GRAPH NEURAL NETWORKS FOR PROPERTY-GUIDED MOLECULE OPTIMIZATION

by

Dat Quoc Ngo

APPROVED BY SUPERVISORY COMMITTEE:

Jessica Jin Ouyang, Chair

Feng Chen

Latifur Khan

This thesis class file

is dedicated to my students,

who suffered without a proper one

until the present time.

GRAPH NEURAL NETWORKS FOR PROPERTY-GUIDED MOLECULE OPTIMIZATION

by

DAT QUOC NGO, BS

THESIS

Presented to the Faculty of

The University of Texas at Dallas

in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE IN

COMPUTER SCIENCE

THE UNIVERSITY OF TEXAS AT DALLAS

December 2022

ACKNOWLEDGMENTS

I would like to express my gratefulness to Dr. Stefan C. Mihaela, a Chemistry and Bio-Chemistry Professor, and her students in gathering molecule data and providing valuable feedback on my thesis.

November 2022

GRAPH NEURAL NETWORKS FOR PROPERTY-GUIDED MOLECULE OPTIMIZATION

Dat Quoc Ngo, MS
The University of Texas at Dallas, 2022

Supervising Professor: Jessica Jin Ouyang, Chair

This mock dissertation concerns the development and usage of a LaTeX class file that eases the task of creating UTD theses and dissertations. The class file automatically creates margins, page headers and footers, page numbers, paragraph parameters, title pages, and table/figure captions consistent with the guidelines set forth by the UTD graduate school. In contrast to many prior works, care has been taken to respect relevant LaTeX coding conventions and standards. This helps to maximize compatibility with other LaTeX packages, and eases the incorporation of existing publication texts into a dissertation master document.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

Light has emerged as an alternative and clean source of energy in the fight against climate change. However, light intensity of indoor environments is significantly lower than that of the outdoor environment (sunlight). Therefore, to produce electricity as efficiently as in the outdoor environment, indoor light energy production requires new materials, such as organic photovoltaic (OPV) molecules, to produce specialized cells capable of harvesting energy from less intense light and with limited panel surface area.

The evolution of such material design has been largely driven by domain expert researchers and incremental modification of materials with much trial and error. Computational approaches can dramatically reduce the amout of time and human labor required to design new materials. In this thesis, we are specifically interested in molecule exploration, a computational sub-field of material design, for OPV. The molecule exploration process includes two stages: generating molecule candidates and verifying their performance (also called screening molecules). Early research efforts were invested into accelerating the performance verification of molecule candidates to avoid high-cost evaluation via physical experiments. Researchers utilized screening tools, combining quantum chemistry calculations and cheminformatics, to virtually identify promising molecule candidates by determining whether their properties satisfy the desired properties. Only promising molecules are verified by physical experiments.

Due to the high computational cost of the quantum-chemistry screening process, research effort has been invested to improve the screening tools to accelerate molecule exploration. Machine learning (ML) has been widely applied to accelerate molecule exploration with two main approaches: (1) ML-based screening to predict the properties of new, human-designed molecules to avoid the high computational cost of the standard screening process, and (2) deep generative models to propose new molecules that meet the desired properties. In this

thesis, we design a deep generative model for generating OPV molecules while controlling for two properties related to their energy capture efficiency: the *highest occupied molecular orbital* (HOMO) and *lowest unoccupied molecular orbital* (LUMO) scores.

ML-based molecule exploration algorithms are generally designed for one of two common molecule representations (see Figure 1.1): Simplified Molecular Input Line Entry System (SMILES) (Weininger, D., 1988), a string representation of a molecule, and the Lewis structure, a graph consisting of atom nodes and bond edges (from this point, we refer Lewis structures as "molecular graphs" for simplicity). Given a candidate molecule in either the SMILES or graph representation, past work on ML-based screening algorithms used cheminformatics to extract its graphical and chemical descriptors, a process that requires domain expertise to identify relevant features. In more recent work, it is possible to avoid manual feature selection by using deep learning. (Chen et al., 2019) proposed a graph neural network (GNN) framework to learn attributes of atoms, bonds, and their connectivity represented by the molecular graph and to yield regression outputs for the corresponding properties.

Following recent work (Jin et-al., 2018, 2019, 2020), this thesis focuses on graph-based generative models that generate molecular graphs using graph substructures called *motifs*. Besides proposing new OPV molecules, our goal is to ensure they have the desired HOMO and LUMO properties. Hence, we can formulate our problem as a property-guided molecule optimization task that iteratively modifies candidate molecules towards target HOMO/LUMO values. Compared with normal organic molecules in existing datasets, OPV molecules are relatively larger by the number of atoms and bonds (see Table 4.3 for statistics of molecule datasets). The size of the molecules requires more decoding steps such that generative models are prone to make errors. Hence, we adopt the hierarchical, motif-based Variational AutoEncoder (VAE) proposed by (Jin et-al., 2020) to serve as our model's backbone. By representing molecular graphs using motifs (i.e. subgraphs), rather than single atoms, we significantly reduce the number of decoding steps. To fit the OPV domain, we pretrain the

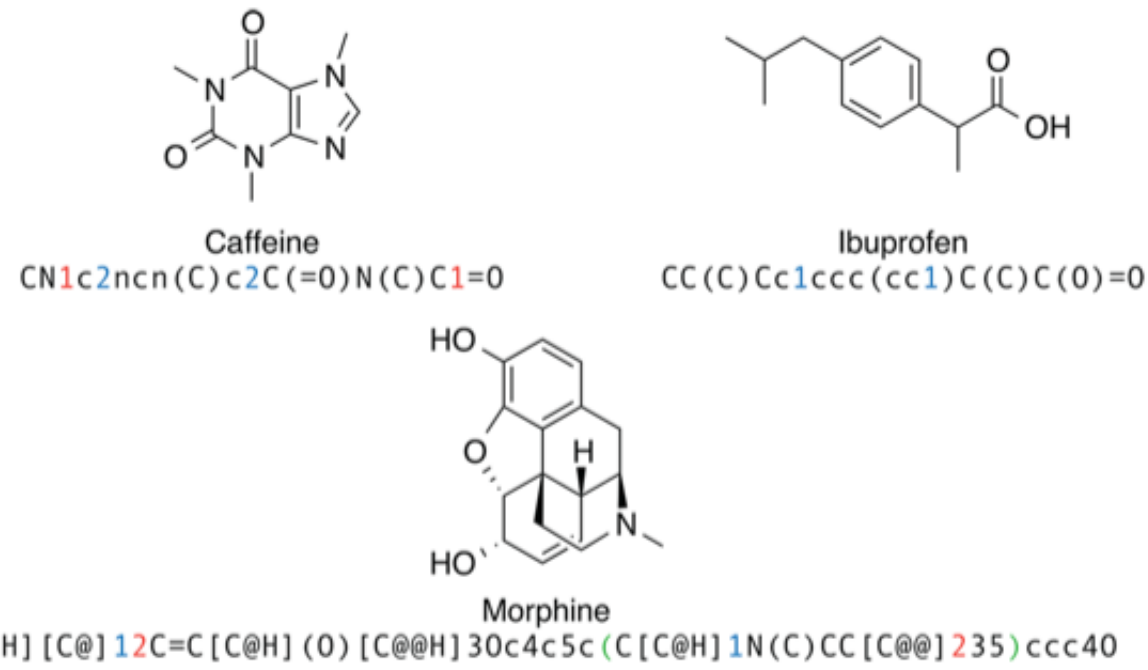


Figure 1.1: Examples of molecules and their SMILES representation (Segler et-al., 2018).

backbone on the molecule reconstruction task using a mixture of existing OPV datasets and a small set of human-curated OPV molecules from our university's Chemistry Department 4.1.

Additionally, following work in (Takeda et al., 2019; Jin et-al., 2019, 2020), we pair the backbone with two dense neural networks that are jointly trained on HOMO and LUMO regression objectives, while the backbone continues to be fine-tuned on the molecule reconstruction objective. Then, during inference, we use the property regressors to search for candidate latent vectors that yield molecules with the desired HOMO/LUMO values. Bayesian optimization and genetic algorithms are two popular, derivative-free latent-vector-searching methods for their data efficiency (i.e., time or iterations to reach global optimum) in the low-dimensional latent search space. Their efficiency quickly degrades when the latent vector size becomes larger to capture more information of large molecules like OPVs. Hence,

we propose to individually optimize the property regressors to seek for the candidate feature vectors.

The main contributions of this thesis are as following:

- We develop a hierarchical generative model tailored for organic photovoltaic molecules.
- We propose the property-guided optimization to seek for candidate molecules, using property regressors to control molecule generation.

CHAPTER 2

LITERATURE REVIEW

Molecular generative models are generally categorized by two common molecular representations: SMILES and molecular graph. The graph representation can be further decomposed into the atom and motif levels. Unlike the left-to-right order of characters in SMILES, the order of atom or motif nodes in molecular graphs is not guaranteed to be unique despite fixed starting nodes. The selection of the graph serializing algorithms affects the robustness and performance of the graph-based generative modeling.

Optimizing pretrained generative models is necessary to seek for new molecules satisfying desired properties. The optimization process is commonly categorized into two approaches, exhaustive and unsupervised searching, that attempt to look for special latent vectors representing desired properties. Evaluating the molecule generation is not a trivial step because a molecule may have different node orderings. Besides the molecule generation metrics (e.g. Frechet distance to measure similarity between true and output molecules), the molecular evaluation considers generally chemical metrics (e.g. *validity* for valid molecules with correct atoms and bonds) and domain-specified metrics (e.g. *druglikelikness* for drug design).

In the following sections, we discuss recent work on two common (*SMILES-based* and *graph-based*) molecular generation approaches, and the *graph serializing* methods. for molecular generation. The graph-based molecular generation is sub-categorized into *atom-based* and *motif-based* such that atoms or motifs serve as the atomic generation units.

2.1 Molecular Generators

2.1.1 SMILES-based Molecular Generation

SMILES is a string notation representing the graphical Lewis structures of molecules in two dimensions. SMILES grammatically describes molecules with an alphabet of characters to

represent atoms (e.g. c and C for carbons, o and O for oxygen) and -, =, and # for single, double, and triple bonds (See Figure 1.1 for examples). To indicate rings (cycles), a number is inserted at the two atoms where the ring is closed. An alternative string-formatted molecular representation is InChl (Heller et-al., 2013), which adds chemical property information (e.g. electronic charge) not present in SMILES. However, SMILES is a more popular choice for the molecule representation and for input to molecular generative models.

Since SMILES is a string format, SMILES-based molecular generative models (Gomez-Bombarelli et-al., 1988; Segler et-al., 2018; Kusner et-al., 2017; Dai et-al., 2018; Guimaraes et-al., 2018; Olivecrona, Blaschke, Engkvist, and Chen, 2017) are language-like generative models that learn SMILES grammar and keep track of long-term dependencies (i.e. numbers and brackets) to eventually close rings and chains. Recurrent Neural Networks (RNNs) are commonly used for modeling text and the linear input formats like SMILES. The Variational AutoEncoder framework, composed of a Gaussian distribution sampler and an RNN encoder-decoder in the sequence-to-sequence scheme (Bowman et-al., 2015) has been commonly used for text or text-like generation. In (Gomez-Bombarelli et-al., 1988; Kusner et-al., 2017; Dai et-al., 2018; Guimaraes et-al., 2018; Kang and Cho, 2018), VAE is adapted to the SMILES-based molecular generation task by replacing the text input with the SMILES input and adding chemical validity checker that rejects SMILES output strings corresponding to invalid molecular graphs.

Learning the SMILES grammar implicitly through the the string representation is unstable: small changes may lead to completely different molecules or often invalid molecules. To address this issue, (Kusner et-al., 2017) and (Dai et-al., 2018) explicitly predefine a set of SMILES grammar rules to decompose SMILES-formatted molecules into a sequence of rules that constructs a context-free-grammar parse tree. The VAE is then trained to generate a sequence of grammar rules to construct a valid molecule. Alternative frameworks for the molecule discovery task include Reinforcement Learning (RL) (Tropsha et-al., 2018; Olivecrona, Blaschke, Engkvist, and Chen, 2017) and Generative Adversarial Network (GAN)

(Guimaraes et-al., 2018). Despite their potentially superior performance compared to VAE, these two framework do not fit in our context since their output can be random molecules. In this work, we expect VAE-based generative models to generate new OPV molecule candidates which are improvement (i.e., better chemical properties) from input OPV molecules. Additionally, optimizing RL- and GAN-based generative models for best molecules generally can be more expensive compared to VAEs for our proposed task.

2.1.2 Graph-based Molecular Generators

(Jin et-al., 2018, 2019, 2020; You et-al., 2019; Liu et-al., 2019; Kang and Cho, 2018) proposed graph-based molecular generative models to accept molecular graphs as direct input to learn the graphical structures of molecules and to avoid converting molecular graphs to other formats that may lead to the pipeline error (e.g. a minor change in SMILES yields the corresponding invalid molecule). In a molecular graph, the nodes are atoms; the edges are bonds between adjacent atoms, with three bond types: single, double, or triple. To apply machine learning to molecules, it is common to map atoms and edges to trainable continuous representations to learn the molecular graphs.

