Semi-Supervised Junction Tree Variational Autoencoder for Molecular Graphs

by

Atia Hamidi Zadeh

B.Sc., Sharif University of Technology, 2019

Thesis Submitted in Partial Fulfillment o f the
Requirements for the Degree of
Master of Science

in the
School of Computing Science
Faculty of Applied Sciences

SIMON FRASER UNIVERSITY
Spring 2023

Declaration of Committee

Name:	**Atia Hamidi Zadeh**
Degree:	**Master of Science**
Thesis title:	**Semi-Supervised Junction Tree Variational Autoencoder for Molecular Graphs**
Committee:	**Chair:** Ali Mahdavi-Amiri Assistant Professor, Computing Science
	Martin Ester Supervisor Professor, Computing Science
	Manolis Savva Committee Member Assistant Professor, Computing Science
	Ke Li Examiner Assistant Professor, Computing Science

Abstract

Molecular Representation Learning is essential to solving many drug discovery and computational chemistry problems. It is a challenging problem due to the complex structure of molecules and the vast chemical space. Graph representations of molecules are more expressive than traditional representations, such as molecular fingerprints. T herefore, t hey can improve the performance of machine learning models. We propose SeMole, a method that augments the Junction Tree Variational Autoencoders, a state-of-the-art generative model for molecular graphs, with semi-supervised learning. SeMole aims to improve the accuracy of molecular property prediction when having limited labeled data by exploiting unlabeled data. We enforce that the model generates molecular graphs conditioned on target properties by incorporating the property into the latent representation. We propose an additional pre-training phase to improve the training process for our semi-supervised generative model. We perform an experimental evaluation on the ZINC dataset using three different molecule properties and demonstrate the benefits o f s emi-supervision. W e p erform f urther experiments on the QM9 dataset including twelve molecule properties.

Keywords: Semi-Supervised Learning; Molecular Property Prediction; Generative Models; Molecule Generation

Dedication

This thesis is dedicated to my parents and my brother for their endless love, support, and encouragement.

Acknowledgements

In this section, I want to acknowledge my supervisor, Prof. Martin Ester for constantly guiding me through this project patiently, and continuously caring for my personal and professional growth during my master's. I would also like to thank my teammate in this project, Tony Shen. I could not finish this thesis without his support and hard work. I would also like to thank Terramera for giving me the chance to form my thesis idea during my internship in their interdisciplinary environment. I would like to thank Dr. Sadegh Shokatian and Dr. Oliver Snow for everything I have learned from them during this internship. Lastly, I want to express my gratitude to my labmates in Ester lab for their friendly support and helpful feedback on my project.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

One of the challenges in the drug development pipeline is to discover small molecules with desired properties. Testing candidate molecules experimentally in the wet lab is time-consuming and expensive. The main challenge for computational methods is the vast chemical space and the challenging nature of navigating through this space where molecule representation learning attracts attention. Molecule representation learning has been utilized by either end-to-end training or pretrained strategies to solve drug discovery problems such as generating molecules [19, 7, 33], and molecule property prediction [11, 63, 17]. Recently, advancements in learning molecule representations have been made to propose novel molecules with targeted desired properties [12, 21, 42, 43]

Several molecular representation learning models have been designed by representing molecules as graphs, where nodes are atoms and bonds are edges. Utilizing Graph Neural Networks(GNNs) [59] have resulted in promising performances for unconditionally generating molecules [19, 37, 64, 49, 65]. Junction Tree Variational Autoencoders(JTVAE) [19] is one of a state-of-the-art graph-based methods for generating molecules. This method converts the graph structure to the associated tree structure of molecules by breaking them into components predefined in the vocabulary of chemical bonds. This ensures the model generates molecules by assembling chemically valid blocks that result in generating 100% valid molecules.

Previous generative methods do not generate molecules conditioned on target properties, but optimize the latent space based on a target property. [19],[33], and [7] learn the latent representation in an unsupervised manner, minimizing the reconstruction error, and optimize molecules with respect to the desired property afterward, following the optimization approach proposed by [12]. Despite these methods' promising performance, the optimization approaches do not allow setting the property to a specific value and depend on the definition of the objective function for the optimization task. Therefore They are also not scalable for generating molecules with multiple desired properties.

Moreover, these supervised training methods assume that there is enough labeled data to train a classifier or regressor. These solutions largely depend on the availability of labeled

datasets, which is not the case in many real-world datasets. Recently, researchers have proposed methods for semi-supervised graph classification [54, 16] to solve the data scarcity problem. Therefore semi-supervised learning could be beneficial in enhancing the power of molecule generative models.

Kang et al. [21] proposed a semi-supervised variational autoencoder (SSVAE) that conditionally generates molecules without any post hoc optimization. However, SSVAE represents molecules as SMILES strings [58], which often leads the model into generating invalid molecules. Since a single variation in text-based representation might change the molecule structure significantly and even invalidate the molecule. Also, graph representation of molecules can be broken down into meaningful subgraphs, such as bonds and rings, while substrings do not necessarily represent a meaningful division of the molecule structure, which prevents us from validity checks during the generation process [64]. The SMILES representation of molecules is also not sensitive to molecule similarities, making it hard to learn a smooth embedding of molecules [19]. Moreover, semi-supervised learning with generative models [24] is challenging to train end-to-end [39].

In this work, we extend a state-of-the-art generative model, JTVAE, for molecular graphs with semi-supervised learning to learn molecular properties directly as part of the latent representations via partial supervision. We propose an additional pre-training phase to improve the training process for the semi-supervised generative model. While JTVAE achieved state-of-the-art performance on generating molecules unconditionally, generating molecules conditioned on target properties with limited labeled data remains largely untapped. In conclusion, this paper makes the following contributions:

- We combine a semi-supervised model with a state-of-the-art molecular graph generative model, JTVAE, for molecule property prediction and generation of valid molecules with desired properties.
- We improve the training process of the Semi-supervised Variational Autoencoder by adding a pre-training phase.
- We performed an experimental evaluation on ZINC dataset for three different molecule properties on molecule property prediction and generation with respect to target properties.
- We also performed experimental evaluations on the QM9 dataset for twelve different molecule properties on semi-supervised molecule property prediction.

The rest of this thesis is organized as follows: Chapter 2 represents the related work. Chapter 3 discusses the methodology of our proposed method. Chapter 4 includes experiments, datasets, and results. The last section will conclude the thesis and explains the limitations and future work.

I want to acknowledge that a short version of this thesis was accepted by Deep Learning on Graph Workshop at AAAI 2023, and it was done in collaboration with Tony Shen, an undergraduate student at SFU.

Chapter 2

Related Work

2.1 Molecules as graphs

Molecule structures can be represented by 2D graphs intuitively, where nodes represent atoms and edges represent bonds. The 2D molecular graphs effectively capture pairwise interactions between atoms in a molecule. Both nodes and edges in molecular graphs are explained by features/attributes. Edge attributes mainly consist of bond types, being conjugated, or existing in a ring where node attributes include atomic mass, atom type, chirality, aromaticity, etc. [63]. The 2D graph representation of molecules has several advantages over other ways of representing molecule structures. First molecular graph representations are reversible, meaning converting a molecular graph back to a molecular structure is trivial. At the same time, this does not hold for molecular fingerprints such as Morgan (ECFP) fingerprints [45] or physical descriptors [40]. For instance, a generated fingerprint cannot be mapped to a molecular structure. Second string-based representation of molecules, SMILES, cannot capture molecule similarities, meaning similar molecule structures could be represented in different SMILES strings [19]. Finally, graph representation of molecules can be broken down to meaningful subgraphs such as bonds and rings, while substrings do not necessarily represent a meaningful division of the molecule structure.

However, the 2D representation of molecular graphs only contains limited information about the spatial structure of molecules as part of the node and edge attributes, which prevents them from recognizing spatial isomerism and conformation preferences. Moreover, there are chemical compounds that cannot be described by the graph notation, such as molecules with ionic bonds or compounds where bonds are unstable and are formed and deformed. Another disadvantage of graph representations is their scalability. As the molecules become larger, it becomes more challenging the allocate the required memory for representing their graph structure of them. Therefore computational tasks such as generating graphs are also becoming more challenging.

3D representation of molecules could gain more meaningful information about the molecule structure. Molecules can be represented in 3D format by incorporating cartesian coordinates

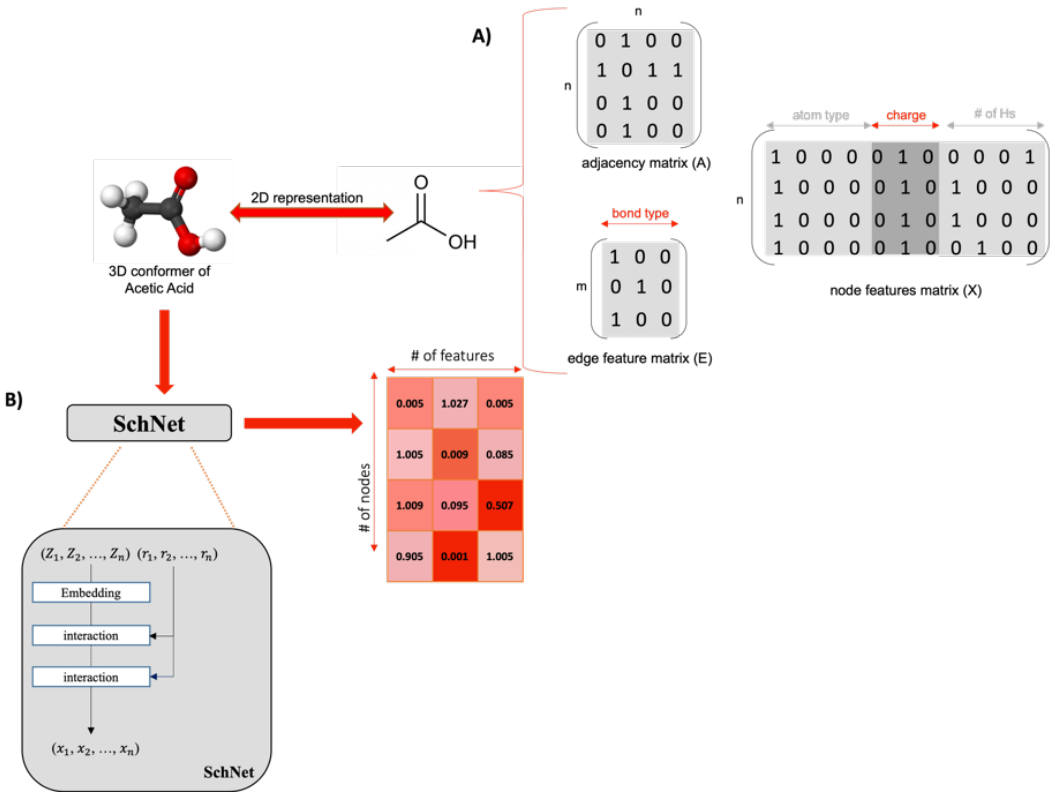


Figure 2.1: Representation of 2D topology and 3D structures drug molecules as graphs. (A) The molecular topology is defined by constituent elements and corresponding chemical bonds. Nodes and edges are attributed by atomic and bond features in the node feature matrix and edge feature matrix. (B) SchNet[48] provides molecule representation for atomic systems that can be described with atomic types and positions.

