pfnet-research/chainer-chemistry

[2]https://github.com/rdkit/rdkit

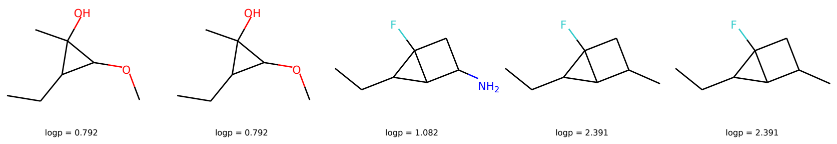

Figure 7: Chemical property optimization. We select a molecule from QM9 dataset randomly and then interpolate its latent vector along the axis which maximizes water-octanol partition coefficient (logP).

E Additional Visualizations

Fig. 7 illustrates an example of chemical property optimization for water-octanol partition coefficient (logP) on QM9 dataset.