Like the SMILES-based molecular generation, (Li et-al., 2017; Liu et-al., 2019; Zhou et-al., 2019; Samanta et-al., 2020; You et-al., 2019; Jin et-al., 2018, 2019, 2020) proposed graph-based generative models, composed of VAE, for the molecule generation task. Their encoders utilize graph neural networks (GNN) (Hamilton et-al., 2017) or graph convolutional networks (GCN) (Kipf and Welling, 2016) to learn the molecules' graphical structures. Generative models in (Li et-al., 2017; Liu et-al., 2019; Samanta et-al., 2020; Zhou et-al., 2019) follow the deep auto-regressive decoding approach in (You et-al., 2018) that decodes molecules sequentially, node by node, conditioned on past nodes. During training, decoders usually follow the decoding ordering of nodes in depth-first-search where the starting node is the first atom in the canonical atom order. During inference, the decoding stops if it reaches

the maximum number of iterations allowed or predicts the ending node. The validity check can run on the partially generated subgraphs when new nodes are added, or on the complete graph.

In contrast, (Simonovsky and Komodakis, 2018; De Cao and Kipf, 2018; Ma, Chen, and Xiao, 2018) developed generative models which output the adjacency matrices and node labels of the graph at once.

Instead of VAE, (Zhou et-al., 2019) and (De Cao and Kipf, 2018) proposed to use RL and GAN to optimize molecule output.

Regardless the framework choice, the graph-based generative models can be subcategorized by the resolution of the molecule graph input: *atom-based* or *motif-based*. The formation of substructures is constrained by rings, chains, or flexible rules like in (Jin et-al., 2020) that can represent molecules at multiple resolutions (supporting both atoms and substructures).

Atom-based Methods

The atom-based generative model proposed by (Simonovsky and Komodakis, 2018) uses VAE with GNN to encode molecules. The decoder outputs a probabilistic fully-connected graph of a predefined maximum size at once. However, this at-once approach requires an additional step, called graph-matching, to train a simple fully-connected neural network to find nodes significantly similar to the original graph input. (De Cao and Kipf, 2018) follows the same at-once decoding approach but replaces VAE with RL. The at-once decoding is prone to generate invalid molecules. (Ma, Chen, and Xiao, 2018) proposed generation constraints to regularize the output probabilistic graph as a step toward semantic validity. This decoding method also limits size of the graph output.

To address the above issues, (Li et-al., 2017; Liu et-al., 2019) formulate the molecular generation task as auto-regressively yielding a sequence of structure building decisions, i.e.

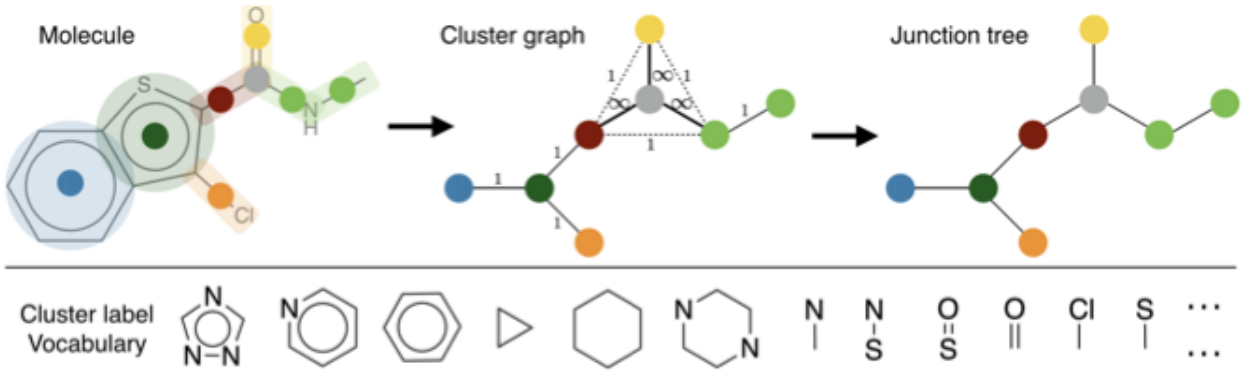


Figure 2.1: A molecular graph G is decomposed into its junction tree T_G, where each colored node in the tree represents a substructure in the molecule. Atoms, bonds, and their types are reserved within substructures. Every two adjacent substructures overlap by at least one atom. Cycles in purple and red are the simplest rings in the molecule. (Jin et-al., 2018)

(1) *add_node*, (2) *add_edge*, (3) and *pick_node* - pick a node connected to the new node. The generation of building decisions is repeated until the ending node is generated. At each step, the model uses a message passing network (MPN) (Hamilton et-al., 2017) to extract the visiting nodes' representations by aggregating from their neighboring nodes and edges. The *add_node* step decides whether to add another node or to terminate the algorithm. If adding another node, and the *add_edge* step decides whether or not to connect the new node with the best pre-existing node (which is determined at the *pick_node* step); otherwise, the algorithm sees the new node as the starting node of another molecule branch that will be attached to the existing branch at some other node.

Motif-based Methods

Atoms are the smallest building blocks and require a large number of decoding steps to generate a whole molecule. Hence, the atom-based methods are prone to make errors when generating large molecules. To reduce the number of decoding steps, (Jin et-al., 2018, 2019, 2020) proposed to utilize tree decomposition to decompose molecule graphs into subgraphs called *motifs*, such that overlapping motifs completely cover molecule graphs (See Figure-2.1

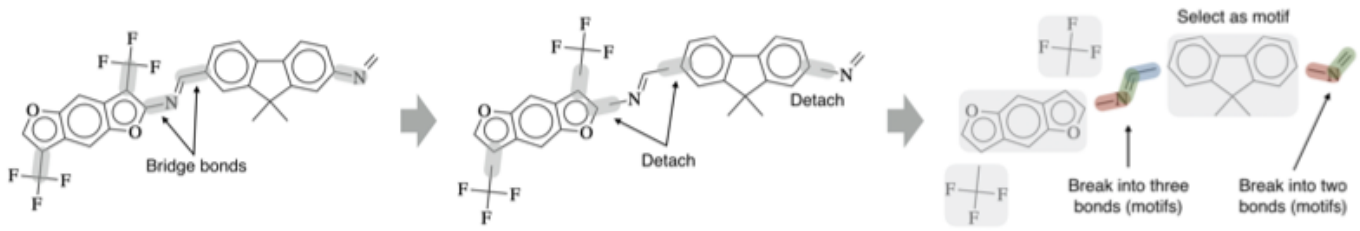


Figure 2.2: Illustration of the motif extraction: (1) identify bridge bonds that connect rings and non-ring subgraphs, (2) convert molecule into disconnected subgraphs by breaking bridge bonds, (3) select subgraphs with frequency higher than the predefined f threshold, (4) break remaining subgraphs into simpler rings and bonds. (Jin et-al., 2020)

for details). In the resulting trees, adjacent motifs are guaranteed to overlap each other by at least one common atom. The motif vocabulary extracted from the tree decomposition consists of not only individual atoms (e.g. carbon C, oxygen O, chloride Cl, etc.), as in atom-based methods, but also rings and bonds. The ring vocabulary is extracted by finding the smallest sets of smallest rings (SSSR) (Baumer, Sale, and Sello, 1991), e.g. hexagons in Figure-2.1.

A challenge of building the motif vocabulary is to identify large or complex enough motifs that occur with high frequency in order to reduce the decoding steps. Instead of constraining into simple rings by SSSR, (Jin et-al., 2020) further improved the motif extraction with a sequence of decomposing rules (See Figure-2.2 for details): (1) identify bridge bonds that connect rings and non-ring subgraphs, (2) convert molecular graphs into disconnected subgraphs by breaking bridge bonds, (3) select subgraphs with frequency higher than the predefined f frequency threshold, (4) break remaining subgraphs into simpler rings and bonds.

Since molecules are represented both by subgraphs and individual atoms, motif-based generative models have tree and graph encoders to respectively learn molecular graphs represented by junction trees. Similarly, there are tree and graph decoders to decode junction trees autoregressively (Jin et-al., 2018, 2019, 2020). Hence, the decoding process happens in two phases: tree decoding to predict next motifs and graph decoding to attach the new

motif with the existing tree. The tree decoding phases includes: (1) topological prediction to decide whether to add another node, (2) label prediction to determine the motif type of the new node. Given the new tree node, the graph encoding phase performs two steps: (1) sampling different combinations between the new node and its direct parent node, (2) scoring and selecting the valid combination with the highest score. The attachment happens only if there exists at least a valid combination between common atoms shared by the new node and the existing graph. If no valid combination, the algorithm discards the new node and backtracks to its ancestors.

2.1.3 Serializing Molecule Graphs

Since being formulated similarly as a text generation task, SMILES-based molecular generative models encode and decode SMILES strings in a fixed order (e.g., left to right) such that the first character in the SMILES string is the starting node of the molecular graph. However, the ordering of atoms or motifs in the graph-based representation is non-deterministic. Serializing molecular graphs is a non-trivial step since the ordering of nodes and edges is critical to the robustness of the molecular generation task. The starting node can be any atom or motif picked randomly. Different starting nodes lead to different node orderings. Given the starting node, the node ordering is usually determined by depth-first-traversal (DFT), as in (Jin et-al., 2018, 2019, 2020), or breath-first-traversal (BFT), as in (You et-al., 2018; Liu et-al., 2019). A fixed node ordering is applied for all molecular graphs for consistency in training.

Most graph serializing algorithms (e.g., DFT or BFT) performs the graph-to-tree conversion that could accidentally flatten rings or significant cycles (i.e., a self-closed chain of rings and atoms) describing molecules' properties. For example, the ring structure of benzene (C6H6) could be decomposed to an opened chain of atoms. As a result, the tree representation may loose a certain amount of the structural information of molecules. The fixed node

ordering is preferred for simplicity but not able to fully capture the molecule structural information. Experiments by (Li et-al., 2017) showed that the random ordering (random pick of the starting node) outperformed the uniform ordering by validity and novelty metrics.

2.2 Molecular Optimization

Molecular optimization is an extended-task of the molecular generation that map original molecules into new molecules to satisfy the desired properties. The common approach is to couple and jointly train a property predictor with the molecule generator, for example the VAE + property-predictor framework.

CHAPTER 3

APPROACH

In this thesis, we adopt a graph substructure-based generation approach. We apply the RDKit package (Landrum, 2016) to convert each molecule in a dataset, which are usually stored as SMILES strings, to the corresponding molecular graph $G = (V, E)$ with atoms V as nodes and bonds E as edges. First, we extract motif substructures from the training set of molecular graphs to build a motif vocabulary that is used to represent all graphs in both the training and inference stages. The graphs are then serialized into sequences of motifs in a fixed order, which determines the decoding steps. Given these serialized motif sequences as input, we pretrain our graph variational autoencoder for the molecule reconstruction task. Next, we fine-tune the molecule optimizer, composed of the pretrained molecule generator and two property regressors, in order to generate molecules constrained by their HOMO and LUMO properties. Finally, during inference, the molecule optimizer maps input molecules to new forms with HOMO and LUMO properties that are closer to the target values given by the Chemistry Department. The rest of this chapter describes each of these five steps in detail.

3.1 Motif Extractor

We apply the motif extraction algorithm proposed by (Jin et-al., 2020) over all molecules in the training set to extract motifs and to construct the motif vocabulary V_s. Each motif $S_i = (V_i, E_i)$ is a subgraph of a molecule $G(V, E)$. For each molecule, the algorithm decomposes G into a set of motifs $S1, ..., S_n$ such that their union completely covers the entire graph ($V = \bigcup_i V_i$ and $E = \bigcup_i E_i$). A motif does not have a deterministic list of intersection atoms used to connect to neighboring motifs. Upon different sets of neighboring motifs, a motif S_i has respectively distinct attachments $A_i = \{S_i, u \mid u \in \bigcup_k S_i \cap S_k\}$, each of which

denotes exactly a set of intersection atoms u between S_i and its motif neighbors S_k. By this definition, to connect two motifs, we have to decide two corresponding attachments that can be compatibly attached together via intersection atoms (e.g., two attachments must have available atoms of the same type). Figure-2.2 illustrates the following steps in the motif extraction algorithm:

1. For a given molecular graph $G = (V, E)$, find all the *bridge* bonds $(u, v) \in E$, where both u and v have vertex degree $\Delta_u, \Delta_v \geq 2$, and either u or v is a part of a ring (cycle).

2. Disconnect the bridge bonds to yield a set of disconnected subgraphs $G_1, ..., G_n$.

3. Add a subgraph G_i to the the motif vocabulary if it occurs at least $f = 100$ times in the training set (we use the same frequency threshold as Jin et al.).

4. Decompose the remaining subgraphs into simple rings (i.e. single cycles) and bonds and add them to the motif vocabulary.

Given the motif vocabulary $\mathbb{S}$, we can represent a molecular graph as $G = (S, A)$ with motifs $S = S_1, ..., S_n$ and attachments $A = A_1, ..., A_n$. For each motif $S_i \in \mathbb{S}$ extracted from the training set, we collect all its attachments to build the motif-specific attachment vocabulary $\mathbb{A}_{S_i}$. During inference, if the motif extraction algorithm yields unknown motifs or attachments, the molecule is rejected. Additionally, we build an atom vocabulary $\mathbb{V}$ from the periodic table and a bond-type vocabulary $\mathbb{E} = \{\text{single}, \text{double}, \text{triple}\}$ that are independent from the training set.

3.2 Graph Serialization

We use the RDKit package (Landrum, 2016) to convert a molecule into its atom-level molecular graph $G = (V, E)$ in which the root atom $v_0 \in V$ is the first atom in the canonical

atom order of the molecule. After converting to a motif graph $G = (S, A)$, we can select the root motif $S_0 = (V_0, E_0)$ such that $v_0 \in V_0$. From the root motif, we perform Depth First Traversal to serialize the motif graph into a sequence of motifs $S_0, ..., S_n$, which serves as the input to the molecule encoder and also represents the fixed decoding order for generation.