for atoms in space, i.e., the atomistic geometry. Each molecule is represented by a set of atoms, including their atomic types $Z = (Z_1, \ldots, Z_n)$ and positions $R = (\mathbf{r}_1, \ldots, \mathbf{r}_n)$. [48] [55]. These cartesian coordinates are not static since atoms could move in reality. Therefore, different positions are possible for the atoms in a molecule, leading to different potential energy levels. [1]. Molecular structures with a local minimum of potential energy are called *conformers*. [36]

Many methods applied on 3D molecules use SchNet [48] as a step to extract molecule embedding [51, 62, 1, 36]. SchNet provides molecule representation for atomic systems that can be described with atomic types and positions. The main blocks in SchNet architecture are *embedding block* and *interaction block*. The embedding block is responsible is a linear process applied on the atom type that maps the input to features, while the interaction block is used to learn pairwise interaction between each atom and its neighbors. The interaction block consists of dense layers along with continuous filter convolutional layers that generalize convolutional layers in image applications since, unlike image pixels, atoms are located in

continuous positions. The combination of these two components allows the interaction block to model the pairwise interaction.

2.2 Graph Representation Learning

Advancements in Deep Learning [13] have led to remarkable success in a wide range of real-world applications, from computer vision [56] and speech recognition [41] to genomics [10] and drug discovery [5]. There are increasing applications among deep learning tasks where data is represented as graphs such as Social Network [57], Protein Protein Interaction [32], and Knowledge Graphs [18]. Extending deep learning approaches to graph data is advantageous but challenging due to graphs' complex nature and interdependent structure. A few of these challenges are the variable number of nodes and their unordered structure.

Computational tasks performed on graphs can be divided into two main categories: node level tasks and graph level tasks. The former includes tasks where data points are nodes, such as Node Classification and Link Prediction. The latter contains tasks where each data point is a graph, such as Graph Classification and Generation.

Before further discussion, it is important to define our meaning of graphs formally. In this section, whenever the word *graph* is used, the purpose is an undirected unweighted graph. Following the notation of [60], graphs are represented as $G = (V, E)$ where V is the set of nodes and $n = |V|$ is the number of nodes. E is defined as the set of edges between nodes. Let A denote the adjacency matrix of G such that A_{ij} is 1 if there is an edge between node i and node j, and it will be 0 otherwise. Each node may have features that are represented by the matrix X. Edges could also have a feature matrix X^e.

Graph Neural Networks(GNNs) [46] are a category of neural networks focusing on applying deep learning to a wide range of applications where data is represented as graphs. GNNs are designed to learn better node features in node-level or graph-level tasks, while an aggregation step known as *graph pooling* is added in graph-level tasks to reach a graph-level representation from multiple node-level representations. The process of learning better node features given the graph structure, and the current node features have been called *graph filtering* [38] and can be summarized as follows:

$$\mathbf{F}' = h\left(\mathbf{A}, \mathbf{Fc}\right) \tag{2.1}$$

where F and F' denote the current and the output node feature matrix, respectively, and h represents the operator for graph filtering.

There is a wide range of biochemistry problems that deep learning approaches have addressed since many tasks in this application are time-consuming and expensive to experiment with in the wet lab. Molecules, proteins, drug-protein interactions, and protein-protein networks are examples of naturally representable data as graphs. For instance, a chemical

compound can be represented as a graph such that nodes indicate atoms and chemical bonds indicate edges.

2.3 Molecule Representation Learning

Molecule representation learning methods are the backbone of most learning approaches for drug discovery. They enable learning informative representation of molecular graphs for downstream prediction tasks such as molecule property prediction. The main challenges of this task are the variable size of molecular graphs and the generalization of the prediction model. Message Passing Neural Networks (MPNNs)[11] is a general framework for graph classification tasks on molecular graphs. MPNNs consists of two steps. The message passing phase iteratively performs message functions M_t and update functions U_t on node features at each step. The readout phase aggregates updated node embeddings to gain molecular graph representation by the readout function R. The node embeddings are updated as follows:

$$m_v^{t+1} = \sum_{w \in N(v)} M_t \left(h_v^t, h_w^t, e_{vw} \right) \tag{2.2}$$

$$h_v^{t+1} = U_t \left(h_v^t, m_v^{t+1} \right) \tag{2.3}$$

Where h_v^t denotes the node embedding for v at step t and $N(v)$ represents the neirborhood of v. The readout function is invariant to permutation and operates as follows:

$$\hat{y} = R \left(\left\{ h_v^T \mid v \in G \right\} \right) \tag{2.4}$$

Duvenaud at el. [9], Kearnes et al. [22], and Kipf Welling [28] are a few models that fit into this framework. Directed Message Passing Networks (D-MPNNs) [63] extend MPNNs by using messages associated with bonds instead of messages associated with nodes, preventing the message passing process from being stuck in unnecessary loops.

Molecular property prediction models mentioned above are based on supervised training, assuming enough labeled data to train a classifier or regressor. These solutions largely depend on the availability of labeled datasets, which is not the case in many real-world datasets [54, 16]. have been proposed as semi-supervised approaches to graph classification to solve the data scarcity challenge. ASGN [16] addressed the data scarcity problem using a semi-supervised method that predicts the most informative samples obtained by an active learning approach. The active learning model enables the domain expert to effectively help the model by labeling the most representative samples to add to the training data. InfoGraph [54] is a semi-supervised graph-level representation learning method that has been evaluated on molecular property prediction. This method employs a student-teacher framework where the teacher model is trained on unlabeled data, and the student model is trained on the labeled data using a supervised objective. InfoGraph maximizes the mutual information

between the representations learned by these two models so that the student model learns from the teacher model. Experiments show that the semi-supervised method performs better than the supervised version of the model.

Pre-trained models improve the generalization powers of machine learning methods. In most cases, the distribution of the training and testing data are different, which makes the prediction tasks challenging, particularly. At the same time, datasets are time-consuming and expensive to annotate, and labels for each task are scarce. Pre-training models leverage the huge unlabeled chemical space to improve out-of-distribution generalization. Hu et al. [17] proposed a framework to compare the representation learning capabilities of different GNNs. Their strategy for pre-training GNNs is designed at the node and graph levels. For node-level pre-training, two self-supervised tasks are performed: attribute masking, which refers to masking the attributes for nodes or edges and encouraging GNN to predict the ground-truth attribute based on the neighborhood. Context prediction refers to taking a subgraph and predicting the graph structure in the neighborhood of that subgraph. For graph-level pre-training, a set of labels are jointly predicted for each graph. Performing the graph-level pretraining is not solely efficient since not all the supervised tasks in pre-training are related to the downstream task, which can result in a negative transfer. The node level pre-training enables the model to learn domain-specific knowledge into node embeddings that later form the graph level representations. This strategy significantly improved out-of-distribution generalization for molecule property prediction and protein-protein interaction.

2.4 Molecular Graph Generation

One of the most significant challenges in drug discovery is discovering novel, valid chemical compounds with desired properties. This process is time-consuming and expensive, which led to proposed computational algorithms to tackle this challenge. Graph neural networks have been utilized for generating molecular graphs. Generating graphs is challenging due to node-node dependencies and the variable number of nodes and edges. Generating molecular graphs is particularly challenging since the vast and discrete chemical space while minor changes to the generated molecular graph lead to molecules with entirely different properties [30]. To tackle these challenges, many approaches have been proposed. Auto-regressive approaches are the idea behind many state-of-the-art graph generation methods. In this approach, molecules are generated atom by atom. There are domain-specific constraints incorporated in generating the molecule, which helps to prevent the model from generating invalid molecules. These models leverage three main categories of generative models in the deep learning literature: Variational Autoencoders, Generative Adversarial Networks, and Normalizing Flows.

Variational Autoencoders (VAEs)[25] map the input to a continuous latent variable that is limited to a probability distribution like Gaussian and reconstructs the input by

sampling from that probability distribution using the learned parameters in the latent space. CGVAE [35], [37] use the VAE framework along with GNN-based encoders and decoders to generate molecular graphs. Following autoregressive approaches using VAE, Jin et al. [19] proposed Junction Tree Variational Auto-encoders (JTVAE) that convert molecular graphs to associated tree structures of molecules by breaking them into components predefined in the vocabulary of chemical bonds. Unlike previous methods, the molecule is generated by putting these chemical blocks together instead of generating molecules atom by atom, which helps generate valid molecular structures.

Generative Adversarial Networks (GANs) [14] consists of two networks. The generator network learns a mapping from a Gaussian noise to the data distribution. The discriminator networks distinguish samples generated by the generator (fake samples) from ground truth data (real samples). GANs have been widely used for generating molecular graphs. [64] uses GANs along with a Reinforcement Learning(RL) agent that optimizes the generated molecules by incorporating positive and negative rewards for molecular properties or violating chemical criteria, respectively.

Normalizing Flows (NFs) [31] generate molecules from a prior distribution like guassian. Data is generated using a series of invertible transformations that map the prior distribution to the data distribution. The benefits of Normalizing Flows over VAEs in the image applications lead to applying them to molecular graphs [49]. GraphAF iteratively generates nodes and edges based on the existing sub-graphs and incorporates domain-specific rules such as valency check in the process, which leads to generating 100% valid molecules. The main advantage of flow-based methods such as GraphAF over VAE-based methods is their ability to find the exact data likelihood insead of an approximation, making the training process more efficient. Another category besides auto-regressive models are approaches that generate adjacency matrix, node, and edge features of the molecular graph [52] [8]. Generative models for graphs aim to learn the distribution of the graph-structured data in the training set and generate new objects similar to them.
The task of generating molecules attempts to generate molecules with desired properties that preferreably are valid and synthesizable.
Methods of graph generation come from three categories of deep generative models:
(1) Variational autoencoders (VAEs) (2) Generative Adversarial Networks (GANs) (3) Normalizing Flows (NF). We will not discuss methods that exist out of these three categories. However, Table 2.1 demonstrates some of these methods.