3.3 Pre-Training: Molecule Generator

The task of generating graphs is challenging, especially graphs like the organic photovoltaic (OPV) molecules in this thesis. We are limited from utilizing existing pretrained generative models due to the significant difference in structures among molecule types. Compared to general organic molecules (e.g. the QM9 dataset (Ramakrishnan, Dral, Rupp, and Lilienfield, 2014; Ruddigkeit et al., 2012)), organic photovoltaic molecules have a much larger number of atoms (see Table 4.3). Additionally, each molecule type may have unique structural patterns. Therefore, to fit the OPV domain, we pretrain our molecule generator on the molecule reconstruction objective using a custom dataset of only OPV molecules (see Section 4.1 for dataset details).

3.3.1 Hierarchical Graph-to-Graph (HierG2G) Model

To reduce the number of decoding steps required by the large number of atoms in OPV molecules, we adopt the Hierarchical Graph-to-Graph (HierG2G) generative model proposed by (Jin et-al., 2020), composed of a Variational AutoEncoder (Kingma and Welling, 2013) with a hierarchical graph encoder and decoder, to generate molecular graphs motif by motif.

Hierarchical Graph Encoder

The graph encoder extracts essential information from the hierarchical graph H_G of a molecule G at three layers (see Figure 3.1):

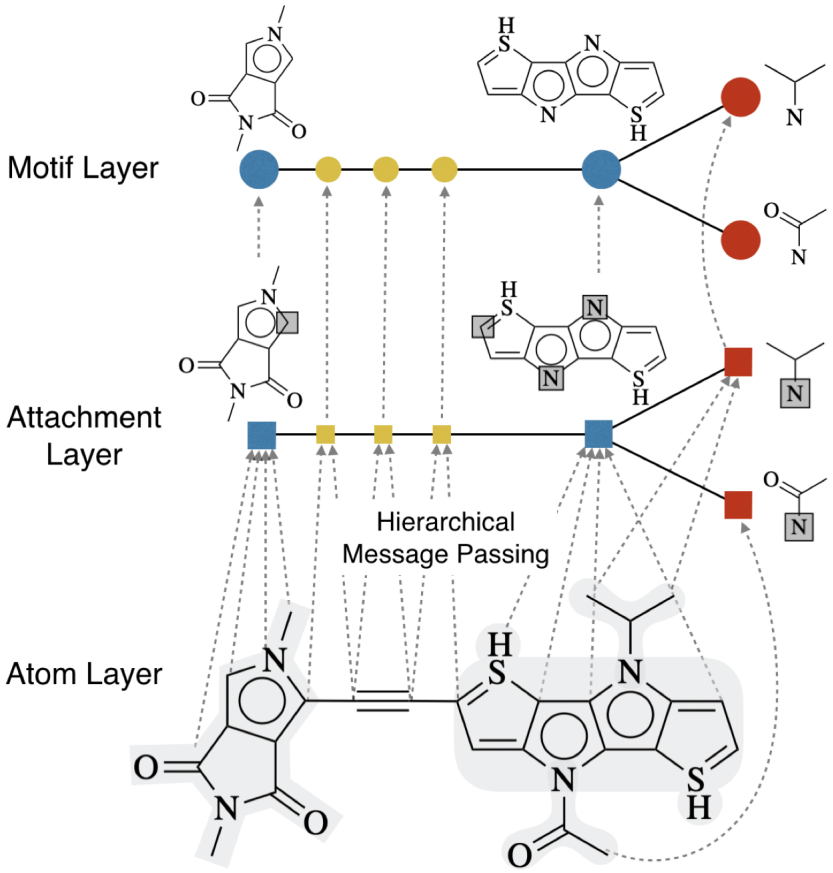


Figure 3.1: Hierarchical graph encoder layers. Dashed arrows connect each atom to the motifs it belongs to. In the attachment layer, each node A_i is a particular attachment configuration of motif S_i; the atoms in the intersection between each motif and its neighbors are highlighted in the grey boxes.

1. **Motif Layer.** H_G^S describes the coarse connetivity information among motifs and is crucial information for *motif prediction* when decoding. At this layer, the motif connectivity is a fixed tree structure due to the graph serializing step in Section 3.2. The edge between two motif nodes S_i, S_j has weight $d_{ij} = k$ if S_i is the kth child of S_j in the serialized graph, or $d_{ij} = 0$ if S_i is the parent of S_j. Note that the fixed tree structure may not fully describe the connectivity of the molecule G, since the graph serializing algorithm linearizes any cycles.

2. **Attachment Layer.** H_G^A contains attachment nodes $A_i = \{S_i, \{u\}\}$, which capture the connectivity of a motif S_i and one of its neighbors via an intersection atom u; the attachment edges are weighted in the same way as the motif edges. This fine-grained connectivity information is significant to the *attachment prediction* step of the graph decoder.

3. **Atom Layer.** H_G^G is the original molecular graph $G = (V, E)$. The atom connectivity at this finest layer is necessary for the *atom prediction* step of the graph decoder.

In a hierarchical graph H_G, a direct edge connects atoms v in the atom layer to the attachments A_i in the attachment layer that contain them; a direct edge also connects A_i to its motif S_i in the motif layer. The encoder uses three Message Passing Networks (MPNs, see Appendix B) to encode each of the three layers in the hierarchical graph. In the equations below, we denote each MPN module as $\text{MPN}(\cdot)$; we use $\text{MLP}(\mathbf{x}, \mathbf{y})$ to denote a multi-layer perceptron network given the concatenation of $\mathbf{x}$ and $\mathbf{y}$ as input.

Atom Layer MPN. The input to the atom layer MPN is the atom and bond embedding vectors $\{e(u)\}, \{e(u, v)\}$ of all atoms and bonds in H_G^G. MPN propagates the message vectors between an atom u and all of its atom neighbors for T_G iterations. The output atom representation is h_u per atom u:

$$\{h_u\} = \text{MPN}_G \left(H_G^G,\ \{e(u)\},\ \{e(u, v)\} \right) \tag{3.1}$$

Attachment Layer MPN. For each attachment node A_i, its attachment embedding $e(A_i)$ and the sum of its atom representations $\{h_u \mid u \in S_i\}$ from the previous layer are fused altogether via MLP into the attachment node feature vector:

$$f_{A_i} = \text{MLP} \left(e(A_i), \sum_{u \in S_i} h_u \right) \tag{3.2}$$

The input to the attachment layer MPN consists of the attachment node features $\{f_{S_i}\}$ and embedding vectors of edges $\{e(d_{ij})\}$. Then, we run message passing for T_A iterations to obtain the attachment representations:

$$\{h_{A_i}\} = \text{MPN}_A\left(H_G^A,\ \{f_{A_i}\},\ \{e(d_{ij})\}\right) \tag{3.3}$$

Motif Layer MPN. The motif node feature vector f_{S_i} is the concatenation of the motif embedding vector $e(S_i)$ and its corresponding attachment representation h_{A_i} from the previous layer:

$$f_{S_i} = \text{MLP}\left(e(S_i),\ h_{A_i}\right) \tag{3.4}$$

The MPN runs for T_S iterations to obtain the motif representations:

$$\{h_{S_i}\} = \text{MPN}_S\Big(H_G^S,\ \{f_{S_i}\},\ \{e(d_{ij})\}\Big) \tag{3.5}$$

Finally, we compute the latent vector z_G sampled via the reparameterization trick with mean $\mu(h_{S_0})$ and log variance $\Sigma(h_{S_0})$, where S_0 is the first motif to be generated during the reconstruction (the root motif):

$$z_G = \mu(h_{S_0}) + \exp(\sum(h_{S_0})) \cdot \epsilon;\ \epsilon \sim N(0, I) \tag{3.6}$$

Hierarchical Graph Decoder

The graph decoder performs auto-regressive decoding to generate the next motif given the previously generated motifs. In each generation step, the decoder first utilizes the graph encoder to hierarchically encode the current partially generated graph $H_G^{(t-1)} = S_0, S_1, ..., S_{t-1}$ to obtain the partial graph features $h_G^{(t-1)}$. Conditioned on this encoding, the decoder then predicts whether or not to generate a new motif S_t (*topology prediction*), and if so, predicts

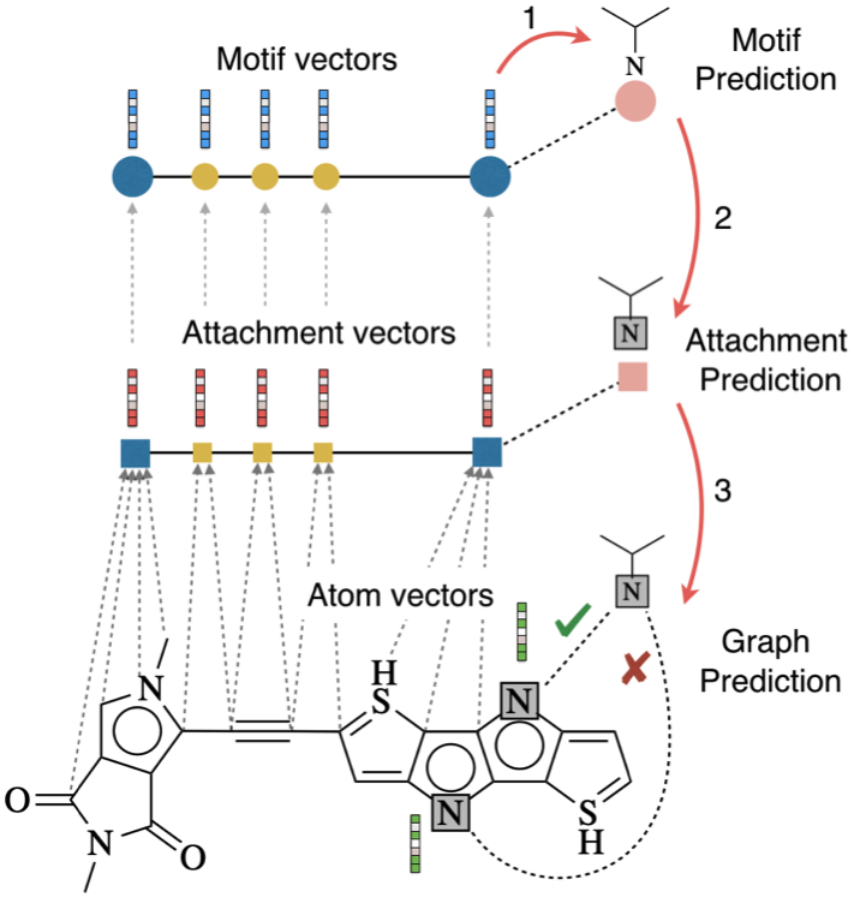


Figure 3.2: Hierarchical graph decoder. In each step, the decoder first runs hierarchical message passing to compute motif, attachment and atom vectors. Then it performs motif and attachment prediction for the next motif node. Finally, it decides how the new motif should be attached to the current graph via graph prediction.

the motif type of S_t (*motif prediction*). Next, it decides which attachment A_t of the new motif S_t will be used to connect with the partial graph (*attachment prediction*). Finally, it looks for the best atom pair (u, v), where $u \in A_k$ and $v \in A_t$, to connect the two motifs S_k, S_t, where $S_k \in H_G^{(t-1)}$ (*graph completion*). These steps are dependent on each other such that a change at any step would lead to a different result. The hierarchical generation process eventually outputs a complete molecule at the atom level $G = (V, E)$.

During decoding, a stack $\mathcal{F}$ maintains a list of unique frontier attachments $\{A_f \mid A_f \in \mathcal{F}\}$, motifs that have at least one neighbor not yet generated. The stack $\mathcal{F}$ stores motifs in their serialized, depth-first-search order (Section 3.2). Given the latent representation z_G and the attachment A_f at the top of stack $\mathcal{F}$ at step t, the generation process is as following:

1. **Topology Prediction.** The decoder predicts whether or not to add a new motif.

$$y_{Topo_t} = \sigma\Big((MLP)(h_{S_f}, z_G)\Big) \tag{3.7}$$

   If not, A_f is popped off the stack and this step is repeated with the next frontier attachment A'_f.

2. **Motif Prediction.** The decoder predicts the next motif S_t to attach to A_f. The prediction is cast as a classification over the motif vocabulary $\mathbb{S}$.

$$y_{S_t} = \text{softmax}\Big(\text{MLP}(h_{S_f}, z_G)\Big) \tag{3.8}$$

3. **Attachment Prediction.** The decoder predicts which attachment A_t of motif S_t will be used to connect to A_f (i.e., decide intersecting atoms in A_t to connect to A_f). Note that no two attachments of S_t contain the same set of atoms (i.e., two attachments must differ by at least one atom type or index); and not all intersecting atoms in A_t will be used to connect to A_f. The prediction is cast as a classification over the motif-specific attachment vocabulary $\mathbb{A}_{S_t}$.

$$y_{A_t} = \text{softmax}\Big(\text{MLP}(h_{S_f}, z_G)\Big) \tag{3.9}$$

4. **Graph Completion.** Given A_t and A_f, their Cartesian product of intersecting atoms forms pairs $\big(u, v\big)$ where atoms $u \in A_f$ and $v \in A_t$ correspond to a single atom in the overall molecule graph that is shared between the two attachments. During inference,

the candidate attachment check (see Appendix C) selects the first chemically valid atom pair among the k highest-scoring pair candidates to attach A_t and A_f.

$$y_{C_t} = \text{softmax}\Big(\text{MLP}(h_M, z_G)\Big) \tag{3.10}$$

$$h_{C_t} = \sum_j \text{MLP}(h_u, h_v) \tag{3.11}$$

If a valid attachment is found, the motif S_t is added to the partial hierarchical graph $H_G^{(t)}$. If no valid attachments are found, the candidate motif S_t is rejected, and the frontier attachment A_f is popped from the stack $\mathcal{F}$.

During training, the decision to add a motif S_t or to pop a motif S_f from the stack is guided by the depth-first-search generation order provided by graph serialization (Section 3.2). During inference, a newly generated motif S_t is added to the stack if its attachment A_{S_t} has at least one available attaching atom. The top-of-stack motif S_k is removed if its attachment A_{S_k} has all atoms attached neighbors. For each input molecule, the generation process repeats until reaching the predefined max decoding step threshold or no more frontier attachments in $\mathcal{F}$.

Pretraining

We apply teacher forcing (Sutskever, Vinyals, and Le, 2014; Werbos, 1990) to pretrain the molecule generator for the unsupervised graph reconstruction task. At every step t, a start-of-graph motif S_0 (when $t = 0$) or a sequence of gold motifs $S_0, S_1, ...S_{t-1}$ is provided as input to the molecule decoder to generate S_t. Given a training set of molecules, the training objective is to minimize the negative Evidence Lower Bound (ELBO) (Kingma and Welling, 2013):

$$
\begin{aligned}
L_{Gen} &= \lambda_{KL} D_{KL}[Q(z_G|G) \mid\mid P(z_G)] + -\mathbb{E}_{z\sim Q}[logP(G|z_G)] \\
&= \lambda_{KL} D_{KL}[Q(z_G|G) \mid\mid P(z_G)] + L_{Rec}
\end{aligned}
\tag{3.12}
$$

The first right-hand-side (RHS) term is the Kullback-Leibler (KL) divergence to approximate the decoder posterior $Q(z_G|G)$ close to the encoder prior $P(z_G)$. Given the latent representation z_G derived from the graph encoder (Equation 3.6), the second RHS term is the cross-entropy reconstruction loss of the molecule graph G. This measure describes the performance of the graph decoder in reconstructing the molecule.

For all generation time steps T, The graph reconstruction loss L_{Rec} can be rewritten as the sum of the cross entropy losses of the four hierarchical generation steps in the graph decoder: the topology prediction loss L_{Topo}, the motif prediction loss L_S, the attachment prediction loss L_A between the new motif and the existing subgraph, and the graph completion loss L_{Graph}.

$$
L_{Rec} = L_{Topo} + L_S + L_A + L_G \tag{3.13}
$$

$$
L_{Topo} = -\frac{1}{T} \sum_{t \in T} \tilde{y}_{Topo,t} \log(y_{Topo,t}) + (1 - \tilde{y}_{Topo,t}) \log(1 - y_{Topo,t}) \tag{3.14}
$$

$$
L_S = -\frac{1}{T} \sum_{t \in T} \sum_{i \in V_{S,t}} \tilde{y}_{S_i,t} \log(y_{S_i,t}) \tag{3.15}
$$

$$
L_A = -\frac{1}{T} \sum_{i \in T} \sum_{i \in V_{A_S}} \tilde{y}_{A_i,t} \log(y_{A_i,t}) \tag{3.16}
$$

$$
L_C = -\frac{1}{T} \sum_{t \in T} \sum_{i \in V_{Atom}} \tilde{y}_{C_i,t} \log(y_{C_i,t}) \tag{3.17}
$$

where $\tilde{y}$ is the classification ground truth and y is the classification output.

3.3.2 Motif-level Graph-to-Graph (MotifG2G)

The state-of-the-art performance of motif-based generative models in generating large molecules (Jin et-al., 2019, 2020) has shown that the atom representation negatively affected the generation performance (i.e. degrades the reconstruction accuracy and restricts the output length), especially towards large molecules. Hence, we argue that the atomic level in the graph space could not fully capture the structural information corresponding to properties. In other words, properties could be majorly attributed by the motif-based structures (Jia et al., 2022; Zhang et al., 2022). Hence, we propose MotifG2G, a modification of HierG2G, to minimize the intake of the atom information. HierG2G and MotifG2G shares the same pretraining procedure described in Section 3.3.1.

Motif Graph Encoder To avoid encoding atoms, the MotifG2G graph encoder uses only the attachment and motif layers in the hierarchical graph H_G. Due to the removal of the atom layer, input to the attachment layer MPN includes only the embedding vectors of the attachments $\{e(A_i)$, and not the feature vector $f(A_i)$ derived from the atom layer representations used in HierG2G. The attachment representation is computed as following:

$$\{h_{A_i}\} = \text{MPN}_a \left(H_G^A,\ \{e(A_i)\},\ \{e(d_{ij})\} \right) \tag{3.18}$$

Motif Graph Decoder Since the final output format must be at the atom level, MotifG2G and HierG2G share the same graph decoder, except for which encoder is used to encode the partial graph: in MotifG2G, the partial graph encoder does not use the atom layer.

Pretraining We pretrain MotifG2G in the same way as previously described for HierG2G.

3.4 Fine-Tuning: Property Regressors

In this thesis, molecular optimization is seen as mapping input organic photovoltaic molecules to new forms with preferable HOMO and LUMO properties. Hence, we formulate the molecular optimization task as jointly learning two: generating molecular graphs and approximating HOMO and LUMO properties. Our molecule optimizer is composed of the pretrained molecule generator described in the previous section and two multilayer perceptron property regressors trained to predict the HOMO and LUMO scores of the input molecule G given the latent representation z_G produced by the graph encoder.

While the pretrained molecule generator is fine-tuned for the reconstruction task, the regressors are trained to minimize the Mean Squared Errors (MSE) between the ground truth property values $\tilde{y}$ and the output property values y:

$$L_{HOMO} = \frac{1}{T} \sum_{t \in T} (y_{t,HOMO} - \tilde{y}_{t,HOMO})^2 \tag{3.19}$$

$$L_{LUMO} = \frac{1}{T} \sum_{t \in T} (y_{t,LUMO} - \tilde{y}_{i,LLUMO})^2 \tag{3.20}$$

The joint loss (Equation 3.21) is the sum of the molecule generation loss (Equation 3.12) and the two regression losses.

$$L = L_{Gen} + L_{HOMO} + L_{LUMO} \tag{3.21}$$

The uniform weights among loss terms in Equation 3.21 can lead to unbalanced convergence rates between the graph generation and property regression tasks. On the one hand, property regression is a much easier task than graph generation; on the other, the graph generator is pretrained, while the property regressors are training from scratch. To address this issue, we employ two methods: (1) task-dependent loss scaling and (2) individually optimizing subnetworks.

3.4.1 Task-Dependent Loss Scaling

To adjust convergence rates, we couple each loss term with a weight λ_i such that $\sum_{i=1}^{n} \lambda_i = 1$, where n is the number of loss terms. Each weight λ_i may enlarge or cut a term's contribution to the final loss. Hence, the final loss is the weighted sum of the individual loss terms:

$$L = \sum_{i}^{n} \lambda_i L_i \tag{3.22}$$

To avoid an exhaustive search for optimal loss weightings, we adopt the task-dependent weighting approach proposed by (Kendall et al., 2015) that frames the regression or classification tasks as maximizing the Gaussian likelihood with mean μ and variance σ. Hence, the approach couples each loss term with a trainable variance parameter σ to introduce more noise to compensate the uncertainty.

We rewrite the MSE losses in Equations 3.19 and 3.20 for the HOMO and LUMO approximation tasks in terms of the noise parameter σ:

$$L^{'}_{HOMO} = \frac{1}{2\sigma^2} L_{HOMO} + log\sigma \tag{3.23}$$

$$L^{'}_{LUMO} = \frac{1}{2\sigma^2} L_{LUMO} + log\sigma \tag{3.24}$$

Similarly, we rewrite the reconstruction loss in Equation 3.13 in terms of the noise parameter σ:

$$L^{'}_{Gen} = \frac{1}{2\sigma^2_{Gen}} L_{Gen} + log\sigma_{Gen} \tag{3.25}$$

In each loss term, the increase in the variance σ scales down its respective loss. On the other hands, when the variance σ decreases, the loss increases. The log variance $log\sigma$ acts

as penalty to discourage the variance σ from increasing too large. Following (Kendall et al., 2015), in implementation, we substitute $log\sigma$ with $s := log\sigma^2$ where its exponential mapping to $\frac{1}{2\sigma^2}$ is $\exp(-s)$ for numerical stability. This technique avoids division by zero in the first term in each scaled loss equation. The scaled final loss equation is

$$L^{'} = L^{'}_{Gen} + L^{'}_{HOMO} + L^{'}_{LUMO} \tag{3.26}$$

Note that L' could be negative when each loss term reaches sub-optimum. Then, the model may stop improving. Hence, we clip the negative total loss to zero and add a small amount of random noise if the total loss is negative.

3.4.2 Individual Optimization

In addition to tuning the loss contribution weights, we attempt to tune the learning rate, an important hyper-parameter, for each task. We use four instances of learning rate optimization algorithms, each with its own initial learning rate, to individually optimize the four subnetworks of our model: the graph encoder, the graph decoder, the HOMO regressor, and the LUMO regressor. Each optimizer controls the convergence rate of the corresponding subnetwork. In our implementation, all four subnetworks use the different instances of the same optimization algorithm, with different initial learning learning rates. Since the graph generation task is more difficult than the property regression task, we expect the HOMO and LUMO regressors to have smaller learning rates than those of the graph encoder and decoder.

3.5 Inference: Property-Guided Optimization

We perform property-guided molecule generation using a three-stage inference process. In the first stage, the graph encoder encodes the graph G of an input molecule and yields the latent vector z_G of length n. In the second stage, z_G is split into two halves $z_{HOMO} = [z_0, z_2, ..., z_{\frac{n}{2}-1}]$

and $z_{LUMO} = [z_{n/2}, ..., z_{n-1}, z_n]$, which are the inputs to the HOMO and LUMO property regressors. We use the two regressors to search for two new latent vectors z'_{HOMO} and z'_{LUMO}, which we discuss in the rest of the chapter. In the final stage, the graph decoder takes as input the concatenation of z'_{HOMO} and z'_{LUMO} to generate a new molecule G' with the desired HOMO/LUMO values. The size of G' is limited by a predefined maximum number of decoding steps (i.e. the number of motifs) and atoms.

To find latent vectors $z^{'}_{HOMO}$ and $z^{'}_{LUMO}$ that produce G' with more desirable HOMO/LUMO values, we repeatedly apply gradient ascent or descent. The optimization process starts by predicting the property values y_{HOMO}, y_{LUMO} of the latent vector z_G and calculating the gradients of the regressors ∇_{HOMO} and ∇_{LUMO}. Using HOMO as an example, we then apply gradient descent to update its input z_{HOMO} if the predicted y_{HOMO} is larger than the desired value $\tilde{y}_{HOMO}$, or gradient ascent if it is smaller (see Equation 3.27). This process is repeated to search for the best possible z'_{HOMO}, and the process for z'_{LUMO} is exactly the same.

$$
\begin{aligned}
z^{(t+1)}_{HOMO} &= z^{(t)}_{HOMO} - switch * \eta * \nabla^{t}_{HOMO} \\
switch &= \begin{cases} 1 & \text{if } y'_{HOMO} < y_{HOMO} \\ -1 & \text{if } y'_{HOMO} \geq y_{HOMO} \end{cases}
\end{aligned} \tag{3.27}
$$

where η is the learning rate.

To control the termination of this iterative optimization process, we experiment with three control methods: (1) hard control, optimizing over a T number of iterations; (2) patience, optimizing until improvement plateaus; and (3) soft control, optimizing until reaching the minimum loss gap. All methods are designed to avoid overfitting and wasting computational resources.

3.5.1 Hard Control

This is a straightforward method that terminates the search after T iterations.

Algorithm 1 Hard Control

Model: Two property regressors, MLP_{HOMO} and MLP_{LUMO}, in the fine-tuned molecule optimizer described in Section 3.4.

Input: HOMO/LUMO latent vectors (z_{HOMO} and z_{LUMO}), learning rate η, and gold HOMO/LUMO value ($\tilde{y}_{HOMO}$ and $\tilde{y}_{LUMO}$).

Output: In-place optimized HOMO and LUMO latent vectors, z_{HOMO} and z_{LUMO}.

for $t < T$ **do**

$y_{HOMO},\ MSE_{HOMO} \leftarrow \text{MLP}_{HOMO}(z_{HOMO})$

$y_{LUMO},\ MSE_{LUMO} \leftarrow \text{MLP}_{LUMO}(z_{LUMO})$

$MSE \leftarrow MSE_{HOMO} + MSE_{LUMO}$

∇_{HOMO} and $\nabla_{LUMO} \leftarrow$ EXTRACT_GRADIENTS(MSE);

$z_{HOMO} \leftarrow$ LATENT_VECTOR_UPDATE($z_{HOMO}, \eta, \nabla_{HOMO}, \tilde{y}_{HOMO}, y_{HOMO}$)

$z_{LUMO} \leftarrow$ LATENT_VECTOR_UPDATE($z_{LUMO}, \eta, \nabla_{LUMO}, \tilde{y}_{LUMO}, y_{LUMO}$)

end for

3.5.2 Patience Control

This method terminates the search when the predicted HOMO/LUMO of the latent vector $z^{(t)}$ improves by less than a predefined threshold λ for N iterations in a row. We define improvement as the ratio of the total MSE regression loss to the target HOMO/LUMO value between the current time step t and the previous time step $t-1$. If there always exists the significant improvement, we stop optimizing after a maximum T iterations.