2.4.1 Variational Autoencoders for Graph Generation

Variational Autoencoders (VAEs) [25] has shown considerable performance in solving the graph/molecule generation problem. Variational Graph Autoencoders (VGAE) [29] is an unsupervised learning method based on VAE to learn graph-structured data that incorporates node features as a given variable to the inference model in addition to the adjacency

VAEs	GANs	NF	Others
JTNN [19]	GCPN [64]	GraphAF [49]	GraphRNN [65]
VJTNN [20]	NetGAN [3]		MoleculeChef [4]
SemVAE [37]	MolGAN [8]		
CVAE [35]			

Table 2.1: Categorization of the molecule generation methods

matrix.

Inspired by VAE, VGAE consists of two components. The infenece model and the generative model. The inference model is parameterized by a two-layer Graph Convolution network(GCN) [28]. In Equation 2.5 $\mu = GCN_{\mu}(X, A)$ and $\log \sigma = GCN_{\sigma}(X, A)$.

$$q(\mathbf{Z} \mid \mathbf{X}, \mathbf{A}) = \prod_{i=1}^{N} q\left(\mathbf{z}_i \mid \mathbf{X}, \mathbf{A}\right), \text{ with } q\left(\mathbf{z}_i \mid \mathbf{X}, \mathbf{A}\right) = \mathcal{N}\left(\mathbf{z}_i \mid \boldsymbol{\mu}_i, \operatorname{diag}\left(\boldsymbol{\sigma}_i^2\right)\right) \tag{2.5}$$

The propagation formula of each GCN layer is defined as:

$$\operatorname{GCN}(\mathbf{X}, \mathbf{A}) = \tilde{\mathbf{A}} \operatorname{ReLU}\left(\tilde{\mathbf{A}} \mathbf{X} \mathbf{W}_0\right) \mathbf{W}_1 \tag{2.6}$$

and $\tilde{\mathbf{A}} = \mathbf{D}^{-\frac{1}{2}} \mathbf{A} \mathbf{D}^{-\frac{1}{2}}$ is the normalized adjacency matrix where D is the degree matrix of the nodes.

The generated adjacency matrix is given by an inner product between latent variables.

$$p(\mathbf{A} \mid \mathbf{Z}) = \prod_{i=1}^{N} \prod_{j=1}^{N} p\left(A_{ij} \mid \mathbf{z}_i, \mathbf{z}_j\right), \text{ with } p\left(A_{ij} = 1 \mid \mathbf{z}_i, \mathbf{z}_j\right) = \sigma\left(\mathbf{z}_i^{\top} \mathbf{z}_j\right) \tag{2.7}$$

where $\sigma(.)$ is the logistic sigmoid function.

VGAE levreges this encoder-decoder architecture to maximize the variational lower bound in 2.5. A Gaussian prior has been taken. Full-batch gradient descent and reparameterization trick have been applied during training.

$$\mathcal{L} = \mathbb{E}_{q(\mathbf{Z}|\mathbf{X},\mathbf{A})}[\log p(\mathbf{A} \mid \mathbf{Z})] - \mathrm{KL}[q(\mathbf{Z} \mid \mathbf{X}, \mathbf{A}) \| p(\mathbf{Z})] \tag{2.8}$$

VGAE [29] has performed reasanably well on learning a meaningful embedding for link prediction tasks. This model has been also tested on link prediction tasks without any node features. To sum up, this model has been used as a representation learning model more than a generative model, although it reconstucts the graph's adjacency matrix. One of the challenges of these models is the simplifying assumption for the prior. Also, the model only reconstructs the adjacency matrix.

Junction tree variational autoencoder [19] is one of the state-of-the-art methods for molecule generation based on VAEs. In this approach, molecules have been decomposed into a tree-structured object. Molecules are decomposed into trees by changing certain vertices into a

node in a way that the graph becomes cycle-free. In the junction trees of molecules, nodes are a vocabulary of valid chemical substructures or components that are extracted from training data. This tree structure demonstrates the subgraph components of the molecule. Then these subgraphs have been combined together to form a complete molecule
The encoder consists of learning the encodings for both molecular graph and junction tree. The tree encoding is designed to represent the subgraphs from the vocabulary of chemical bonds, rings, and individual atoms. Then the encoding of the molecule captures how these subgraphs should be connected to each other.
The graph is encoded by a message passing network [11] and message passing is performed in a loopy belief propagation fashion because molecule graphs have cycles. In Equation 2.9 $\nu_{uv}^{(t)}$ is a hidden vector denoting the message from vertex u to v in the t-th iteration of the message passing. Here, $\mathbf{x}_u$ is the node feature vector consisting atoms' properties and $\mathbf{x}_{uv}$ is the edge feature vector indicating the bond type between two atoms. The initial amount for the hidden vectors are zero.

$$\nu_{uv}^{(t)} = \tau\left(\mathbf{W}_1^g \mathbf{x}_u + \mathbf{W}_2^g \mathbf{x}_{uv} + \mathbf{W}_3^g \sum_{w \in N(u)\setminus v} \nu_{wu}^{(t-1)}\right) \tag{2.9}$$

Messages are aggregated after T iterations of message passing as the latent vector for each node. Equation 2.10 indicates the aggregation fomula for each vertex. The mean μ_G and variance σ_G of the posterior approximation is calculate by a two layer network from h_G, e.i, the graph representation which is $\sum_i \mathbf{h}_i/|V|$. Finaly the latent representation of the graph is sampled from a Gaussian distribution parameterized by μ_G and σ_G.

$$\mathbf{h}_u = \tau\left(\mathbf{U}_1^g \mathbf{x}_u + \sum_{v \in N(u)} \mathbf{U}_2^g \boldsymbol{\nu}_{vu}^{(T)}\right) \tag{2.10}$$

Tree encoding is different from the graph encoding. Here, the encoding is inspired by belief propagation since the tree structure is cycle-free. An arbitrary node has been picked as the root. The message is being propagated from root to leaf nodes and another way around. Gated Recurrent Unit(GRU) [6] is used for updating messages in each iteration:

$$\mathbf{m}_{ij} = \mathrm{GRU}\left(\mathbf{x}_i, \{\mathbf{m}_{ki}\}_{k \in N(i)\setminus j}\right) \tag{2.11}$$

where m_{ij} is the message from i-th component to j-th component and x_i is a one-hot encoding indicating the label of the component. The latent representation of each node is calculated by Equation 2.12 aggregating all inward messages. Unlike graph encoding, here, the tree representation is the aggregated message for the root.

$$\mathbf{h}_i = \tau\left(\mathbf{W}^o \mathbf{x}_i + \sum_{k \in N(i)} \mathbf{U}^o \mathbf{m}_{ki}\right) \tag{2.12}$$

For the decoder, the junction tree has been decoded by generating one node at a time. For each node, the decoder first decides if this node has children or not. The decoder makes this decision by calculating this probability using z_T, node features and inward messages. Then each new generated child is being labeled with

$$\mathbf{q}_j = \text{softmax}\left(\mathbf{U}^l \tau\left(\mathbf{W}_1^l \mathbf{z}_\mathcal{T} + \mathbf{W}_2^l \mathbf{h}_{ij}\right)\right) \tag{2.13}$$

where q_j is a distribution over the chemical vocabulary.
The tree decoder minimizes the following cross entropy loss:

$$\mathcal{L}_c(\mathcal{T}) = \sum_t \mathcal{L}^d\left(p_t, \hat{p}_t\right) + \sum_j \mathcal{L}^l\left(\mathbf{q}_j, \hat{\mathbf{q}}_j\right) \tag{2.14}$$

given $\hat{p}_t$ and $\hat{q}_j$ as the true labels for the binary prediction of generating a new node and its label from the vocabulary.
Then the molecular graph is formed by a graph decoder from the junction tree. The reason for generating the tree structure is to make sure of the validity of the molecule by first identifying the components.

The graph decoder aims to assemble the tree cluster nodes together into the correct molecular graph. For each tree $\mathcal{T}$, there is a set of potential molecular graphs $\mathcal{G}(\mathcal{T})$. f^a is a scoring function over potential graphs and subgraphs. Let G_i be a potential subgraph assembled the nodes in the tree starting from node i. G_i is scored by first calculating $\mathbf{h}_{G_i}$, Then the scoring function is calculated using $f_i^a\left(G_i\right) = \mathbf{h}_{G_i} \cdot \mathbf{z}_G$. Note that a is used the indicate the position of the atoms in the junction tree.

The graph decoder is parameterized to maximize the log-likelihood of predicting correct subgraphs starting from the tree root.

$$\mathcal{L}_g(G) = \sum_i \left[f^a\left(G_i\right) - \log \sum_{G_i' \in \mathcal{G}_i} \exp\left(f^a\left(G_i'\right)\right) \right] \tag{2.15}$$

Figure 2.2 illustrates the method.

One of the challenges in molecule generation is to ensure that the generated graph is semantically valid. For example, atoms' valency is one of the constraints for a valid molecule that should not exceed specific amounts. Moreover, although there are methods defining the generative process as a sequence of decisions about whether one node or bond should be inserted or not, as the number of nodes growing, it is more challenging to model the generative process as a sequence of decisions conditioned on the constructed graph. Moreover in this case there is no predefined order for the sequences so the permutatoin problem remains unsolved.
Ma [37] proposed a method that addresses these problems by regularizing VAEs with validity constraints and focusing on matrix representation of the graph that generates molecules in

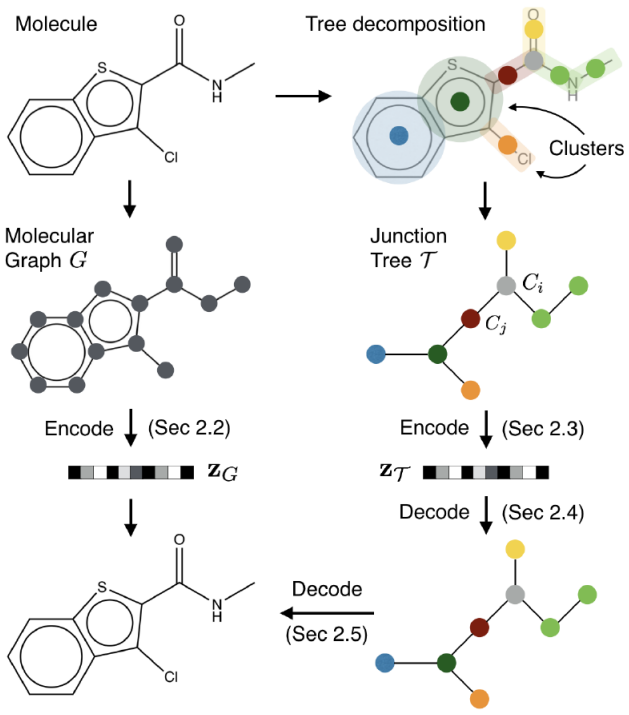


Figure 2.2: Overview of the Junction Tree Variational Autoencoder method [19]

a *one-shot* style.

Here, $\widetilde{G}$ is a random graph model and the probability of sampling a graph from $\widetilde{G}$ is defined as follows:

$$\prod_{i=1}^{N} \prod_{r=1}^{1+d} \widetilde{F}(i,r)^{F(i,r)} \prod_{i<j}^{1+t} \widetilde{E}(i,j,k)^{E(i,j,k)} \tag{2.16}$$

where $\widetilde{F}$ is the probabilty of nodes belonging to type r and $\widetilde{E}$ is the probability of edge (i,j) belonging to type k.

A standard VAE[25] model parameterizes the likelihood $p_\theta(G \mid z)$, so we are not bringing the equations again here. Adding the regualarization for constraints the objective function is defined as:

$$-L_{\text{ELBO}}(\theta,\phi) + \mu \sum_i \left[\int g_i(\theta,z)^2 p_\theta(z) dz \right]^{\frac{1}{2}} \tag{2.17}$$

where $g_i(\theta,z) \leq 0$ is defined as the vailidity constraint function.

2.4.2 Generative Adversarial networks for Graph Generation

Generative Adversarial networks (GANs) [14] are growing in different areas of machine learning. In this architecture, the model attempts to capture the distribution of the data using an adversarial process. This model has two components. A generative model that aims

to generate realistic fake samples and a discriminative model that seeks to identify real data from the generated one. These two components work together in the training phase so that the generative model minimizes its error and the discriminative model maximizes its accuracy to discriminate fake samples.

In the task of molecular graph generation, it is challenging to design molecules with desired properties such as drug-likeness while maintaining physical laws such as chemical valency. Since the chemical space is very large while this space is discrete and small changes in molecule structure can change the properties drastically. Moreover, the generator should be able to search the molecule space in whole because the distribution of the molecules with same properties are not necessary close. [64]

Graph Convolutional Policy Network (GCPN) [64] attempts to solve this problem using reinforcement learning and adversarial training. Reinforcement learning is capable of incorporating desired properties and constraints into the objective function by adding reward functions. Reinforcment learning also gives more power to the model in order to explore chemical space beyond the training data. Adversarial traning on the other hand is used to let the model to make use of learning from the example molecules [64].

GCPN learns an RL agent to generates molecules iteratively by predicting the action of adding a bond between two existing atoms or connecting a new atom or a substructure to an exsting partially generated molecule.

The generating process is formulated as a Markov Decission Process. In Figure 2.3 each row corresponds to an state. $S = s_i$ consists of all the states in the Markov process including all the intermediate and final graphs. The set of scafold subgraphs is also available to the model at each state in order to make the decision of adding a bond or not. It is assumed that G_0, e.i. the intermediate graph of the first state contatins a single carbon atom.

In the next step the node embeddings are calculated using message passing by GCN [28]. Next step is action prediction. The link predicton depends on four other action predictions. The choice of the first node that is extracted from the probability of selecting each node. This probability is calculated from the feature vector X by m_f which indicates a Multilayer Perception(MLP). The feature vector of the first selected node X_{afirst} is concatenated with each node in the feature vector X. m_s is another MLP to map the concatenated embedding to the probability distribution of choosing each node as the second node. m_e is the other MLP that produces the categorical edge type of the potential bond between the first and second node. As the last step of the action prediction the node embeddings are aggregated by AGG and the prbability of termination is caculated by m_t from the aggregated graph embedding.

$$f_{\text{first}}\ (s_t) = \text{SOFTMAX}\left(m_f(X)\right), \quad a_{\text{first}}\ \sim f_{\text{first}}\ (s_t) \in \{0,1\}^n \tag{2.18}$$

$$f_{\text{second}}\ (s_t) = \text{SOFTMAX}\left(m_s\left(X_{a_{\text{first}}}, X\right)\right), \quad a_{\text{second}}\ \sim f_{\text{second}}\ (s_t) \in \{0,1\}^{n+c} \tag{2.19}$$

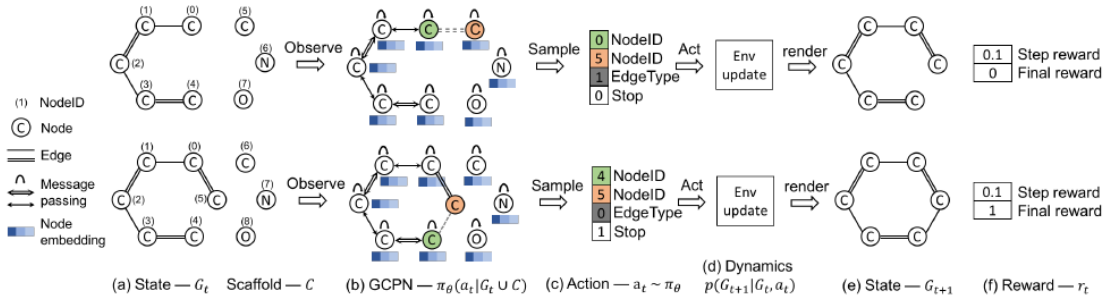


Figure 2.3: A overview of the GCPN method [64]

$$f_{\text{edge}}\ (s_t) = \text{SOFTMAX}\left(m_e\left(X_{a_{\text{first}}}, X_{a_{\text{second}}}\right)\right), \quad a_{\text{edge}} \sim f_{\text{edge}}\ (s_t) \in \{0,1\}^b \tag{2.20}$$

$$f_{\text{stop}}\ (s_t) = \text{SOFTMAX}\left(m_t(\ \text{AGG}\ (X))\right), \quad a_{\text{stop}} \sim f_{\text{stop}}\ (s_t) \in \{0,1\} \tag{2.21}$$

After identifying the actions, the environment incorporates chemical knowledge such as valency check to eliminate infeasible actions proposed by policy network. The model is able to do this step because of the graph representaion of the molecule and it cannot be done as a step in the generative process on a text-based representation.

The final and step rewards are designed to guide the RL agent. The final reward is the sum of domain-specific reward and adversarial loss. The domain-specific rewards consist of property scores such as molecular weight and druglikeness(QED). The domain-specific rewards also incorporates penalizations for molecules that are not following some chemical criteria. Likewise, the intermediate rewards consists of validity rewards and adversarrial rewards.

To encourage the generated molecule resembling example molecules, a Generative Adversarial Network (GAN) with Graph Convolution Network [28] has been used to define an adversarial loss [64] as follows:

$$\min_\theta \max_\phi V\left(\pi_\theta, D_\phi\right) = \mathbb{E}_{x\sim p_{\text{data}}}\left[\log D_\phi(x)\right] + \mathbb{E}_{x\sim\pi_\theta}\left[\log D_\phi(1-x)\right] \tag{2.22}$$

The modification of the graph at each step can be modeled as the distribution of the transition between states: $p\left(s_{t+1} \mid s_t, \ldots, s_0\right) = \sum_{a_t} p\left(a_t \mid s_t, \ldots s_0\right) p\left(s_{t+1} \mid s_t, \ldots s_0, a_t\right)$ where the probability of action given states is represented as a policy network π_θ. The objective function of the policy gradient method is defined as follows to optimize a reward of molecule properties and adversarial training:

$$\max L^{\text{CLIP}}(\theta) = \mathbb{E}_t\left[\min\left(r_t(\theta)\hat{A}_t, \text{clip}\left(r_t(\theta), 1-\epsilon, 1+\epsilon\right)\hat{A}_t\right)\right], r_t(\theta) = \frac{\pi_\theta\left(a_t \mid s_t\right)}{\pi_{\theta_{\text{old}}}\left(a_t \mid s_t\right)} \tag{2.23}$$

2.4.3 Normalizing Flows for Graph Generation

Normalizing Flows [31] are another family of generative models. These models start from a simple base distribution (e.g., Normal distribution) sequentially implement a parameterized invertible deterministic functions in order to find the complex distribution behind the data. These models can be trained using maximum likelihood. One of the benefits of this method is that the invertible mapping allows the calculation of the exact data likelihood so that we can track the distribution of the data. The density function of the real-world data is computed as follows:

$$p_Z(z) = p_{\mathcal{E}}\left(f_\theta^{-1}(z)\right)\left|\det \frac{\partial f_\theta^{-1}(z)}{\partial z}\right| \tag{2.24}$$

where $f : \mathcal{E} \to \mathcal{Z}$ is an invertible transformation and $\epsilon \sim p_{\mathcal{E}}(\epsilon)$ is the base distribution. In the generative process of NF, for each data point z, the exact density can be calculated by $\epsilon = f_\theta^{-1}(z)$. For the sampling, z can be sampled from $p_Z(z)$ by first sample ϵ from $p_{\mathcal{E}}(\epsilon)$ and then feed ϵ to the feed forward network and calculate z

GraphAF [49] is an autoregressive and flow-based model that generates nodes and bonds based on the previous molecule substructures. This model can be trained end-to-end from a base distribution to the molecule graph. This sequential generation allows leveraging chemical knowledge for valency checking and incorporating chemical knowledge.

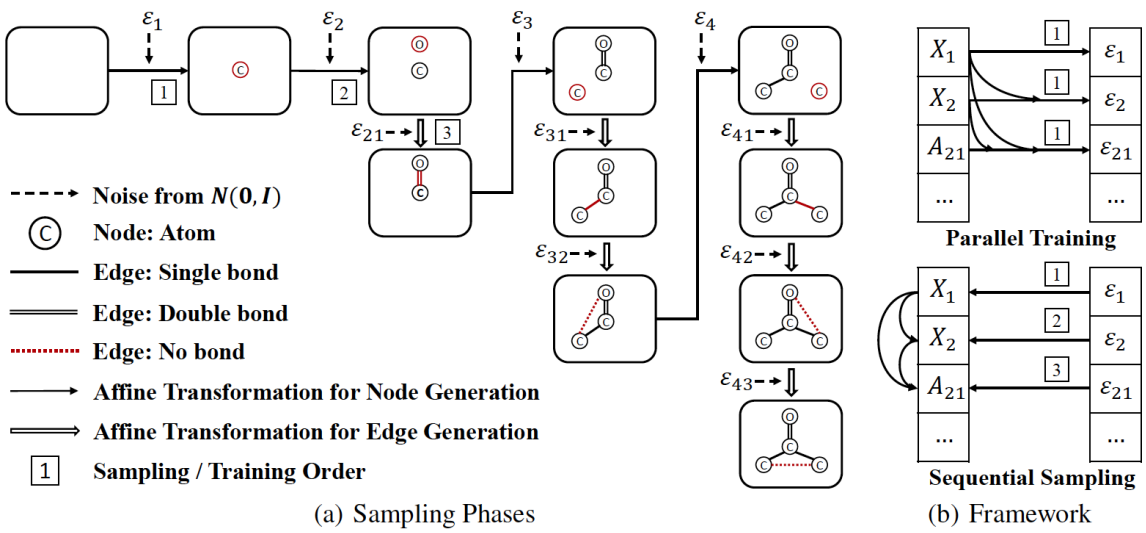
Despite the benefits of autoregressive models for molecule generation, the sequential generation performs slowly. [49] attempted to solve this challenge by calculating the exact likelihood of the mapping from data to latent space in parallel with the autoregressive model mapping the latent space to data.