3.5.3 Soft Control

This method is identical to patience control, except that we also terminate the search when the total MSE regression loss at step t is below a predefined threshold δ.

Algorithm 2 Patience Control

Model: Two property regressors, MLP_{HOMO} and MLP_{LUMO}, in the fine-tuned molecule optimizer described in Section 3.4.

Input: HOMO/LUMO latent vectors (z_{HOMO} and z_{LUMO}), learning rate η, gold HOMO/LUMO value ($\tilde{y}_{HOMO}$ and $\tilde{y}_{LUMO}$), the improvement threshold λ, and N patience times.

Output: In-place optimized HOMO and LUMO latent vectors, z_{HOMO} and z_{LUMO}.

PREV_LOSS $\leftarrow 0$
n $\leftarrow N$
while $n > 0$ **do**
  $y_{HOMO},\ MSE_{HOMO} \leftarrow \text{MLP}_{HOMO}(z_{HOMO})$
  $y_{LUMO},\ MSE_{LUMO} \leftarrow \text{MLP}_{LUMO}(z_{LUMO})$
  $MSE \leftarrow MSE_{HOMO} + MSE_{LUMO}$

  if $\frac{\|MSE - \text{PREV\_LOSS}\|}{\text{PREV\_LOSS}} \leq \lambda$ **then**
    $n \leftarrow n - 1$
  else
    $n \leftarrow N$
  end if
  PREV_LOSS $\leftarrow MSE$

  ∇_{HOMO} and $\nabla_{LUMO} \leftarrow$ EXTRACT_GRADIENTS(MSE)

  $z_{HOMO} \leftarrow$ LATENT_VECTOR_UPDATE($z_{HOMO}, \eta, \nabla_{HOMO}, \tilde{y}_{HOMO}, y_{HOMO}$)
  $z_{LUMO} \leftarrow$ LATENT_VECTOR_UPDATE($z_{LUMO}, \eta, \nabla_{LUMO}, \tilde{y}_{LUMO}, y_{LUMO}$)
end while

Algorithm 3 Soft Control

Model: Two property regressors, MLP_{HOMO} and MLP_{LUMO}, in the fine-tuned molecule optimizer described in Section 3.4.

Input: HOMO/LUMO latent vectors (z_{HOMO} and z_{LUMO}), learning rate η, gold HOMO/LUMO value ($\tilde{y}_{HOMO}$ and $\tilde{y}_{LUMO}$), the improvement threshold λ, N patience times, and the loss threshold δ.

Output: In-place optimized HOMO and LUMO latent vectors, z_{HOMO} and z_{LUMO}.

```
PREV_LOSS ← 0
n ← N
while n > 0 do
  y_HOMO, MSE_HOMO ← MLP_HOMO(z_HOMO)
  y_LUMO, MSE_LUMO ← MLP_LUMO(z_LUMO)
  MSE ← MSE_HOMO + MSE_LUMO

  if MSE ≤ δ then
    Terminate the process.
  end if

  if ‖MSE − PREV_LOSS‖ / PREV_LOSS ≤ λ then
    n ← n − 1
  else
    n ← N
  end if
  PREV_LOSS ← MSE

  ∇_HOMO and ∇_LUMO ← EXTRACT_GRADIENTS(MSE)

  z_HOMO ← LATENT_VECTOR_UPDATE(z_HOMO, η, ∇_HOMO, ỹ_HOMO, y_HOMO)
  z_LUMO ← LATENT_VECTOR_UPDATE(z_LUMO, η, ∇_LUMO, ỹ_LUMO, y_LUMO)
end while
```

CHAPTER 4

EXPERIMENTS

In this chapter, we describe datasets used in this thesis and how we preprocess them. Then, we provide our choice of hyperparameters for our approach and illustrate training details of each experiment. Finally, we describe our selection of statistical and human evaluation metrics for the molecule optimization task, especially for OPVs.

4.1 Datasets

Due to the limited availability of OPV datasets, we attempt to combine OPV datasets with datasets of molecules that share some chemical properties with OPVs. Hence, we merge altogether three datasets: QM9, containing organic molecules; HOPV15, containing OPV molecules; and Curated-OPV, containing human-curated, large OPV molecules from the Chemistry Department. The goal is to leverage the "organic" feature similarity and the large number of molecules in QM9 to encourage our generative models learn a robust representation of generic organic molecule structure, and to use HOPV15 to provide some additional, smaller OPV molecules, along with Curated-OPV. The description of each of three datasets is as follows:

- **QM9** (Ramakrishnan, Dral, Rupp, and Lilienfield, 2014; Ruddigkeit et al., 2012) contains 133,845 organic molecules in the SMILES format, as well as their various chemical properties. We extract triplets of SMILES strings and HOMO/LUMO values to form our custom QM9 dataset. Despite containing mostly small molecules, the large size of this dataset is suitable for the unsupervised molecule reconstruction task.

- **HOPV15**, or Harvard Organic Photovoltaic Dataset (Aspuru-Guzik, 2015) contains 350 triplets of SMILES strings and HOMO/LUMO properties.

- **Curated-OPV** is provided by the Chemistry Department and contains 96 SMILES strings of OPVs and their HOMO/LUMO values. Some of these molecules are variants of each other, so after removing such duplicates, Curated-OPV contains only 46 triplets of SMILES, HOMO, and LUMO.

Data Preparation For each dataset, we first remove any chemically-invalid SMILES strings (verified by RDKit (Landrum, 2016))[1]. We shuffle the three datasets and partition them by the ratios shown in Table 4.1 into training, validation, test sets:

- **Training Set** combines data from QM9, HOPV15, and Curated-OPV. In addition to pretraining and fine-tuning on this data, we also build the motif vocabulary from the training set. The statistics of the motif vocabulary are available in Table 4.2.
- **Validation Set** is added to the training set during fine-tuning only.
- **Test Sets** All three test sets of QM9, HOPV15, and Curated-OPV are kept separately due to its significantly statistical dfference in molecular graphs (i.e., molecule size) in Table 4.3. Each test set is used to validate the molecule reconstruction and molecule optimization tasks. The test of the Curated-OPV is specifically used to determine the best approach to generate OPVs.

The statistics of molecule graphs and their HOMO/LUMO properties in the three datasets is available in Tables 4.3 and 4.4. Compared to HOPV15 and Curated-OPV, QM9 has a substantially larger number of molecules, but its average molecule size is significantly smaller. This observation suggests that our pretrained generative model might output smaller or invalid OPV candidates. Additionally, the range of HOMO and LUMO scores differs significantly among the three datasets, which might result in our property regressors being biased

[1]We found fewer than 20 invalid SMILES strings in the QM9 ans HOPV15 datasets

Table 4.1: Split Ratios for Training, Validation, and Testing subsets in QM9, HOPV15, and Curated-OPV datasets.

Dataset	**Training Ratio**	**Validation Ratio**	**Test Ratio**
QM9	0.9	0.05	0.05
HOPV15	0.7	0.15	0.15
Curated-OPV	0.5	0.25	0.25

Table 4.2: Statistics of the motif vocabulary built from the training set 4.1. The motif size is defined as the number of atoms per motif. Each motif has multiple possible attachments that each has a set of intersection atoms to connect with neighbor motifs (see Section 3.1). μ: Mean; σ: Standard Deviation

Number of Motifs	$\mu \pm \sigma$ **of Motif Size**	$\mu \pm \sigma$ **of Possible Attachments**	$\mu \pm \sigma$ **of Intersection Atoms**
6125	6.17 ± 1.78	8.62 ± 13.48	2.35 ± 1.34

towards the numerical range of the QM9 and HOPV15 molecules, which are similar to each other and much more numerous than Curated-OPV. A possible solution is data resampling (e.g. undersampling from QM9) to balance the distribution of the property value range, but in our experiments we did not find it necessary.

4.2 Hyperparameter Settings

In this section, we describe the experimental hyperparameter settings for our graph-to-graph molecule generator; the molecule optimizer, composed of a pretrained molecule generator and two property regressors; and the three control methods for stopping the search for the optimized latent vector z' during inference. We denote the number of layers of each message-passing network (MPN) by L, the number of iterations of each MPN by T, the hidden size of any neural network by H, the embedding size by H_{embed}, and the size of latent vector by H_{latent}.

Table 4.3: Statistics of molecule graphs in the three datasets. The triplets are non-empty and unique SMILES strings paired with HOMO and LUMO values. Per-molecule statistics are given as mean ± standard deviation.

Dataset	**Number of Molecules**	**Average Atoms per Molecule**	**Average Rings per Molecule**	**Average Ring Size**	**Average Bonds per Molecule**	**Size of Atom Vocabulary**
QM9	134K	8.79 ± 0.5	9.4 ± 1.17	1.74 ± 1.2	4.25 ± 1.01	4
HOPV15	350	42.78 ± 13.81	49.3 ± 16.1	13.17 ± 5.91	5.5 ± 0.49	7
Curated-OPV	46	98.73 ± 46.63	110.9 ± 52.26	7.52 ± 2.59	5.37 ± 0.48	8

Table 4.4: Mean (μ) and standard deviation (σ) of gold HOMO and LUMO values in the three datasets.

Dataset	$\mu \pm \sigma$ **of HOMO**	$\mu \pm \sigma$ **of LUMO**
QM9	-0.24 ± 0.02	0.01 ± 0.05
HOPV15	-0.19 ± 0.01	-0.9 ± 0.01
Curated-OPV	-5.5 ± 0.39	-3.87 ± 0.46

For all experiments, we set the dropout ratio $p = 0.1$ and the batch size $b = 20$. All pretraining and fine-tuning experiments are run for 20 epochs with initial learning rate $\eta = 1e^{-3}$; after each epoch, the learning rate is exponentially decayed at a rate of 0.9.

4.2.1 Molecule Generator

HierG2G

We use embedding size and hidden size $H_{embed} = H = 250$. In the encoder, the Atom Layer MPN has $L_G = 20, T_G = 5$; the Attachment and Motif Layer MPNs both have $L_A = L_S = 20, T_A = T_S = 1$. (Details of the MPN architecture and activation functions are in Appendix B.) The latent size $H_{latent} = 24$. The decoder shares the same architecture and hyperparameter settings with the encoder. Additionally, the encoder and decoder share atom, attachment, and motif embeddings (so called tie-embeddings) to reduce the overall model size (number of parameters) without sacrificing performance (Press and Wolf, 2017).

MotifG2G and MotifG2G$_{Large}$

The base motif-only model reuses the architecture and hyperparameters of HierG2G, except for the absence of the Atom Layer MPN in the encoder and decoder. To capture the large structures of OPVs, we also train MotifG2G$_{Large}$ with a larger hidden size $H = 700$ and

latent size $H_{latent} = 48$. Due to the larger number of parameters in HierG2G, we could not similarly increase its size without running into memory limitations.

Pre-Training

When pretraining the HierG2G and MotifG2G models, we use gradient clipping (Pascanu, Mikolov, and Bengio, 2013) with threshold $\nabla_{clip} = 20$. Gradient clipping technique is typically useful in the first few epochs of the pretraining stage, where the model tends to make large errors and yield large loss gradients and update steps during backpropagation. These "exploding" gradients could delay or prevent model convergence. Hence, clipping over-large gradient vlaues helps accelerate convergence.

4.2.2 Molecule Optimizer

The molecule optimizer (HierOptimizer, MotifOptimizer, MotifOptimizer$_{Large}$) respectively pairs a pretrained molecule generator (HierG2G, MotifG2G, or MotifG2G$_{Large}$) with two property regressors. The HOMO and LUMO regressors for the base-size models are multilayer perceptrons with $L = 2$ of hidden size $H = 64$. Each layer is paired with a ReLU activation function (Nair and Hinton, 2010). In MotifOptimizer$_{Large}$, we train large versions of each regressor for use with MotifG2G$_{Large}$: $L = 2, H = 128$.

Fine-Tuning

We evaluate the validation loss every 750 training steps and use Early Stopping to avoid overfitting with patience $N = 5$. We consider no improvement to occur if the absolute change in the validation loss between steps t and $t-1$ is less than a threshold $\lambda = 0.1$. Early stopping is applied for all fine-tuning runs.

When using individual optimization (Section 3.4.2) to solve the unbalanced convergence, we assign four different learning rates to four subnetworks ($\eta_{encoder}$, $\eta_{decoder}$, η_{HOMO}, and

η_{LUMO}) such that $\eta_{HOMO}, \eta_{LUMO} < \eta_{encoder}, \eta_{decoder}$. We set $\eta_{encoder} = 1e^{-3}$, $\eta_{decoder} = 1e^{-3}$, $\eta_{HOMO} = 1e^{-4}$, and $\eta_{LUMO} = 1e^{-4}$.

4.2.3 Inference

For each latent space search control method, we experiment with learning rate $\{\eta \mid 0.5 \leq \eta \leq 1e^{-2}\}$. The hyperparameters for each of the three control methods are as follows:

- **Hard Control:** Fixed iterations $T = 50$.
- **Patience Control:** Improvement threshold $\lambda = 0.05$ and patience $N = 5$.
- **Soft Control:** The same as Patience Control, plus the absolute loss threshold $\delta = 0.01$.

4.3 Training Details

All experiments are executed on a single GeForce GTX 1080 graphics card with 16Gb of RAM. Pretraining runs take approximately 28 hours to finish 20 epochs; fine-tuning runs take 4 hours for 5 epochs on average before early stopping is executed.

CHAPTER 5

RESULTS

In next sections, we describe evaluation metrics for the molecule generation task, provide experiment results, and summarize feedback of chemists on generated molecules as human evaluation.

5.1 Evaluation Metrics

The fundamental metrics for evaluating any molecule generation task include chemical validity, uniqueness, diversity, and reconstruction accuracy. Additionally, following (Kusner et-al., 2017; Polykovskiy et-al., 2018), we consider structural and chemical property statistics distributions (i.e., similarity at the structure and property levels) between the predicted and real molecules. For the OPV-specific molecule optimization task, the task is unsupervised, and there are no gold OPVs for comparison. Instead, we consider the similarity of the generated molecules to OPVs as a class: based on domain knowledge gathered from the Chemistry Department, we propose a heuristic metric based on the molecular weight threshold of OPVs. The next two sections describe our selection of evaluation metrics for two tasks: molecule generation and OPV-specific molecule optimization.

5.1.1 Molecular Generation Metrics

We use Moses (Polykovskiy et-al., 2018) to calculate all metrics except reconstruction quality.

Sample Quality metrics measures the sampling distribution of the generated molecules.

- **Validity** is the ratio of the number of chemically valid molecules to the total number of generated molecules. We use the RDkit (Landrum, 2016) package to check the chemical validity based on structural rules.

- **Uniqueness** is the ratio of the number of unique molecules to the number of valid molecules.
- **Diversity** is the pairwise molecular distance among generated molecules (Polykovskiy et-al., 2018).

Property Statistics metrics measure the similarity in properties between the generated and gold molecules.

- **Molecular Weight (MW)** sums the molecular weights of all atoms in a molecule. To evaluate our models' capability for generating large molecules, we use MW for our final model selection.
- **Synthetic Accessibility (SA)** quantifies how easy it is to physically synthesize a molecule.
- **Fréchet ChemNet Distance (FCD)** uses the ChemNet neural network (Preuer et al., 2018) to estimate and measure similarity in the chemical properties between generated and gold molecules. Suggested by (Polykovskiy et-al., 2018), we also use FCD to selec the final model during the pretraining and fine-tuning stages.

Structural Statistics measure similarity in structures between the generated and gold molecules.

- **Nearest Neighbor Similarity (SNN)** is the average Tamimoto similarity of generated molecules to the nearest gold molecule.
- **Fragment Similarity (Frag)** breaks the molecules into fragments and measures the cosine similarity between the generated and gold sets of fragments. If both sets have similar fragments, Frag is large; if some fragments from the generated set do not exist in the gold set, Frag will be lower.

5.1.2 OPV-Specific Metrics and Human Evaluation

Given a generated molecule, its fitness as an OPV instance can be measured through its molecular weight and through human evaluation by domain expert chemists.

Molecular-Weight-based Indicator (MW-I). According to the Chemistry Department, the molecular weight of OPVs ranges from 400 g/mol to 3,000 g/mol. We use Moses (Polykovskiy et-al., 2018) to calculate the molecular weight of a generated molecule. A molecule is labeled 1 or "potential OPV candidate" if its molecular weight falls within the molecular weight range of OPVs.

Human Evaluation. From the pool of generated molecules, only promising OPV molecule candidates are selected using prescreening with RDkit for chemical validity and MW-I. Without time-consuming physical experiments, the chemists could not evaluate the molecule candidates statistically. Hence, for human evaluation, we collect their qualitative feedback on the generated molecules and summarize them into the following factors:

- **Synthesis Accessibility.** We ask chemists if it is possible to physically synthesize the candidate molecule. If so, is it straightforward to do based on current knowledge?

- **Fragment Familiarity.** Molecular generative models are not guaranteed to output valid and useful OPVs, but they could still suggest potential molecular structures. We ask, chemists to examine the candidate molecules and identify OPV-like substructures.

- **Promise.** Each chemist, based on their own chemistry knowledge and their past experience, subjectively decides if the given molecule candidate is promising with minimal modifications.

5.2 Discussion

To understand performance of our approach, we compare the different experiments described in Chapter 4 and provide human evaluation from chemists in following sections.

5.2.1 Generation Quality for Large Molecules

Table 5.1 summarizes the generation performance of different models over three test sets with different molecule sizes. QM9 and Curated-OPV contain the smallest and largest molecules, respectively, with HOPV15 molecule size in the middle. At the smallest molecule size (QM9), the performance of MotifG2G$_{Large}$ is on par with HierG2G in terms of nearest neighbor similarity and fragment similarity and structural similarity metrics. Compared to HierG2G, MotifG2G$_{Large}$ yields lower molecular weight and lower FCD property similarity. These results suggest that the performance of MotifG2G$_{Large}$ remains comparable with that of HierG2G despite losing the fine-grained atom layer information.

When the molecule size increases (HOPV15 and Curated-OPV test sets), MotifG2G$_{Large}$ outperforms HierG2G in terms of molecular weight and FCD property similarity, while preserving performance in structural statistics. This observation supports our argument in Section 3.3.2 that the motif-only representation benefits generating large molecules without harming the molecular property approximation. For each test set on increasing molecule size, the rise in the synthesis accessibility score of MotifG2G$_{Large}$ compared to HierG2G$_{Large}$ is likely due to loss of the atomic information; this is expected, since the goal of omitting the atom representation is output larger molecules. By definition of the SA score, large molecules like OPVs likely leads to lower synthesis accessibility since they contain many complex substructures (i.e., large ring size > 8) and less common fragments compared with reference fragments (Ertl and Schuffenhauer, 2009).

The significant drop in structural similarity metrics on the HOPv15 and Curated-OPV test sets are also expected. Since QM9's small molecules accounts for a major portion of

the training set, the generative models are biased towards learning simple and small graph structures. Suggested by (Polykovskiy et-al., 2018), we use the FCD property similarity to select the final model for fine-tuning. Hence, given the limited number of large molecules available for training, we select HierG2G and MotifG2G$_{Large}$ as backbones for the fine-tuning stage based on the property and structural similarity metrics.

5.2.2 Effects of Loss-Scaling during Fine-Tuning

Table 5.2 summarizes the results of fine-tuning the molecule optimizers and the effects of the three different loss-scaling methods used to address the unbalanced convergence between the graph generation and property regression tasks (Section 3.4). We find that the Task-Dependent (TD) and Individually-Optimizing (IO) methods show no significant improvement compared with the default, equally-weighted loss scaling setting for all metrics. Since the loss-scaling methods yield no improvement among the molecular weight, fragment similarity, and MAEs of HOMO and LUMO scores, which are our primary molecule generation and property regression metrics, we use the default, equally-weighted losses in the rest of our experiments.

5.2.3 Optimization for OPV-Specific Molecule Generation

Looking at Table 5.2, MotifOptimizer$_{Large}$ outperforms HierOptimizer in terms of the molecular weight and FCD property similarity metrics, while preserving equivalent structural similarity for all molecule size ranges, we select MotifOptimizer$_{Large}$ for inference. We compare three optimization methods (Section 3.5) for generating property-optimized OPV molecule candidates in Table 5.3. We also use the Molecular-Weight Indicator (MW-I) to determine if a generated molecule is likely to be an OPV based on the molecular weight thresholds of OPVs (see Section 5.1).

Table 5.1: Results of the molecule reconstruction task on three test sets, QM9, HOPV15, and Curated-OPV, of increasing molecule size. For each test set block, the first row reports the oracle performance using the corresponding test set as generated samples. By default, all models have tied embeddings. *Val.* means chemical validity; *Div.* means diversity; *MW* means molecular weight; *SA* means synthetic accessibility; *FCD* means Fréchet ChemNet Distance; *SNN* means nearest neighbor similarity; *Frag* is fragment similarity. Except property statistics, all metrics are better when higher.

		Sample Quality (↑)			Property Statistics (↓)			Structural Statistics (↑)	
Dataset	Method	Val.	Unique	Div.	MW	SA	FCD	SNN	Frag
QM9	Test Set	100%	100%	0.92	1.53	9.4e-5	7.6e-13	1.00	1.00
	HierG2G	**100%**	**99.0%**	**0.92**	16.9	0.22	0.39	0.73	0.98
	MotifG2G	**100%**	98.0%	**0.92**	25.8	**0.17**	0.84	0.55	0.97
	MotifG2G$_{Large}$	**100%**	93.5%	**0.92**	**13.1**	0.22	**0.32**	**0.75**	**0.99**
HOPV15	Test Set	100%	100%	0.79	2.2e-14	0.00	0.00	1.00	1.00
	HierG2G	**100%**	98.1%	0.83	129.3	**0.08**	14.8	0.47	0.90
	MotifG2G	**100%**	98.1%	**0.87**	294.6	0.57	21.8	0.30	0.88
	MotifG2G$_{Large}$	**100%**	98.1%	0.83	**73.28**	0.36	**11.1**	**0.48**	**0.92**
Curated-OPV	Test Set	100%	100%	0.57	3e-13	0.00	1.00	1.00	1.00
	HierG2G	**100%**	92.0%	0.75	681.0	**0.63**	20.9	**0**.37	**0.89**
	MotifG2G	**100%**	91.6%	**0.77**	1113	2.40	43.5	0.17	0.10
	MotifG2G$_{Large}$	**100%**	**100%**	0.71	**269.9**	1.17	**20.5**	0.35	0.88

Table 5.2: Comparison of three loss-scaling methods on HierOptimizer and MotifOptimizer$_{Large}$ over the three test sets. For display, *HierOpt* means HierOptimizer; *MotifOpt* means MotifOptimizer$_{Large}$; *EW* means loss terms weighted equally; *TD* is the Task-Dependent loss-scaling method; *IO* is the Individually-Optimizing loss-scaling method.

Dataset / Method	Sample Quality (↑)			Property Statistics (↓)			Structural Statistics (↑)		MAE (↓)	
	Val.	Unique	Div.	MW	SA	FCD	SNN	Frag	HOMO	LUMO
QM9										
HierOpt$_{EW}$	**100%**	90.69%	**0.92**	17.77	0.23	0.46	0.68	0.98	0.04	0.04
HierOpt$_{TD}$	**100%**	91.40%	**0.92**	16.95	0.22	0.42	0.67	0.98	**0.02**	**0.03**
HierOpt$_{IO}$	**100%**	**91.87%**	**0.92**	17.45	0.22	0.42	0.71	0.98	**0.02**	**0.03**
MotifOpt$_{EW}$	**100%**	92.71%	**0.92**	13.19	0.23	0.32	0.74	**0.99**	0.05	0.04
MotifOpt$_{TD}$	**100%**	**92.87%**	**0.92**	**13.10**	**0.21**	**0.31**	**0.75**	**0.99**	0.03	**0.03**
MotifOpt$_{IO}$	**100%**	90.94%	**0.92**	15.88	0.23	0.40	0.69	0.98	**0.02**	0.04
HOPV15										
HierOpt$_{EW}$	**100%**	94.34%	0.84	161.98	**0.17**	17.56	0.40	0.91	0.17	**0.06**
HierOpt$_{TD}$	**100%**	**98.11%**	**0.87**	291.16	0.33	22.94	0.33	0.87	0.07	0.13
HierOpt$_{IO}$	**100%**	94.34%	0.84	158.14	0.20	**16.80**	0.41	**0.92**	**0.06**	0.11
MotifOpt$_{EW}$	**100%**	94.34%	0.82	97.26	0.46	12.27	**0.49**	0.91	0.07	0.07
MotifOpt$_{TD}$	**100%**	94.33%	0.83	93.25	0.40	**11.21**	**0.49**	0.91	0.20	0.15
MotifOpt$_{IO}$	**100%**	92.45%	0.83	**81.06**	0.40	13.27	0.46	0.91	0.26	0.11
Curated-OPV										
HierOpt$_{EW}$	**100%**	**100%**	0.80	907.28	1.50	28.26	0.29	0.83	4.99	3.9
HierOpt$_{TD}$	**100%**	**100%**	**0.85**	1325.33	2.06	48.80	0.10	0.45	5.15	4.00
HierOpt$_{IO}$	**100%**	**100%**	0.76	780.01	1.24	25.29	0.35	0.90	5.05	3.97
MotifOpt$_{EW}$	**100%**	91.67%	0.69	295.85	1.11	**17.49**	0.37	**0.92**	5.11	3.88
MotifOpt$_{TD}$	**100%**	**100%**	0.68	**226.11**	**0.91**	20.06	**0.38**	0.87	2.60	1.09
MotifOpt$_{IO}$	**100%**	**100%**	**0.70**	**339.98**	1.1	21.21	0.31	0.84	**1.82**	**0.91**

For both OPV test sets, HOPV15 and Curated-OPV, we see that, compared to the non-optimized MotifOpt$^{\bigstar}$, the Soft and Patience optimization methods produce slight improvements in the molecular weight, synthesis accessibility, structural similarity scores, and the total regression errors (i.e. $DTT-MAE_{HOMO}+DTT-MAE_{LUMO}$). Besides, two methods yield significant improvement in MW-I. Additionally, there is no significant difference in performance between the Soft and Patience methods. This observation is likely due to their similar criteria of stopping optimization (i.e., closely matching target HOMOs/LUMOs vs no improvement in distance to target HOMOs/LUMOs). The inconsistent performance of the Hard optimization method is expected, since there exists no criteria to stop optimization when reaching the optimal latent vectors. The result is higher chance of overshooting the optimal region of the latent space. We select the Soft optimization method since the stopping criteria allows the optimization to reach the minimum distance to targets in an ideal situation.

5.2.4 Human Evaluation

We use the final model MotifOptimier$_{Large}$ from the fine-tuning stage to generate molecule candidates given input molecules from the Curated-OPV test set and collect feedback from chemists on three metrics: Synthesis Accessibility, Fragment Familiarity, and Promise. Figure 5.1 displays the five best generated molecules and their original input molecules.