GraphAF starts from an empty graph. At each step a new node X_i is generated according to $p(X_i|G_i)$. Then the edges between the current graph and the new node is generated sequentially according to $p(A_{ij}|G_i, X_i, A_{i,1:j-1}$, This process is illustrated in Figure 2.4.

In this method node feature is a one-hot vector that indicates the atom type. Therefore, both atom and edge feature are discrete that cannot be the input to a flow-based model. Dequantization teqnique [27] has been used to solve this issue. In this teqniqe discrete data is being converted to continuous by adding a real-valued noise. The conditional distribution for the generation is defined as follows:

$$\begin{aligned} p\left(z_i^X \mid G_i\right) =& \mathcal{N}\left(\mu_i^X, \left(\alpha_i^X\right)^2\right) \\ & \text{where } \mu_i^X = g_\mu x\left(G_i\right), \alpha_i^X = g_\alpha x\left(G_i\right) \end{aligned} \tag{2.25}$$

$$\begin{aligned} p\left(z_{ij}^A \mid G_i, X_i, A_{i,1:j-1}\right) = & \mathcal{N}\left(\mu_{ij}^A, \left(\alpha_{ij}^A\right)^2\right), j \in \{1, 2, \ldots, i-1\} \\ & \text{where } \mu_{ij}^A = g_{\mu^A}\left(G_i, X_i, A_{i,1:j-1}\right), \alpha_{ij}^A = g_{\alpha^A}\left(G_i, X_i, A_{i,1:j-1}\right) \end{aligned} \tag{2.26}$$



(a) Sampling Phases (b) Framework

Figure 2.4: An overview of the GraphAF method [49]

In the above equations, given G_i, the current subgraph, Relational GCN(R-GCN) [47], has been used for the message passing and compute the node embeddings and consequently, the sub-graph embedding. g_μ and g_α are positive and unconstrained scalar functions to calculate the mean and deviation. Here these functions are parameterized neural networks. GraphAF can leverage chemical rules such as valency constraint in each generation step. Here is the valency constraint for i^{th} and j^{th} atoms:

$$\sum_j |A_{ij}| \leq \text{ Valency } (X_i) \text{ and } \sum_i |A_{ij}| \leq \text{ Valency } (X_j) \tag{2.27}$$

2.5 Semi-supervised Learning for Molecular Graphs

Semi-supervised learning is particularly beneficial for chemistry applications due to the vast unlabeled chemical space and frequently available partially labeled data in the pharmaceutical and precision agriculture industry. ASGN [16] addressed the data scarcity problem using a semi-supervised method that predicts the most informative samples obtained by an active learning approach. ASGN [16] uses two Graph neural networks. A teacher model and a student model. For the teacher model, they train it in a semi-supervised learning fashion. The training is guided by the summation of three different losses. Reconstruction loss for node-level pseudo labels. The idea is that the property prediction model should be able to recover the atoms and bonds from the embedding. Clustering loss for graph-level pseudo labels. It learns the distribution of the molecular space. The last one is the property loss that is only optimized on the labeled dataset. These losses are optimized on both labeled and unlabelled data. For the student network, the weight is transferred from the teacher

network after training the semi-supervised teacher model. It is used to finetune the model for the downstream property prediction task. The active data selection component chooses the most informative data to add to the labeled data and retrain the model with the new labeled dataset. ASGN uses the molecule embeddings by the teacher model to select a small batch of the most diversified molecules in the chemical space by calculating the distance between molecule embeddings. In each iteration, they choose a subset of data that maximizes the distance between labeled and unlabeled sets. InfoGraph [54] is a semi-supervised graph-level representation learning method that has been evaluated on molecular property prediction. This method employs a student-teacher framework where the teacher model is trained on unlabeled data, and the student model is trained on the labeled data using a supervised objective. InfoGraph maximizes the mutual information between the representations learned by these two models so that the student model learns from the teacher model.

Semi-supervised learning using the generative models optimizes the prediction jointly with a Variational Autoencoder over the input data [24, 39, 50]. The latent representation is divided into a structured and unstructured part, where the structured part is enforced to represent labels of data points. Only part of the labels is provided during the training; therefore, the model learns the latent representation in a semi-supervised setting. This approach has achieved state-of-the-art performance on image classification [24, 39] and speech synthesis [15] with partially labeled data [26]. Kang et al. [21] proposed a method for conditionally generating molecules inspired by semi-supervised learning with generative models [24].

Chapter 3

Methodology

In this section, we first introduce our problem definition. Then we propose our method, SeMole, for semi-supervised Junction Tree Variational Autoencoder. Afterward, we present a modified version by adding a pretraining phase to training, which we call $\text{SeMole}_{\text{Pretrained}}$.

3.1 Problem Definition

Given a set of labeled molecular graphs $G_L = \{G_1, G_2, \dots G_l\}$ with corresponding property $\{y_1, y_2, ...y_l\}$ and a set of unlabeled molecular graphs $\{G_{l+1}, \dots, G_{l+u}\}$, our goal is to learn a model that predicts the set of labels $\{y_{l+1}, ..., y_{l+u}\}$ for the unlabeled molecular graphs as part of its latent representation. This model should be able to reconstruct the molecular graph G from the learned latent representation. Therefore, changing the labels will lead to generating new molecular graphs conditioned on the target labels.

3.2 SeMole

Our method extends Junction Tree Variational Autoencoder to a semi-supervised generative model for molecular graph property prediction. We propose a semi-supervised generative model consisting of three latent variables z_G, z_T, and y. Figure 3.1 shows the graphical model. y represents the molecular graph's high-level property, such as partially observed solubility. Following Junction Tree Variational Autoencoder, the molecular graph is decomposed to a junction tree by replacing each cluster with a node, and z_T represents the tree structure and the clusters in the tree. z_G represents these clusters' connectivity and how they are connected in the original graph. z_T and z_G are unobserved. y, z_T and z_G are sampled from a normal distribution.

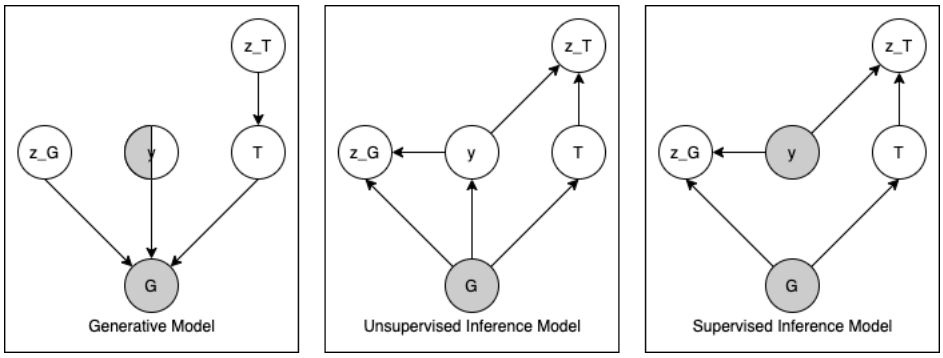


Figure 3.1: Probabilistic semi-supervised model on molecules represented by graphs and trees. **Left**: Generative Model. **Middle**: Unsupervised Inference Model. **Right**: Supervised Inference Model.

$$
\begin{aligned}
&p_\theta(y) = N(y|0,1) \\
&p_\theta(z_T) = N(z_T|0,1), \quad p_\theta(z_G) = N(z_G|0,1) \\
&p_\theta(G|z_G,T,y) = f(G;z_G,T,y,\theta), \quad p_\theta(T|z_T,y) = g(T;z_T,y,\theta)
\end{aligned}
\tag{3.1}
$$

Although molecular graph generation - $p_\theta(G|z_G,T,y)$ depends on junction tree T, which in turn depends on y already, a deliberate choice was made to make both molecular graph and junction tree generation process dependent on y. This is justified because some stereoisomers (and constitutional isomers) pairs can have drastically different molecular properties. It is the responsibility of the graph decoder to select the proper connectivity of the clusters that give rise to the isomer with the desired property. For that reason, latent variable y is also provided to the graph decoder.

3.2.1 Objective

Our goal is to approximately maximize the log-likelihood for both the tree structure and the molecular graph by maximizing a variational lower bound (ELBO) for observed and unobserved y. First, for the case where y is observed the objective function is denoted as follows:

$$
\begin{aligned}
\log p_\theta(G,T,y) \geq\ & E_{q_\phi(z_T,z_G|T,G,y)}\big[\log p_\theta(G|T,z_G,y) + \log p_\theta(T|z_T,y) \\
& + \log p_\theta(z_T) + \log p_\theta(z_G) + 2 * \log p_\theta(y) \\
& - \log q_\phi(z_T|T) - \log q_\phi(z_G|G)\big] \\
=\ & -\mathcal{L}(T,G,y)
\end{aligned}
\tag{3.2}
$$

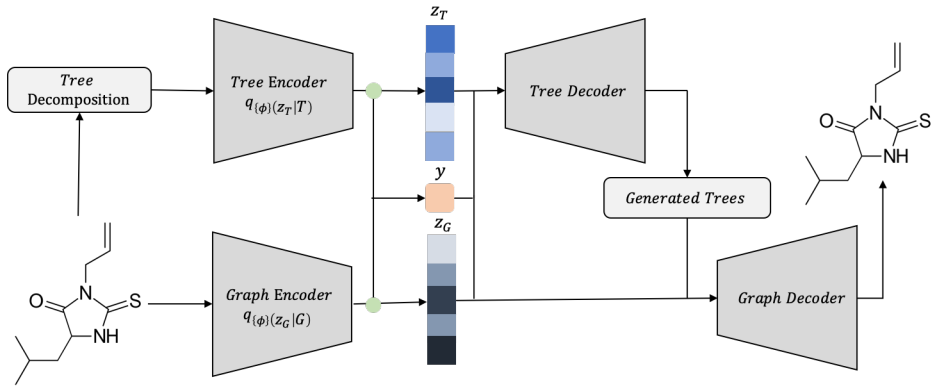


Figure 3.2: An overview of SeMole. Following JTVAE [19], The encoder consists of learning the latent representation of both molecular graph and junction tree. The tree encoding is designed to represent the subgraphs from the vocabulary of chemical bonds, rings, and individual atoms. Then the encoding of the molecule captures how these subgraphs should be connected. The target property, y, is also predicted by the representations learned by the encoders as a latent representation. The decoders take the concatenation of z_T and z_G with y and generate a tree-structure graph using valid subgraphs derived from a vocabulary of chemical bonds. The subgraphs in the tree are then assembled into a molecular graph.

For the case where the label corresponding to a data point is unobserved, we approximate $p_\theta(z_G, z_T, y|T, G)$ using a posterior function $q_\phi(z_G, z_T, y|T, G)$ modelled by neural networks. The detailed derivation could be found in the appendix. The unlabelled dataset objective is as follows:

$$\begin{aligned}
\log p_\theta(G,T) \geq E_{q_\phi(z_T,z_G,y|T,G)}\big[& \\
& \log p_\theta(G|T,z_G,y) + \log p_\theta(T|z_T,y) \\
& + \log p_\theta(z_T) + \log p_\theta(z_G) + 2 * \log p_\theta(y) \\
& - \log q_\phi(z_T|T) - \log q_\phi(z_G|G) - \log q_\phi(y|T,G)\big] \\
= & \sum_y \big[-\mathcal{L}(T,G,y)\big] + \mathcal{H}(q(y|T,G)) \\
= & -\mathcal{U}(T,G)
\end{aligned} \tag{3.3}$$

Following [24], it is desirable to add a loss so the distribution $q_\phi(y|T,G)$ can be learnt with the labelled dataset. This yields our final objective function:

$$\mathcal{J}^a = \sum_{(T,G,y)\sim\mathcal{D}} \mathcal{L}(T,G,y) + \sum_{(T,G)\sim\mathcal{D}} \mathcal{U}(T,G) + a \cdot E\big[\log q_\phi(y|T,G)\big] \tag{3.4}$$

The reconstruction loss follows the JTVAE[19] loss function consisting of a topological and label prediction for the tree structure and the prediction of correct subgraphs for

Algorithm 1 Learning in SeMole

$(\phi, \theta, W_G, W_T) \leftarrow$ initializeModelParameters()
while training() **do**
  $\mathcal{D} \leftarrow$ getRandomMiniBatch()
  $T_i \leftarrow$ decomposeToTree(G_i)
  $H_{Gi} \leftarrow$ encodeGraph(G_i, W_G)
  $H_{Ti} \leftarrow$ encodeTree(T_i, W_T)
  $y_i \sim q_\phi(y_i|H_{Gi}, H_{Ti})$
  $z_{Ti} \sim q_\phi(z_{Ti}|H_{Ti}, y_i) \quad z_{Gi} \sim q_\phi(z_{Gi}|H_{Gi}, y_i)$
  $\mathcal{J}^a \leftarrow$ Equation 3.4
  $(g_\phi, g_\theta, g_{W_G}, g_{W_T}) \leftarrow (\frac{\partial \mathcal{J}^a}{\partial \phi}, \frac{\partial \mathcal{J}^a}{\partial \theta}, \frac{\partial \mathcal{J}^a}{\partial W_G}, \frac{\partial \mathcal{J}^a}{\partial W_T})$
  $(\phi, \theta, W_G, W_T) \leftarrow (\phi, \theta, W_G, W_T) + \Gamma(g_\phi, g_\theta, g_{W_G}, g_{W_T})$
end while

the graph structure. However, unlike JTVAE[19], the graph decoder and the tree decoder generate data conditioned on the target property. The last term is a mean squared error loss that is added for supervised property prediction.

3.3 Optimization

The paramaters of SeMole's encoder, regressor and decoder components are optimized jointly using the objective function in Equation 3.4. Algorithm 1 is the pseudo code of SeMole's training algorithm.

For each iteration, a random mini batch is drawn from our dataset, either labelled or unlabelled; Molecular graphs are then decomposed into junction trees. Note that none of *decomposeToTree*(), *encodeGraph*() and *encodeTree*() are modified in this project. Our added regressor network - $q_\phi(y_i|H_{Gi}, H_{Ti})$ is modeled as a 2-layer feed-forward neural network that takes the input from the result of tree and graph encoding. Latent variables z_T and z_G are drawn from the normal distribution generated conditioned on y and their respective hidden layer H. The model parameters' gradient on objective function is computed for each random mini-batch of training samples and optimized using Adam [23].

3.4 Pretraining

Semi-supervised Variatiatioanl Autoencoders [24] are challenging to train end-to-end due to their multiple stochastic latent variables [39]. Since this model was unable to converge end-to-end, Kingma et al. stacked a pretrained feature extractor to their method, which improves the performance specifically on lower percentage of labeled data. Here we propose a pretraining version of SeMole called $\text{SeMole}_{\text{Pretrained}}$ by setting the coefficient of the supervised loss, α in Equation 3.4, to zero for the first ten training epochs. Since the supervised

loss decreases during the training instead of assigning α to a constant number, we gradually increase α during the training until it reaches the maximum.

pretraining enables this method to first train the Junction Tree Variational Autoencoder using the unsupervised part of the objective function. Then gradually add the supervised part of the objective function by increasing the value of α.

Chapter 4

Experiments

We evaluate the effectiveness of our proposed methods for two tasks: molecular property prediction and conditionally generating molecules with desired properties.

We vary the percentage of the labeled data (5%, 10%, 20%, and 50%) in the molecular property prediction experiments to evaluate the semi-supervised component of our method. We save 5% of the training set for validation. The property prediction task is evaluated by Mean Absolute Error (MAE) between the ground truth target property and the predicted property on the testing dataset, which consists of 10000 molecules.

Considering the prior distribution of target properties in the model, we normalize the properties to have a mean of 0 and a standard deviation of 1. We set the batch size to 16 and the learning rate to 0.001. The dimension of z_T and z_G is set to 56. The tree encoder consists of two GRU networks, and the graph decoder is a Message Passing Neural Network.

We use SSVAE [21] as the baseline which also is a semi-supervised variational autoencoder but representing molecules as SMILES representation instead of molecular graphs. The results are copied from the original paper since we use the same experimental design as our baseline. We further perform an ablation study for three different versions of our proposed model to assess the impact of semi-supervision and pretraining.

We developed a supervised version of SeMole by eliminating the unlabeled data from training to show the impact of semi-supervised learning as a solution to label scarcity. Therefore, $\text{SeMole}_{\text{Supervised}}$ only takes the labeled portion of the training data as the input. Our goal is to show the benefits of leveraging unlabeled data compared to supervised training.

We use the trained models on 50% labeled data to generate molecules conditioned on target properties and set the properties to different specific values. We also generate molecules unconditionally. Following our baseline [21], during the generation of molecules, we check the validity of the generated molecules using RDKit package [34], and we discard invalid molecules, or molecules that already exist in the training data or are already generated by the decoders before. We continue this process until we generate 3000 molecules or we reach the limit of 10000 generated molecules. Then we label the generated molecules for the target properties to assess whether their properties are close to the target properties.

We perform further experiments on another dataset to compare the SeMole's performance on Semi-Supervised molecule property prediction with two state-of-the-art semi-supervised GNNs. We used InfoGraph [54] and ASGN [16] as our baselines for this task. InfoGraph focuses on effectively learning whole graph representations in an unsupervised or semi-supervised fashion. The method maximizes the mutual information between the entire graph representation and all the substructure representations for the unsupervised task. The semi-supervised version consists of two separate encoders, similar to the unsupervised version. On top of that idea here, the model also maximizes the mutual information of the representations learned by the two encoders at all levels(layers). The main difference between this method and SeMole is that InfoGraph maximizes the mutual information between supervised and unsupervised encoders; however, SeMole jointly learns the supervised and unsupervised network and predicts the target as part of the latent representation. For this experiment, following the baselines, we randomly chose 10000 samples for testing, 10000 samples for validation, and 5000 samples as labeled for training. We kept the rest of the data unlabeled. Similar to previous experiments all molecule properties are normalized to have a mean of 0 and a standard deviation of 1. We evaluate Mean Absolute Error(MAE) while minimizing Mean Squared Error(MSE) in training.

4.1 Dataset

We use 310000 drug-like molecules sampled from ZINC [53]. We use the same dataset that was also used to evaluate SSVAE [21]. Following the literature, we use three chemical properties that are available using RDKit Package[34], i.e., molecular weight (MolWt), Wildman-Crippen partition coefficient (LogP), and quantitative estimation of drug-likeness (QED) which are generaly used for this task in the literature.

We also perform experiments on QM9 dataset [44] which includes 134000 molecules and 12 pre-computed quantum mechanical properties. These molecules contain 9 non-hydrogen (heavy) atoms at most.

4.2 Results

Table 4.1 demonstrates the MAE of the molecule property prediction task with the varying number of labels in the training dataset. We repeated the experiment three times by training the model with a new seed number each time and reported the average and standard deviation of the MAE. The pretrained version of SeMole outperformed SSVAE in most of the cases. $\text{SeMole}_{\text{Pretrained}}$ achieved better performance on LogP and QED compared to MolWt. The results show the effectiveness of the pretraining since $\text{SeMole}_{\text{Pretrained}}$ outperforms SeMole in all cases except in two experiments where SeMole performs slightly better. $\text{SeMole}_{\text{Pretrained}}$ substantially outperforms $\text{SeMole}_{\text{Supervised}}$ showing the benefit of semi-supervision while having limited labeled data. The performance of $\text{SeMole}_{\text{Pretrained}}$

Table 4.1: Mean Absolute Error(MAE) of Molecule Property Prediction with Varying Percentage of Labeled data. $\text{SeMole}_{\text{Pretrained}}$ outperformed SSVAE in most of the cases showing the benefit of representing molecules as graphs and the pretraining of semi-supervised generative models for molecular property prediction with partial supervision. $\text{SeMole}_{\text{Pretrained}}$ substantially outperforms $\text{SeMole}_{\text{Supervised}}$ showing the benefit of semi-supervision while having limited labeled data.

% Labeled	target property	SSVAE	$\text{SeMole}_{\text{Pretrained}}$	SeMole	$\text{SeMole}_{\text{Supervised}}$
5%	MolWt	$\mathbf{1.639 \pm 0.577}$	1.658 ± 0.130	1.642 ± 0.165	1.894 ± 0.131
	LogP	0.120 ± 0.006	$\mathbf{0.117 \pm 0.001}$	0.133 ± 0.002	0.154 ± 0.004
	QED	0.028 ± 0.001	$\mathbf{0.021 \pm 0.000}$	0.038 ± 0.000	0.057 ± 0.000
10%	MolWt	1.444 ± 0.618	$\mathbf{1.350 \pm 0.139}$	1.419 ± 0.149	1.597 ± 0.126
	LogP	0.090 ± 0.004	0.092 ± 0.001	$\mathbf{0.089 \pm 0.000}$	0.127 ± 0.006
	QED	0.021 ± 0.001	$\mathbf{0.017 \pm 0.000}$	0.024 ± 0.000	0.046 ± 0.000
20%	MolWt	$\mathbf{1.008 \pm 0.370}$	1.069 ± 0.088	1.187 ± 0.119	1.286 ± 0.121
	LogP	0.071 ± 0.007	$\mathbf{0.070 \pm 0.000}$	0.076 ± 0.000	0.091 ± 0.001
	QED	0.016 ± 0.001	0.012 ± 0.000	$\mathbf{0.011 \pm 0.000}$	0.019 ± 0.000
50%	MolWt	1.050 ± 0.164	$\mathbf{1.049 \pm 0.041}$	1.054 ± 0.387	1.243 ± 0.273
	LogP	0.047 ± 0.003	$\mathbf{0.043 \pm 0.000}$	0.047 ± 0.000	0.061 ± 0.000
	QED	0.010 ± 0.001	$\mathbf{0.009 \pm 0.000}$	0.011 ± 0.000	0.013 ± 0.000

compared to SSVAE shows the benefit of representing molecules as graphs and the pretraining of semi-supervised generative models for molecular property prediction with partial supervision.

Table 4.2 compares the efficacy of conditional generation of molecules for these methods. The table demonstrates the percentage of valid, unique, and novel molecules generated by these methods. The results show that SSVAE and $\text{SeMole}_{\text{Pretrained}}$ demonstrate better performance than $\text{SeMole}_{\text{Supervised}}$ and SeMole. The main distinction between $\text{SeMole}_{\text{Pretrained}}$ and SSVAE is that $\text{SeMole}_{\text{Pretrained}}$ generates 100% valid molecules. $\text{SeMole}_{\text{Pretrained}}$ also outperforms SSVAE on the percentage of the generated molecules with properties within the threshold of 5% difference from the target values.

Table 4.3 compares the MAE of the semi-supervised molecule property prediction task for $\text{SeMole}_{\text{Pretrained}}$, InfoGraph, and ASGN. $\text{SeMole}_{\text{Pretrained}}$ marginally outperforms InfoGraph in most cases; however, ASGN still significantly outperforms both methods. We believe the main reason for the better performance in ASGN is related to the Active Learning component allowing the model to choose the most representative samples to label. In this experiment, although we start with the same number of labeled samples for ASGN and SeMole, the number of samples increases by the active learning component in ASGN while it remains constant in SeMole. The number of labeled samples starts from 5000 in both ASGN and SeMole but reaches 70,000 in ASGN while remaining constant in SeMole through training. We hypothesize that SeMole outperforms InfoGraph since the architecture of Junction Tree Variational Autoencoders enables the model to learn the representation of

Table 4.2: Percentage of novel, unique, valid generated molecules. Novel molecules are those generated molecules that do not exist in the training dataset. Molecules are evaluated by the RDKit package [34] to determine if they represent a valid molecule structure. Unique molecules are considered generated molecules that are not duplicated

model	target property	novel molecules	unique molecules	validity
SSVAE	unconditional generation	95.0	98.5	99.3
	MolWt = 250	95.7	78.3	99.6
	MolWt = 350	96.2	86.9	99.5
	MolWt = 450	93.4	79.1	99.3
	LogP = 1.5	96.3	91.2	99.2
	LogP = 3	95.2	92.8	96.4
	LogP = 4.5	94.7	91.7	99.1
	QED = 0.5	95.1	93.2	98.6
	QED = 0.7	94.9	96.3	99.4
	QED = 0.9	96.1	97.0	99.6
$\text{SeMole}_{\text{Pretrained}}$	unconditional generation	94.6	98.0	100
	MolWt = 250	94.3	80.9	100
	MolWt = 350	93.6	84.1	100
	MolWt = 450	95.6	80.7	100
	LogP = 1.5	93.2	92.0	100
	LogP = 3	96.6	91.9	100
	LogP = 4.5	95.6	90.8	100
	QED = 0.5	95.0	94.7	100
	QED = 0.7	92.7	95.4	100
	QED = 0.9	98.1	97.8	100
SeMole	unconditional generation	89.1	96.4	100
	MolWt = 250	93.8	78.0	100
	MolWt = 350	90.9	82.6	100
	MolWt = 450	94.2	90.3	100
	LogP = 1.5	90.2	87.4	100
	LogP = 3	89.5	86.4	100
	LogP = 4.5	92.9	89.0	100
	QED = 0.5	93.9	90.8	100
	QED = 0.7	91.8	90.4	100
	QED = 0.9	93.0	92.9	100
$\text{SeMole}_{\text{Supervised}}$	unconditional generation	91.8	95.9	100
	MolWt = 250	94.8	81.6	100
	MolWt = 350	92.8	83.0	100
	MolWt = 450	94.7	88.6	100
	LogP = 1.5	88.3	89.0	100
	LogP = 3	86.9	86.3	100
	LogP = 4.5	90.4	91.7	100
	QED = 0.5	92.0	91.8	100
	QED = 0.7	90.7	89.2	100
	QED = 0.9	92.9	90.0	100

Table 4.3: Mean Absolute Error(MAE) of semi-supervised experiments on the QM9 dataset. The table compares the MAE of the semi-supervised molecule property prediction task for $\text{SeMole}_{\text{Pretrained}}$, InfoGraph, and ASGN. Target properties are all scalar values of quantum mechanical properties for molecules in the QM9 dataset. The property values are higher in some cases, such as R2, and lower in other properties, like ZPVE. However, all properties were normalized for training. $\text{SeMole}_{\text{Pretrained}}$ marginally outperforms InfoGraph in most cases; however, ASGN still significantly outperforms both methods benefiting from the active learning component.

target	$\text{SeMole}_{\text{Pretrained}}$	InfoGraph	ASGN
Mu (0)	0.2781 ± 0.0239	0.3168	**0.1947**
Alpha (1)	0.4908 ± 0.0017	0.5444	**0.2818**
HOMO (2)	0.1426 ± 0.0060	0.1605	**0.1190**
LUMO (3)	0.1992 ± 0.0018	0.1659	**0.1061**
Gap (4)	0.2331 ± 0.0240	0.2421	**0.2012**
R2 (5)	6.08 ± 0.37	4.92	**1.38**
ZPVE(6)	0.00023 ± 0.00002	0.00036	**0.00017**
U0 (7)	0.1191 ± 0.0089	0.1410	**0.0562**
U (8)	0.1413 ± 0.0191	0.1702	**0.0594**
H (9)	0.1709 ± 0.0019	0.1552	**0.0583**
G(10)	0.1242 ± 0.0059	0.1592	**0.0560**
Cv (11)	0.1690 ± 0.0182	0.1965	**0.0984**

molecules based on a vocabulary of chemical subgraphs associated with molecule properties, while Graph Convolution Networks used by InfoGraph do not incorporate domain-specific subgraphs.

Chapter 5

Conclusion

We have proposed SeMole, a semi-supervised generative model for molecular graphs. we augmented a state-of-the-art generative model, JTVAE, for molecular graphs with semi-supervised learning. We also added a pretraining phase to improve the training of the semi-supervised generative model. We performed experiments on molecular property prediction and conditional generation only using limited labeled data. These experiments were performed on three properties in the ZINC dataset and twelve quantum mechanical properties in the QM9 dataset. The $\text{SeMole}_{\text{Pretrained}}$ outperforms SSVAE on most molecule property prediction tasks and generates 100% valid molecules conditioned on target properties. The ablation study showed the effectiveness of semi-supervision over supervised methods when labeled data is limited. We also demonstrated the efficacy of our pretraining phase for training a Semi-supervised VAE. We further showed that $\text{SeMole}_{\text{Pretrained}}$ outperforms, InfoGraph, a state-of-the-art semi-supervised GNN in most cases. However, an active learning component improves the performance of the semi-supervised GNN significantly which shows the importance of choosing the right samples for feedback on the semi-supervised model. One of the limitations of this method is the training time. Although the computational complexity of JTVAE is linear in the number of clusters in the trees, the high number of parameters and the complex architecture of the model causes slow training on datasets such as ZINC.

This thesis suggests several directions for future research. $\text{SeMole}_{\text{Pretrained}}$, like other methods in the literature, predicts only one molecular property and generates molecules conditioned on a single property. However, predicting multiple molecular properties could improve the accuracy of the prediction and also is beneficial to discover molecules with multiple target properties. Moreover, extending the 2D representation of molecules to 3D representation could help learn richer latent representations, which lead to more accurate predictions and improved molecule generation [2]. The challenge for incorporating the 3D structure of molecules is that for most molecules, their 3D structures are not available. Deep generative models have been developed to predict valid and stable conformations by

taking 3D coordinates for each atom as the input [61]. Incorporating molecule properties in generating 3D structures is still mainly unexplored.

The generalization power of a machine learning model is even more important in drug discovery since a model that performs well in one region of the chemical space does not necessarily has the same performance in another part of the chemical space. Therefore random splitting that is used in state-of-the-art machine learning in many cases can not beat more traditional methods still used by domain experts due to the arbitrary choice of train and test sample. For example, scaffold splitting is used in drug discovery applications instead of random splitting which splits the chemical space based on molecular graph similarities [63]. Evaluating the semi-supervised molecule property prediction based on different ways of splitting the data could be a beneficial step toward developing models with better generalization.

Bibliography

[1] Simon Axelrod and Rafael Gomez-Bombarelli. Geom, energy-annotated molecular conformations for property prediction and molecular generation. *Scientific Data*, 9(1):1–14, 2022.

[2] Camille Bilodeau, Wengong Jin, Tommi Jaakkola, Regina Barzilay, and Klavs F Jensen. Generative models for molecular discovery: Recent advances and challenges. *Wiley Interdisciplinary Reviews: Computational Molecular Science*, page e1608, 2022.

[3] Aleksandar Bojchevski, Oleksandr Shchur, Daniel Zügner, and Stephan Günnemann. Netgan: Generating graphs via random walks. *arXiv preprint arXiv:1803.00816*, 2018.

[4] John Bradshaw, Brooks Paige, Matt J Kusner, Marwin Segler, and José Miguel Hernández-Lobato. A model to search for synthesizable molecules. In *Advances in Neural Information Processing Systems*, pages 7937–7949, 2019.

[5] Hongming Chen, Ola Engkvist, Yinhai Wang, Marcus Olivecrona, and Thomas Blaschke. The rise of deep learning in drug discovery. *Drug discovery today*, 23(6):1241–1250, 2018.

[6] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[7] Hanjun Dai, Yingtao Tian, Bo Dai, Steven Skiena, and Le Song. Syntax-directed variational autoencoder for structured data. *arXiv preprint arXiv:1802.08786*, 2018.

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

[9] David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alan Aspuru-Guzik, and Ryan P Adams. Convolutional networks on graphs for learning molecular fingerprints. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

[10] Gökcen Eraslan, Žiga Avsec, Julien Gagneur, and Fabian J Theis. Deep learning: new computational modelling techniques for genomics. *Nature Reviews Genetics*, 20(7):389–403, 2019.

[11] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International Conference on Machine Learning*, pages 1263–1272. PMLR, 2017.

[12] 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.

[13] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. The MIT Press, 2016.

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

[15] Raza Habib, Soroosh Mariooryad, Matt Shannon, Eric Battenberg, RJ Skerry-Ryan, Daisy Stanton, David Kao, and Tom Bagby. Semi-supervised generative modeling for controllable speech synthesis. *arXiv preprint arXiv:1910.01709*, 2019.

[16] Zhongkai Hao, Chengqiang Lu, Zhenya Huang, Hao Wang, Zheyuan Hu, Qi Liu, Enhong Chen, and Cheekong Lee. Asgn: An active semi-supervised graph neural network for molecular property prediction. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 731–752, 2020.

[17] Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay Pande, and Jure Leskovec. Strategies for pre-training graph neural networks. *arXiv preprint arXiv:1905.12265*, 2019.

[18] Shaoxiong Ji, Shirui Pan, Erik Cambria, Pekka Marttinen, and S Yu Philip. A survey on knowledge graphs: Representation, acquisition, and applications. *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[19] Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Junction tree variational autoencoder for molecular graph generation. In *International Conference on Machine Learning*, pages 2323–2332. PMLR, 2018.

[20] Wengong Jin, Kevin Yang, Regina Barzilay, and Tommi Jaakkola. Learning multimodal graph-to-graph translation for molecular optimization. *arXiv preprint arXiv:1812.01070*, 2018.

[21] Seokho Kang and Kyunghyun Cho. Conditional molecular design with deep generative models. *Journal of chemical information and modeling*, 59(1):43–52, 2018.

[22] Steven Kearnes, Kevin McCloskey, Marc Berndl, Vijay Pande, and Patrick Riley. Molecular graph convolutions: moving beyond fingerprints. *Journal of computer-aided molecular design*, 30(8):595–608, 2016.

[23] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[24] Diederik P Kingma, Danilo J Rezende, Shakir Mohamed, and Max Welling. Semi-supervised learning with deep generative models. *arXiv preprint arXiv:1406.5298*, 2014.

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

[26] Diederik P Kingma and Max Welling. An introduction to variational autoencoders. *arXiv preprint arXiv:1906.02691*, 2019.

[27] Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in neural information processing systems*, pages 10215–10224, 2018.

[28] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[29] Thomas N Kipf and Max Welling. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016.

[30] Peter Kirkpatrick and Clare Ellis. Chemical space. *Nature*, 432(7019):823–824, 2004.

[31] Ivan Kobyzev, Simon JD Prince, and Marcus A Brubaker. Normalizing flows: An introduction and review of current methods. *IEEE transactions on pattern analysis and machine intelligence*, 43(11):3964–3979, 2020.

[32] Nevan J Krogan, Gerard Cagney, Haiyuan Yu, Gouqing Zhong, Xinghua Guo, Alexandr Ignatchenko, Joyce Li, Shuye Pu, Nira Datta, Aaron P Tikuisis, et al. Global landscape of protein complexes in the yeast saccharomyces cerevisiae. *Nature*, 440(7084):637–643, 2006.

[33] Matt J Kusner, Brooks Paige, and José Miguel Hernández-Lobato. Grammar variational autoencoder. In *International Conference on Machine Learning*, pages 1945–1954. PMLR, 2017.

[34] Greg Landrum. Rdkit: Open-source cheminformatics software. 2016.

[35] Qi Liu, Miltiadis Allamanis, Marc Brockschmidt, and Alexander Gaunt. Constrained graph variational autoencoders for molecule design. *Advances in neural information processing systems*, 31, 2018.

[36] Shengchao Liu, Hanchen Wang, Weiyang Liu, Joan Lasenby, Hongyu Guo, and Jian Tang. Pre-training molecular graph representation with 3d geometry. *arXiv preprint arXiv:2110.07728*, 2021.

[37] Tengfei Ma, Jie Chen, and Cao Xiao. Constrained generation of semantically valid graphs via regularizing variational autoencoders. *arXiv preprint arXiv:1809.02630*, 2018.

[38] Yao Ma and Jiliang Tang. *Deep Learning on Graphs*. Cambridge University Press, 2021.

[39] Lars Maaløe, Casper Kaae Sønderby, Søren Kaae Sønderby, and Ole Winther. Auxiliary deep generative models. In *International conference on machine learning*, pages 1445–1453. PMLR, 2016.

[40] Andrea Mauri, Viviana Consonni, Manuela Pavan, and Roberto Todeschini. Dragon software: An easy approach to molecular descriptor calculations. *Match*, 56(2):237–248, 2006.

[41] Ali Bou Nassif, Ismail Shahin, Imtinan Attili, Mohammad Azzeh, and Khaled Shaalan. Speech recognition using deep neural networks: A systematic review. *IEEE access*, 7:19143–19165, 2019.

[42] Marcus Olivecrona, Thomas Blaschke, Ola Engkvist, and Hongming Chen. Molecular de-novo design through deep reinforcement learning. *Journal of cheminformatics*, 9(1):1–14, 2017.

[43] Mariya Popova, Olexandr Isayev, and Alexander Tropsha. Deep reinforcement learning for de novo drug design. *Science advances*, 4(7):eaap7885, 2018.

[44] Raghunathan Ramakrishnan, Pavlo O Dral, Matthias Rupp, and O Anatole Von Lilienfeld. Quantum chemistry structures and properties of 134 kilo molecules. *Scientific data*, 1(1):1–7, 2014.

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

[46] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2008.

[47] 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*, pages 593–607. Springer, 2018.

[48] Kristof T Schütt, Huziel E Sauceda, P-J Kindermans, Alexandre Tkatchenko, and K-R Müller. Schnet–a deep learning architecture for molecules and materials. *The Journal of Chemical Physics*, 148(24):241722, 2018.

[49] Chence Shi, Minkai Xu, Zhaocheng Zhu, Weinan Zhang, Ming Zhang, and Jian Tang. Graphaf: a flow-based autoregressive model for molecular graph generation. *arXiv preprint arXiv:2001.09382*, 2020.

[50] Narayanaswamy Siddharth, Brooks Paige, Jan-Willem Van de Meent, Alban Desmaison, Noah D Goodman, Pushmeet Kohli, Frank Wood, and Philip HS Torr. Learning disentangled representations with semi-supervised deep generative models. *arXiv preprint arXiv:1706.00400*, 2017.

[51] Gregor Simm, Robert Pinsler, and José Miguel Hernández-Lobato. Reinforcement learning for molecular design guided by quantum mechanics. In *International Conference on Machine Learning*, pages 8959–8969. PMLR, 2020.

[52] Martin Simonovsky and Nikos Komodakis. Graphvae: Towards generation of small graphs using variational autoencoders. In *International conference on artificial neural networks*, pages 412–422. Springer, 2018.

[53] Teague Sterling and John J Irwin. Zinc 15–ligand discovery for everyone. *Journal of chemical information and modeling*, 55(11):2324–2337, 2015.

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

[55] Raphael JL Townshend, Martin Vögele, Patricia Suriana, Alexander Derry, Alexander Powers, Yianni Laloudakis, Sidhika Balachandar, Bowen Jing, Brandon Anderson, Stephan Eismann, et al. Atom3d: Tasks on molecules in three dimensions. *arXiv preprint arXiv:2012.04035*, 2020.

[56] Athanasios Voulodimos, Nikolaos Doulamis, Anastasios Doulamis, and Eftychios Protopapadakis. Deep learning for computer vision: A brief review. *Computational intelligence and neuroscience*, 2018, 2018.

[57] Xiang Wang, Xiangnan He, Yixin Cao, Meng Liu, and Tat-Seng Chua. Kgat: Knowledge graph attention network for recommendation. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 950–958, 2019.

[58] David Weininger. Smiles, a chemical language and information system. 1. introduction to methodology and encoding rules. *Journal of chemical information and computer sciences*, 28(1):31–36, 1988.

[59] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1):4–24, 2020.

[60] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. A comprehensive survey on graph neural networks. *CoRR*, abs / 1901.00596, 2019.

[61] Louis-Pascal Xhonneux, Meng Qu, and Jian Tang. Continuous graph neural networks. In *International Conference on Machine Learning*, pages 10432–10441. PMLR, 2020.

[62] Minkai Xu, Shitong Luo, Yoshua Bengio, Jian Peng, and Jian Tang. Learning neural generative dynamics for molecular conformation generation. *arXiv preprint arXiv:2102.10240*, 2021.

[63] Kevin Yang, Kyle Swanson, Wengong Jin, Connor Coley, Philipp Eiden, Hua Gao, Angel Guzman-Perez, Timothy Hopper, Brian Kelley, Miriam Mathea, et al. Analyzing learned molecular representations for property prediction. *Journal of chemical information and modeling*, 59(8):3370–3388, 2019.

[64] Jiaxuan You, Bowen Liu, Rex Ying, Vijay Pande, and Jure Leskovec. Graph convolutional policy network for goal-directed molecular graph generation. *arXiv preprint arXiv:1806.02473*, 2018.

[65] Jiaxuan You, Rex Ying, Xiang Ren, William Hamilton, and Jure Leskovec. Graphrnn: Generating realistic graphs with deep auto-regressive models. In *International conference on machine learning*, pages 5708–5717. PMLR, 2018.