Looking at each generated molecule in Fig. 5.1, each red square denotes a set of weakly-connected atoms or rings (called weak fragments) that may easily split apart and form multiple separate, smaller molecules. In other words, synthesizing weakly-connected molecules is impossible or only possible under strict conditions. Minimal modifications to the weakly connected atoms/rings could ease or simplify the synthesis process, such as changing the intersection atoms between a ring and its neighbor.

Weak fragments in the generated molecules are similar to few OPV patterns in the original molecules (e.g., compare the blue circles in *(a)* with those in *(b)*. Similarly, other

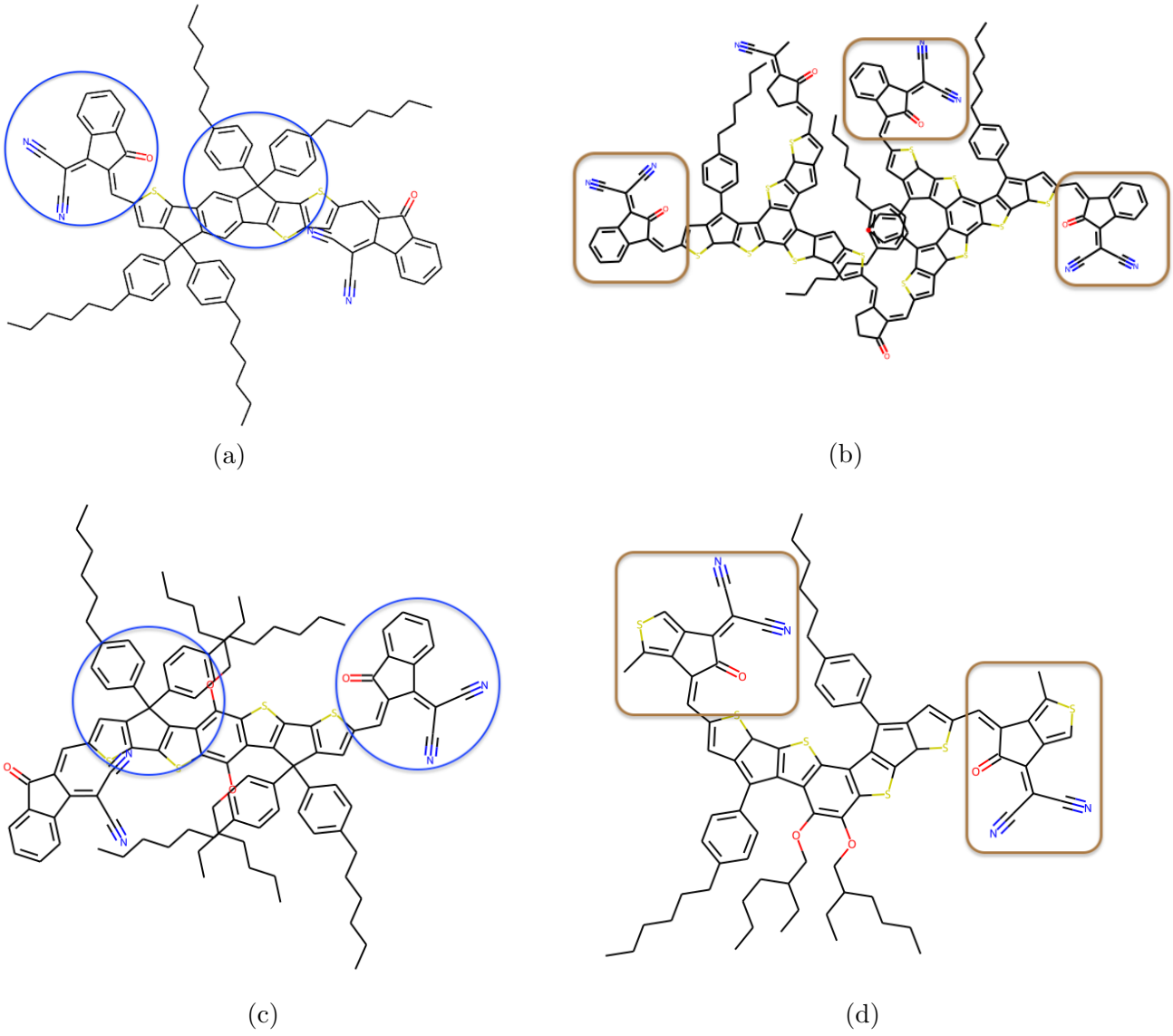
Table 5.3: Comparison of three optimization methods (e.g., Soft, Patience, Hard) on the Curated-OVP test set when learning rate $\eta = 0.2$. *Soft* means Soft Control; *Patience* means Patience Control; *Hard* means Hard Control. In each dataset block, the first row shows results of the best performing model selected from the fine-tuning stage before conducting optimization. ★ means executing the best performing MotifOptimizer$_{Large}$ determined at the fine-tuning stage to predict property values and to generate molecule candidates. *DTT-MAE* used MAE to measure distance from current property values to target property values.

	Sample Quality (↑)			Property Stat. (↓)			Structural Stat.(↑)		DTT-MAE (↓)		
Method	Val.	Unique	Div.	MW	SA	FCD	SNN	Frag	HOMO	LUMO	MW-I
MotifOpt$^{\bigstar}$	**100%**	91.67%	0.69	295.85	1.11	**17.49**	0.37	**0.92**	5.11	3.88	75.00%
MotifOpt$_{\ddagger,Soft}$	**100%**	91.67%	**0.72**	**256.14**	1.09	21.83	**0.38**	0.86	**5.05**	**3.86**	**83.30%**
MotifOpt$_{\ddagger,Patience}$	**100%**	91.67%	**0.72**	**256.14**	1.09	21.83	**0.38**	0.86	**5.05**	**3.86**	**83.30%**
MotifOpt$_{\ddagger,Hard}$	**100%**	**100%**	0.70	271.05	**1.05**	18.67	**0.38**	0.89	5.09	3.87	75.00%

OPV patterns can be seen in the generated molecules (e.g., compare *(e)* and *(f)* or *(g)* and *(h)*). We can see that there exists certain similarities between the original and generated molecules at the fragment level.

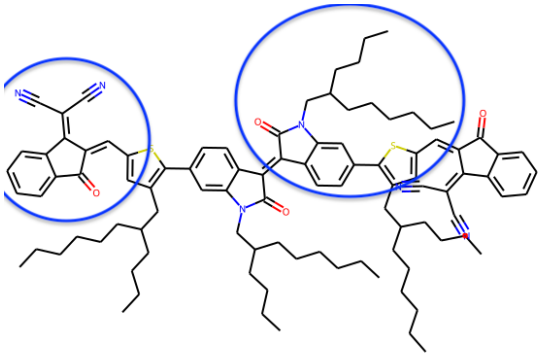
By modifying weak fragments, the generated molecules have the potential to be OPV-type molecules and worth taking the time for physical experiments. The generated molecule *(f)* is judged to be the most useful and promising molecule candidate for physical testing. In the generated molecules *(j)*, the rings in the middle, without the long chains on the sides, had actually been physically tested and used to make solar cells decades ago. Generally, the generated molecules could become potential OPV molecules with some modifications. This evaluation suggests that MotifOptimizer$_{Large}$ is capable of learning useful substructures for OPV molecules.

Figure 5.1: Pairs of the original and generated OPV molecules. For each row, the original molecule is on the left and its corresponding generated molecule is on the right. Blue circles denote substructures of atoms and rings that are familiar in OPV molecules. Red squares denote weak fragments that require modifications (i.e., changes in the connection position between a motif with its neighbors) for higher synthesizability. Generated molecules in images f and j are judge by the chemists to be the two most useful and promising candidates to be OPV molecules.

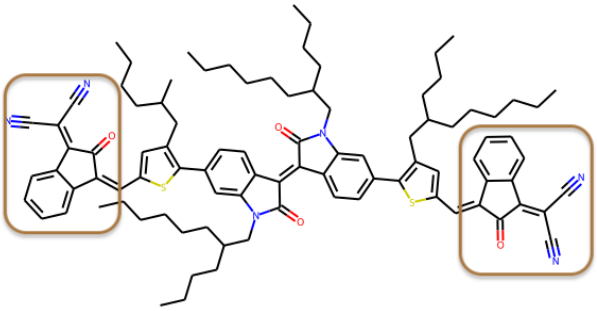


(a) (b)

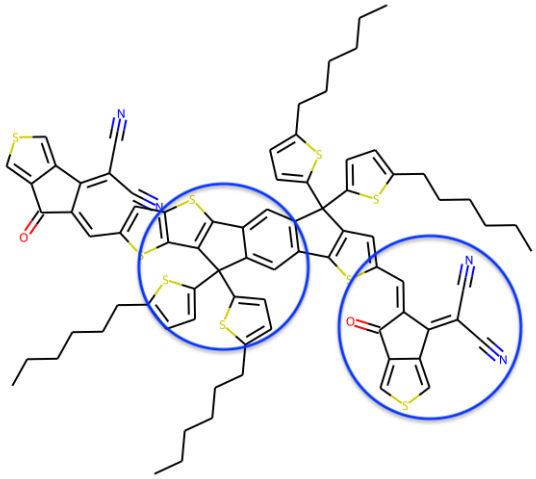
(c) (d)



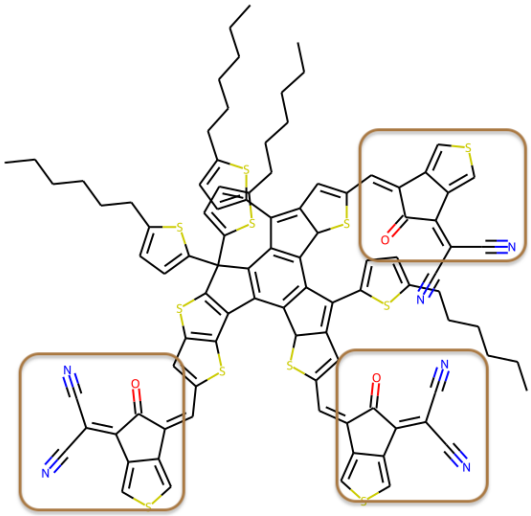
(e)



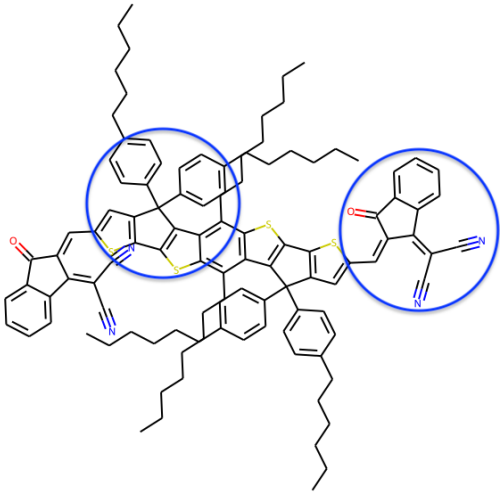
(f)



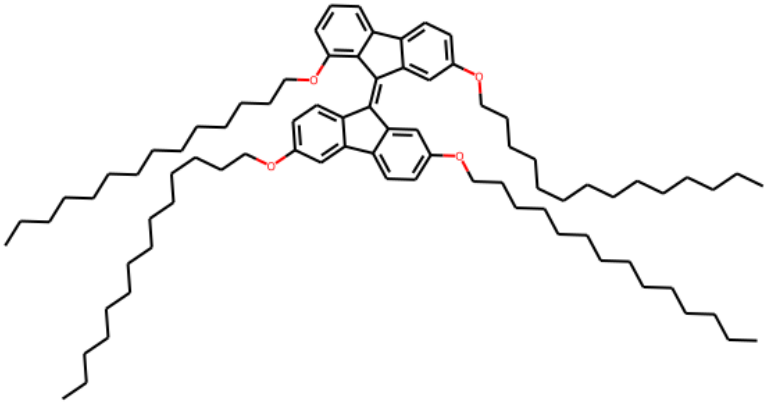
(g)



(h)



(i)



(j)

CHAPTER 6

CONCLUSION

Although the scientific and/or creative contributions of a dissertation are its most important qualities, the impact of those contributions remains contingent upon a clear, lucid presentation that invites wide readership. High-quality typography and consistent typographic style is particularly important when the dissertation content is highly technical.

This UTD thesis class file aids the dissertation author (and future readers) by automating much of the required formatting dictated by the UTD graduate office. It does so in a way that avoids conflicts with existing LaTeX style packages, maximizing the set of tools that remain at the author's disposal.

CHAPTER 7

GRAPH NEURAL NETWORKS FOR PROPERTY-GUIDED MOLECULE OPTIMIZATION

Authors – Dat Quoc Ngo

The Computer Science Department, EC 31

The University of Texas at Dallas

800 West Campbell Road

Richardson, Texas 75080-3021

Key words: motif, graph neural networks, material design, molecule generation
Corresponding author: Dat Quoc Ngo

This sample text demonstrates the formatting of a chapter headed by a *separate chapter title page* (see p. 51). Separate chapter title pages are for chapters comprised of a verbatim work that has already been published or submitted for publication. In general, you should consult your supervising professor and the graduate office to determine whether a separate chapter title page is appropriate for your dissertation chapter.

APPENDIX

PRESCREENING PROMISING MOLECULES

APPENDIX

MESSAGE PASSING NETWORK (MPN)

APPENDIX

CANDIDATE ATTACHMENT CHECK

REFERENCES

Weininger, D. (1988). *SMILES, a chemical language and information system,*. pp. 31-36. Journal of Chemical Information and Modeling.

Gomez-Bombarelli, R., J. N. Wei. , D. Duvenaud, J. M. Hernańdez-Lobato, B. Sańchez-Lengeling, D. Sheberla, J. Aguilera-Iparraguirre, T. D. Hirzel, R. P. Adams, and A. Aspuru-Guzik (2018). *Automatic Chemical Design Using a Data-Driven Continuous Representation of Molecules,*. 4. pp. 268-276. American Chemical Society.

Segler M. H. S., T. Kogej, C. Tyrchan, and M. P. Waller (2018). *Generating Focused Molecule Libraries for Drug Discovery with Recurrent Neural Networks,*. 4. pp. 120-131 American Chemical Society.

Kusner, M. J., B. Paige, and J. M. Hernańdez-Lobato (2017). *Grammar Variational Autoencoder.* International Conference on Machine Learning.

Dai, H., Y. Tian, B. Dai, S. Skiena, and L. Song (2018). *Syntax-Directed Variational Autoencoder for Structured Data.* International Conference on Machine Learning.

Guimaraes, G., B. Sanchez-Lengeling, C. Outeiral, P. L. C. Farias, and A. Aspuru-Guzik (2018). *Objective-Reinforced Generative Adversarial Networks (ORGAN) for Sequence Generation Models.*

Jin, W., R. Barzilay, and T. Jaakkola (2020). *Hierarchical Generation of Molecular Graphs using Structural Motifs.* International Conference on Machine Learning.

Kang, S. and K. Cho (2018). *Conditional Molecular Design with Deep Generative Models.* Journal of Chemical Information and Modeling.

You, J., B. Liu, R. Ying, V. Pande, and J. Leskovec (2019). *Graph Convolutional Policy Network for Goal-Directed Molecular Graph Generation.* Neural Information Processing Systems.

Liu, Q., M. Allamanis, M. Brockschmidt, and A. L. Gaunt (2019). *Constrained Graph Variational Autoencoders for Molecule Design.* Neural Information Processing Systems.

Jin, W., R. Barzilay, and T. Jaakkola (2018). *Junction Tree Variational Autoencoder for Molecular Graph Generation.* International Conference on Machine Learning.

Jin, W., K. Yang, R. Barzilay, and T. Jaakkola (2019). *Learning Multimodal Graph-to-Graph Translation for Molecular Optimization.* International Conference on Machine Learning.

Heller, S., A. McNaught, S. Stein, D. Tchekhovskoi, and I. Pletnev (2013). *InChl - the Worldwide Chemical Structure Identifier Standard.* Journal of Cheminformatics.

Hochreiter, S. and J. Schmidhuber (1997). *Long Short-Term Memory.* Neural Computation.

Sutskever, S., O. Vinyals, and Q. V. Le (2014). *Sequence to Sequence Learning with Neural Networks.* Neural Information Processing Systems.

Kingma, D. and M. Welling (2013). *Auto-Encoding Variational Bayes.* International Conference on Learning Representations.

Olivecrona, M., T. Blaschke, O. Engkvist, and H. Chen (2017). *Molecular De-novo Design Through Deep Reinforcement Learning.* Journal of Cheminformatics.

Bowman, S. R., L. Vilnis, O. Vinyals, A. M. Dai, R. Jozefowicz, and S. Bengio (2015). *Generating Sentences from a Continouss Space.* Conference on Computational Natural Language Learning.

Popova, M., O. Isayev, and A. Tropsha (2018). *Deep Reinforcement Learning for De Novo Drug Design.*

Landrum, G. (2016). *RDKit.*

Hamilton, W., Z. Ying, and J. Leskovec (2017). *Inductive representation learning on large graphs.* Neural Information Processing Systems.

Kipf, N. and M. Welling (2016). *Semi-Supervised Classification with Graph Convolutional Networks.* International Conference on Learning Representations.

Li, Y., O. Vinyals, C. Dyer, R. Pascanu, and P. Battaglia (2018). *Learning Deep Generative Models of Graphs.*

You, J., R. Ying, X. Ren, W. L. Hamilton, and J. Lesokvec (2018). *GraphRNN: a Deep Generative Model for Graphs.* International Conference on Machine Learning.

Samanta, B., A. De, G. Jana, V. Gomez, P. K. Chattaraj, N. Ganguly, and M. Gomez-Rodriguez (2020). *NeVAE: A Deep Generative Model for Molecular Graphs.* Journal of Machine Learning Research.

Zhou, Z. S. Kearnes, R. N. Zare, and P. Riley (2019). *Optimization of Molecules via Deep Reinforcement Learning.*

Ma, T., J. Chen, and C. Xiao (2018). *Constrained Generation of Semantically Valid Graphs via Regularization* Neural Information Processing Systems.

De Cao, N. and T. Kipf (2018). *MolGAN: An Implicit Generative Model for Small Molecular Graphs*

Simonovsky, M. and N. Komodakis (2018). *GraphVAE: Towards Generation of Small Graphs using Variational Autoencoders.*

Baumer, L., G. Sala, and G. Sello (1991). *Ring Perception in Organic Structures: A New Algorithm for Finding SSSR.* Computational Geometry.

Polykovskiy, D., A. Zhebrak, B. Sanchez-Lengeling, S. Golovanov, O. Tatanov, S. Belyaev, R. Kurbanov, A. Artamonov, V. Aladinskiy, M. Veselov, A. Kadurin, S. Nikolenko, A. Aspuru-Guizk, and A. Zhavoronkov (2018). *Molecular Sets (MOSES): A Benchmarking Platform for Molecular Generation Models*

Rogers, D. and M. Hahn (2010). *Extended-connectivity fingerprints.* Journal of Chemical Information and Modeling.

Ruddigkeit, L., Deursen, R., Blum, L., and J. Reymond (2012). *Enumeration of 166 Billion Organic Small Molecules in the Chemical Universe Dataset GDB-17.* Journal of Chemical Information and Modeling.

Ramakrishnan, R., Dral, P., Rupp, M., and O. Lilienfeld, (2014). *Quantum Chemistry Structures and Properties of 134 Kilo Molecules.* Scientific Data 1.

A. Aspuru-Guzik (2015). *Harvard Organic Photovoltaic Dataset (HOPV15) Dataset*

Kendall, A., Gal, Y., and R. Cipolla (2018). *Multi-Task Learning Using Uncertainty to Weigh Losses for Scene Geometry and Semantics.* Conference on Computer Vision and Pattern Recognition.

Kingma, D. and J. Ba (2014). *Adam: A Method for Stochastic Optimization.* International Conference on Learning Representations.

Choudhary, K., M. Bercx, J. Jiang, R. Pachter, D. Lamoen, and F. Tavazza(2019). *Accelerated Discovery of Efficient Solar Cell Materials Using Quantum and Machine-Learning Methods.* American Chemical Society.

Saeki, A., and K. Kranthiraja (2020). *A High Throughput Molecular Screening for Organic Eletronics via Machine Learning: Present Status and Perspective.* Japanese Journal of Applied Physics.

Kanal, I., S. Owens, J. Bechtel, and G. Hutchison (2013). *Efficient Computational Screening of Organic Polymer Photovoltaics.* American Chemical Society.

Gomez-Bombaelli, R., J. Aguilera-Iparraguirre, T. Hiel, D. Duvenaud, D. Maclaurin, M. Blood-Forsthe, H. Chae, M. Einzinger, D. Ha, T. Wu, G. Markopoulos, S. Jeon, H. Kang, H. Miyazaki, M. Numata, S. Kim, W. Huang, S. Hong, M. Baldo, R¿ Adams, and A. Aspuru-Guzik (2016). *Design of Efficient Molecular Organic Light-Emitting Diodes by a High-Throughput Virtual Screening and Experimental Approach.* Nature Materials.

Li, H., Z. Zhong, L. Li, R. Gao, J. Cui, T. Gao, L. Hu, Y. Lu, Z. Su, and H. Li (2015). *A cascaded QSAR model for efficient prediction of overall power conversion efficiency of all-organic dye-sensitized solar cells.* Journal of Computational Chemistry.

Pereira, F., K. Xiao, D. Latino, C. Wu, Q. Zhang, and J. Aires-de-Sousa (2017). *Machine Learning Methods to Predict Density Functional Theory B3LYP Energies of HOMO and LUMO Orbitals.* American Chemical Society.

Chen, C., W. Ye, Y. Zuo, C. Zheng, and S. Ong (2019). *Graph Networks as a Universial Machine Learning Framework for Models and Crystals.* American Chemical Society.

Takeda S., T. Hama, H. Hsu, V. Piunova, D. Zubarev, D. Sanders, J. Pitera, M. Kogoh, T. Hongo, Y. Cheng, W¿ Bocanett, H. Nakashika, A. Fujita, Y. Tsuchiya, K. Hino, K. Yano, S. Hirose, H. Toda, Y. Orii, and D. Nakano (2020). *Molecular Inverse-Design Platform for Material Industries.* ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.

Werbos, P.J. (1990). *Backpropagation Through Time: What It Does And How To Do It.* Proceedings of the IEEE.

Pascanu, R., T. Mikolov, and Y. Bengio (2013). *On the Difficulty of Training Recurrent Neural Networks.* Proceedings of the 30th International Conference on Machine Learning.

Press, O. and L. Wolf (2017). *Using the Output Embedding to Improve Language Models.* Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics.

Nair, V. and Geoffrey E. Hinton. (2010). *Rectified Linear Units Improve Restricted Boltzmann Machines.* Proceedings of the 27th International Conference on Machine Learning.

Preuer, K., P. Renz, T. Unterthiner, S. Hochreiter, and G. Klambauer. (2018). *Fréchet ChemNet Distance: A Metric for Generative Models for Molecules in Drug Discovery.* Journal of Cheminformatics.

Smith, D., L. A. Burns, A. C. Simmonett, R. M. Parrish, M. C. Schieber, R. Galvelis, P. Kraus, H. Kruse, R. Di Remigio, A. Alenaizan, A. M. James, S. Lehtola, J. P. Misiewicz, M. Scheurer, R. A. Shaw, J. B. Schriber, Y. Xie, Z. L. Glick, D. A. Sirianni, J. S. O'Brien, J. M. Waldrop, A. Kumar, E. G. Hohenstein, B. P. Pritchard, B. R. Brooks, H. F. Schaefer III, A. Yu. Sokolov, K. Patkowski, A. E. DePrince III, U. Bozkaya, R. A. King, F. A. Evangelista, J. M. Turney, T. D. Crawford, C. D. Sherrill. (2020). *PSI4: Open-Source Software for High-Throughput Quantum Chemsitry.* Journal of Chemistry Physics.

Ramsundar, B., P. Eastmand, P. Walters, V. Pande, K. Leswing, and Z. Wu (2019). *Deep Learning for the Life Sciences.* O'Reilly Media.

Chithrananda, S., G. Grand, and B. Ramsundar (2020). *ChemBERTa: Large-Scale Self-Supervised Pretraining for Molecular Property Prediction.* Neural Information Processing Systems (NeurIPS).

Sennrich, R., B. Haddow, and A. Brich (2016). *Neural Machine Translation of Rare Words with Subword Units.* Associate of Computational Linguistics (ACL).

Wu, Z., B. Ramsundar, E. Feinberg, J. Gomes, C. geniesse, A. Pappu, K. Leswing, and V. Pande (2018). *MoleculeNet: A Benchmark for Molecular Machine Learning.*

Cheng, H., Y. Zhao, and Y. Yang (2022). *Toward High-Performance Semitransparent Organic Photovoltaics with Narrow-Bandgap Donors and Non-Fullerence Accepts.* Advanced Energy Materials.

Schütt, K, P. Kindermans, H. Sauceda, S. Chmiela, A. Tkatchenko, K. Müller (2017). *SchNet: A Continuous-Filter Convolutional Neural Network for Modeling Quantum Interactions.* Journal of Chemical Physics.

Xia, L., Z. Feng, H. Zhang, J. Song, Z. Zhong, S. Yao, and M. Song (2022). *Explainable Fragment-based Molecular Property Attribution.* Journal of Advanced Intelligent Systems.

Zhang, Z., J. Guan, S. Zhou (2021). *FraGAT: a fragment-oriented multi-scale graph attention model for molecular property prediction.* Journal of Advanced Intelligent Systems.

Ertl, P., and A. Schuffenhauer (2009). *Estimation of synthetic accessibility score of drug-like molecules based on molecular complexity and fragment contributions.* Journal of Cheminformatics.

BIOGRAPHICAL SKETCH

Dat Quoc Ngo

CURRICULUM VITAE

Dat Quoc Ngo

November 11, 2022

Contact Information:

Department of Computer Science
The University of Texas at Dallas
800 W. Campbell Rd.
Richardson, TX 75080-3021, U.S.A.

Voice: (714) 813-0576
Email: dqn170000@utdallas.edu

Educational History:

B.S., Computer Science, The University of Texas at Dallas, 2020
M.S., Computer Science, The University of Texas at Dallas, 2022

Graph Neural Networks for Property-guided Molecule Generation
M.S. Thesis
Department of Computer Science, The University of Texas at Dallas
Advisors: Dr. Jessica Jin Ouyang

Employment History:

Software Engineer Intern, Meta Platforms Inc., May 2022 - Aug. 2022
Machine Learning Engineer Intern, 7-Eleven, Jan. 2022 - May 2022
Research Intern, Samsung Research America, June 2021 - Aug. 2021
Research Intern, deepkapha.ai, Oct. 2020 - May 2021
Software Engineer Intern, Surfboard One, Mar. 2020 - Oct. 2020

Professional Recognitions and Honors:

Professional Memberships: