Multi-resolution Autoregressive Graph-to-Graph Translation for Molecules

Wengong Jin **Regina Barzilay** **Tommi Jaakkola**
CSAIL, Massachusetts Institute of Technology
{wengong, regina, tommi}@csail.mit.edu

Abstract

The problem of accelerating drug discovery relies heavily on automatic tools to optimize precursor molecules to afford them with better biochemical properties. Our work in this paper substantially extends prior state-of-the-art [22] on graph-to-graph translation methods for molecular optimization. In particular, we realize coherent multi-resolution representations by interweaving trees over substructures with the atom-level encoding of the original molecular graph. Moreover, our graph decoder is fully autoregressive, and interleaves each step of adding a new substructure with the process of resolving its connectivity to the emerging molecule. We evaluate our model on multiple molecular optimization tasks and show that our model outperforms previous state-of-the-art baselines by a large margin.

1 Introduction

The goal of molecular optimization is to learn to modify compounds so as to improve their biochemical properties. The task is challenging since the chemical space of potential candidates is vast, and molecular properties can be complex functions of structural features of varying sizes. This task has received significant recent interest from the ML community owing to its practical importance and increasing access to property datasets. Existing methods can be divided roughly speaking along two axes – representation and optimization. On the representation side, they either operate on SMILES strings [13, 29, 6] or directly on molecular graphs [21, 41, 35]. On the optimization side, the task has been formulated as reinforcement learning [15, 51, 53], continuous optimization in the latent space learned by variational autoencoders [13, 29, 21], or molecular translation [22].

We take graph-to-graph translation as a starting point. At a high level, the approach is analogous to machine translation: given a corpus of molecular pairs $\{(X, Y)\}$, where Y is a paraphrase of X with better chemical properties, the model is trained to translate any input molecular graph into its better form. Similar to machine translation, success in this task is predicated on the inductive biases built into the encoder-decoder architecture, in particular the process of realizing candidate molecular structures. Prior work [22] proposed a junction tree encoder-decoder and employed a tree structured graph (the junction tree) over chemical substructures in addition to the original atom-level graph. While successful, the approach remains limited in several ways. The encoding for the two graphs was carried out separately, and decoding proceeded in strictly successive steps, first the junction tree for the new molecule, followed by assembling its substructures together. The assembling process involved local independence assumptions so as to limit combinatorial explosion. Therefore the resulting method has limited control over substructure-attachment relations.

We propose a multi-resolution, hierarchically coupled encoder-decoder where the auto-regressive decoder interleaves the prediction of substructure components and how such components should be attached to the molecule being built. The encoding of each molecule proceeds across three levels, from graph convolutions of atoms at the lowest level to the coarse tree over substructures at the highest level. The intermediate layer representing contextualized substructures (with attachments) serves as

Preprint. Under review.

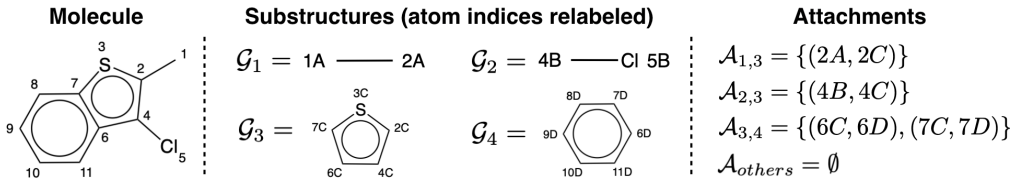

Figure 1: Substructures and their attachments of a given molecule. Each substructure $\mathcal{G}_i$ is a subgraph in the molecule. Each attachment $\mathcal{A}_{i,j}$ is a list of matched atom pairs between subgraphs i and j. Atom indices in the substructures are relabeled by appending A in $\mathcal{G}_1$, B in $\mathcal{G}_2$, C in $\mathcal{G}_3$ and D in $\mathcal{G}_4$.

the glue to span the hierarchy. Our auto-regressive decoder enables us to model strong dependencies between successive attachments and substructure choices. The target graphs are unraveled as a sequence of triplet choices (where to expand the graph, new substructure type, its attachment). We also extend the method to handle directed or conditional translation where the desired characteristics (criteria) are fed as input to the translation process. This enables the method to translate any precursor to a target that satisfies combinations of criteria not present during training.

We evaluate our new model on multiple molecular optimization tasks, from single-property to multi-property optimization. As baselines, we compare against previous state-of-the-art graph generation methods [22, 51]. Our model significantly outperforms these methods in discovering molecules with desired properties, yielding 11% relative gain on average and 28% best relative improvement on QED optimization. Moreover, we show that directed translation can succeed (generalize) even when trained on molecular pairs with only 5% of them having desired target property combination.

2 Background and Motivation

Molecules are represented as graphs $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ is the set of atoms and $\mathcal{E}$ the set of bonds. Following [22], we frame molecular optimization as a graph to graph translation task. Given a corpus of molecular pairs $\{(X, Y)\}$, the task is to learn to translate an input molecular graph X into another graph Y with better chemical properties. Our work significantly extends prior work [22] by providing an integrated multi-resolution graph encoding as well as by introducing a multi-resolution, auto-regressive graph decoding. Here we motivate our modeling choices for graph generation.

Graphs are challenging objects to realize. We must first decide the appropriate building blocks for constructing molecules. In this respect, we proceed similarly to previous works [21, 22] that use valid chemical substructures (e.g., aromatic rings) as structural building blocks. The key advantage of building molecules in terms of larger units is that the decoder can realize valid molecules without being forced to go through chemically invalid intermediaries as in atom-by-atom generation.

We can write the probability of a graph $\mathcal{G}$ as a joint distribution over substructures $\mathcal{G}_1, \cdots, \mathcal{G}_n$ constituting $\mathcal{G}$, together with their attachments $\{\mathcal{A}_{i,j} \mid 1 \leq i < j \leq n\}$. Each attachment $\mathcal{A}_{i,j}$ represents the set of intersecting atoms between structures $\mathcal{G}_i$ and $\mathcal{G}_j$ (see Figure 1). Prior works [21, 22] adopted a two-stage procedure for realizing this joint distribution. The first step generates a junction tree with substructures $\mathcal{G}_1, \mathcal{G}_2, \cdots, \mathcal{G}_n$ as nodes, capturing a coarse relative arrangements of the components. The second step resolves the full graph by specifying how the substructures were attached to each other. The corresponding factorization is:

$$P(\mathcal{G}) = P_\theta(\mathcal{G}_1, \mathcal{G}_2, \cdots, \mathcal{G}_n) P_\psi(\{\mathcal{A}_{i,j}\} \mid \mathcal{G}_1, \mathcal{G}_2, \cdots, \mathcal{G}_n) \tag{1}$$

A local independence assumption was needed to define $P_\psi(\cdot)$ so as to avoid combinatorially enumerating all possible attachment configurations. Moreover, distributions P_θ and P_ψ were learned separately, and, critically, applied stage-wise to first realize the junction tree and only then reconciling attachments without feedback. Taken together, the major drawback lies in the fact that the decoder is not auto-regressive and predicted attachments do not impact the substructure choices.

Based on these observations, we propose a different factorization of $P(\mathcal{G})$ that allows us to learn the distribution over substructures and their attachments in a strongly coupled, auto-regressive manner (Eq.(2)). We will show in the next section how we can predict $\mathcal{G}_k$ and $\mathcal{A}_k$ in a hierarchical manner.

$$P(\mathcal{G}) = \prod_i P(\mathcal{G}_k, \mathcal{A}_k \mid \mathcal{G}_{<k}, \{\mathcal{A}_{i,j}\}_{1 \leq i < j < k}) \quad \mathcal{A}_k = \{\mathcal{A}_{1,k}, \cdots, \mathcal{A}_{k-1,k}\} \tag{2}$$

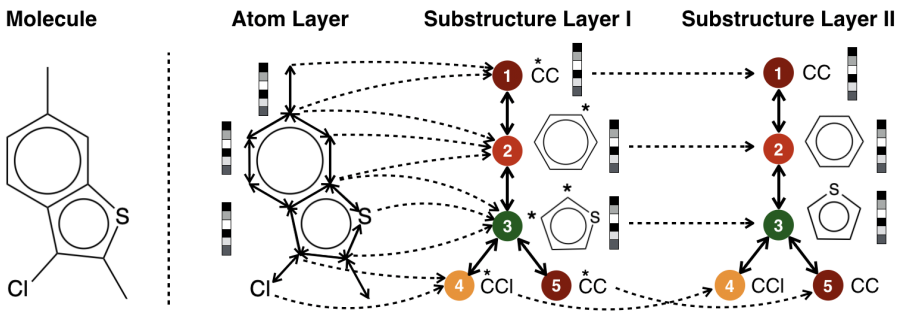

Figure 2: Hierarchical graph representation with an atom layer and two substructure layers. Dashed lines connect each atom to the substructures it belongs (some edges omitted due to space limit). Each substructure node has two labels: structure type $\mathcal{C}(\mathcal{G}_i)$ and its contextualized version $\mathcal{C}_{\mathcal{G}}(\mathcal{G}_i)$, where atoms in the intersection of $\mathcal{G}_i$ and $\mathcal{G}_{pa(i)}$ are marked with $*$. Numbers in the circle indicate the ordering between the substructure nodes. Each layer is encoded by a message passing network.

3 Hierarchical Generation of Molecular Graphs

Hierarchical Graph Representation Formally, each substructure $\mathcal{G}_i = (\mathcal{V}_i, \mathcal{E}_i)$ is a subgraph of $\mathcal{G}$ induced by atoms $\mathcal{V}_i = \{v_1^i, v_2^i, \cdots, v_m^i\}$. We extract a set of substructures $\mathcal{G}_1, \cdots, \mathcal{G}_n$ such that their union covers the entire molecular graph: $\mathcal{V} = \bigcup_i \mathcal{V}_i$ and $\mathcal{E} = \bigcup_i \mathcal{E}_i$. We consider two types of substructures: rings and bonds. To enable structure-based generation, we need to represent molecules at multiple resolutions: the connectivity between individual atoms as well as the connectivity between chemical substructures. As a result, it is insufficient to represent molecules by their molecular graphs alone. Here we propose to represent a molecule by a hierarchical graph that augments the molecular graph with additional nodes and edges to incorporate substructure information. The hierarchical graph has three components (see Figure 2):

1. **Atom layer**: The atom layer is the molecular graph $\mathcal{G}$ representing how the atoms are connected. Each atom node v is associated with label a_v indicating its atom type and charge. Each edge (u, v) in the atom layer is labeled with b_{uv} indicating its bond type.
2. **Substructure layer I**: This layer contains n nodes representing substructures $\mathcal{G}_1, \cdots, \mathcal{G}_n$. We draw an edge between $\mathcal{G}_i$ and $\mathcal{G}_j$ if $\mathcal{V}_i \cap \mathcal{V}_j \neq \emptyset$. For each substructure $\mathcal{G}_i$, we draw a directed edge from atom v to node $\mathcal{G}_i$ if $v \in \mathcal{G}_i$ so that the atom information can be propagated to this layer during graph encoding and decoding. Each substructure $\mathcal{G}_i$ is labeled with its contextualized structure type $\mathcal{C}_{\mathcal{G}}(\mathcal{G}_i)$ made by taking the subgraph $\mathcal{G}_i$ and marking all atoms with $*$ symbol that belong to the intersection between $\mathcal{G}_i$ and its parent substructure $\mathcal{G}_{pa(i)}$ ($pa(i)$ is determined by the node ordering discussed below). This label preserves more information about atom connectivity than $\mathcal{G}_i$ itself and allows us to assemble these substructures much easier during decoding.
3. **Substructure layer II**: This layer has the same layout as the previous layer but with different node labels. Each substructure $\mathcal{G}_i$ is labeled with $\mathcal{C}(\mathcal{G}_i)$ that is the subgraph $\mathcal{G}_i$ itself. This layer represents a molecule purely at the substructure level and its labels are order-independent.

Node Ordering To generate graphs in an autoregressive manner, we need to designate an ordering among the substructure nodes as there is no natural ordering among them. In particular, we select a substructure node as root and traverse the substructure layer in a depth-first order. This leads to a *ordered* list of substructures $\mathcal{G}_1, \cdots, \mathcal{G}_n$, with $\mathcal{G}_i$'s predecessor being $\mathcal{G}_{pa(i)}$. Nodes in the atom layer remain unordered. When translating from input X to target Y, we select their root nodes such that their node orderings have approximately the smallest edit distance. The algorithm for this node alignment is described in the appendix.

Tree Decomposition For efficient graph decoding, we apply Jin et al. [21]'s tree decomposition method to enforce both substructure layers to be tree-structured. This modification will mostly retain its original layout as all the rings in the molecule have been contracted into single nodes and thus the substructure layer is already very tree-like. The two substructure layers will still have the same layout after this operation. While our substructure layers are very similar to junction tree in [21], our key departure is that we integrate the substructure layers with the molecular graph as a hierarchical graph to be encoded jointly. In contrast, the junction tree is encoded separately from the molecular graph.

3.1 Hierarchical Graph Encoder

Given a molecule X, we encode its hierarchical graph $\mathcal{H}_X$ using a hierarchical message passing network with three layers to learn representations for all atoms and substructures (see Figure 2). These three message passing layers are computed sequentially as follows:

Atom Layer We first encode the atom layer of the hierarchical graph (i.e., molecular graph). We define $\boldsymbol{e}(\cdot)$ as an embedding function of discrete labels, and $N(v)$ as the set of neighbor nodes of v. During encoding, each edge (u, v) in the atom layer is associated with two hidden vectors $\boldsymbol{\nu}_{uv}$ and $\boldsymbol{\nu}_{vu}$, representing the message from u to v and vice versa. These messages are updated iteratively via LSTM [19] adapted for message passing networks [5, 12] (details are shown in the appendix):

$$\boldsymbol{\nu}_{uv}^{(t)} = \text{LSTM}_{\psi_1}\left(\boldsymbol{e}(a_u), \boldsymbol{e}(b_{uv}), \{\boldsymbol{\nu}_{wu}^{(t-1)}\}_{w \in N(u) \setminus v}\right) \tag{3}$$

where ψ_1 is the LSTM cell parameters and $\boldsymbol{\nu}_{uv}^{(t)}$ is the message computed in the t-th iteration, initialized with $\boldsymbol{\nu}_{uv}^{(0)} = \mathbf{0}$. After T iterations, we compute the atom representation $\boldsymbol{x}_u^{\mathcal{G}}$ by combining its inward messages and its atom embedding. This encodes the local environment of atom u:

$$\boldsymbol{x}_u^{\mathcal{G}} = \text{ReLU}\left(\boldsymbol{W}_1^o \boldsymbol{e}(a_u) + \boldsymbol{U}_1^o \sum\nolimits_{w \in N(u)} \boldsymbol{\nu}_{wu}^{(T)}\right) \tag{4}$$

Substructure Layer I To propagate atom level representations to this layer, we compute the input feature of substructure $\mathcal{G}_i$ based on its contextualized label $\mathcal{C}_{\mathcal{G}}(\mathcal{G}_i)$ and atom vectors $\{\boldsymbol{x}_u^{\mathcal{G}} | u \in \mathcal{G}_i\}$:

$$\boldsymbol{f}_i^{\mathcal{S}} = \text{ReLU}\left(\boldsymbol{W}_2^i \, \boldsymbol{e}(\mathcal{C}_{\mathcal{G}}(\mathcal{G}_i)) + \boldsymbol{U}_2^i \sum\nolimits_{u \in \mathcal{G}_i} \boldsymbol{x}_u^{\mathcal{G}}\right) \tag{5}$$

We run T iterations of message passing with node features $\boldsymbol{f}_i^{\mathcal{S}}$ and edge features $\boldsymbol{e}(d_{ij})$ describing the relative ordering between node i and j. Specifically, we set $d_{ij} = l$ if node i is the l-th child of node j and $d_{ij} = 0$ if $i = pa(j)$. This gives us a set of substructure representations $\{\boldsymbol{x}_i^{\mathcal{S}}\}$:

$$\boldsymbol{\mu}_{ij}^{(t)} = \text{LSTM}_{\psi_2}\left(\boldsymbol{f}_i^{\mathcal{S}}, \boldsymbol{e}(d_{ij}), \{\boldsymbol{\mu}_{ki}^{(t-1)}\}_{k \in N(i) \setminus j}\right) \quad \boldsymbol{x}_i^{\mathcal{S}} = \text{ReLU}\left(\boldsymbol{W}_2^o \boldsymbol{f}_i^{\mathcal{S}} + \boldsymbol{U}_2^o \sum\nolimits_{k \in N(i)} \boldsymbol{\mu}_{ki}^{(T)}\right) \tag{6}$$

Substructure Layer II The input feature $\boldsymbol{f}_i^{\mathcal{T}}$ in this layer is computed based on the label $\mathcal{C}(\mathcal{G}_i)$ and the output from the previous layer $\boldsymbol{x}_i^{\mathcal{S}}$. We then compute the neural messages $\boldsymbol{m}_{ij}^{(\cdot)}$ with $\boldsymbol{f}_i^{\mathcal{T}}$ as input:

$$\boldsymbol{f}_i^{\mathcal{T}} = \text{ReLU}\left(\boldsymbol{W}_3^i \, \boldsymbol{e}(\mathcal{C}(\mathcal{G}_i)) + \boldsymbol{U}_3^i \boldsymbol{x}_i^{\mathcal{S}}\right) \quad \boldsymbol{m}_{ij}^{(t)} = \text{LSTM}_{\psi_3}\left(\boldsymbol{f}_i^{\mathcal{T}}, \boldsymbol{e}(d_{ij}), \{\boldsymbol{m}_{ki}^{(t-1)}\}_{k \in N(i) \setminus j}\right) \tag{7}$$

Finally, we compute the substructure representation $\boldsymbol{x}_i^{\mathcal{T}} = \text{ReLU}\left(\boldsymbol{W}_3^o \boldsymbol{f}_i^{\mathcal{T}} + \boldsymbol{U}_3^o \sum_{k \in N(i)} \boldsymbol{m}_{ki}^{(T)}\right)$. The final outputs of our encoder are continuous vectors $\{\boldsymbol{x}_i^{\mathcal{G}}\}, \{\boldsymbol{x}_i^{\mathcal{S}}\}, \{\boldsymbol{x}_i^{\mathcal{T}}\}$ representing the molecule at multiple resolutions.

3.2 Hierarchical Graph Decoder

Our graph decoder generates a molecule Y by incrementally expanding its hierarchical graph $\mathcal{H}_Y$, following the order $\mathcal{G}_1, \cdots, \mathcal{G}_n$. In each step, the decoder needs to compute the probability of the next substructure and its connectivity to previous substructures $P(\mathcal{G}_k, \mathcal{A}_k \mid \mathcal{G}_{<k}, \{\mathcal{A}_{i,j}\}_{1 \le i < j < k})$. The key observation is that this probability can be computed efficiently since the substructure layer is a tree and each substructure will only be attached to its parent node, i.e. $\forall j \neq pa(k), j < k : \mathcal{A}_{j,k} = \emptyset$. Thus the model only need to predict which node is k's parent and the value of $\mathcal{A}_{pa(k),k}$. In other words, $P(\mathcal{G}_k, \mathcal{A}_k) = P(\mathcal{G}_k, \mathcal{A}_{pa(k),k}, pa(k)) = P(\mathcal{A}_{pa(k),k} | \mathcal{G}_k, pa(k)) P(\mathcal{G}_k | pa(k)) P(pa(k))$ (conditional variables omitted for simplicity). The probabilities are realized as follows:

1. **Topological Prediction** $P(pa(k))$: The model predicts if there will be a new substructure appended to the current node i. If so, the model expands a new node k from i in the substructure layer, otherwise it backtracks to i's parent node $pa(i)$.
2. **Substructure Prediction** $P(\mathcal{G}_k | pa(k))$: The model then predicts the content of the new substructure $\mathcal{G}_k$: its structure type $\mathcal{C}(\mathcal{G}_k)$ and its contextualized type $\mathcal{C}_{\mathcal{G}}(\mathcal{G}_k)$. The atoms marked with $*$ in $\mathcal{C}_{\mathcal{G}}(\mathcal{G}_k)$ will be attached to the parent substructure $\mathcal{G}_i$.

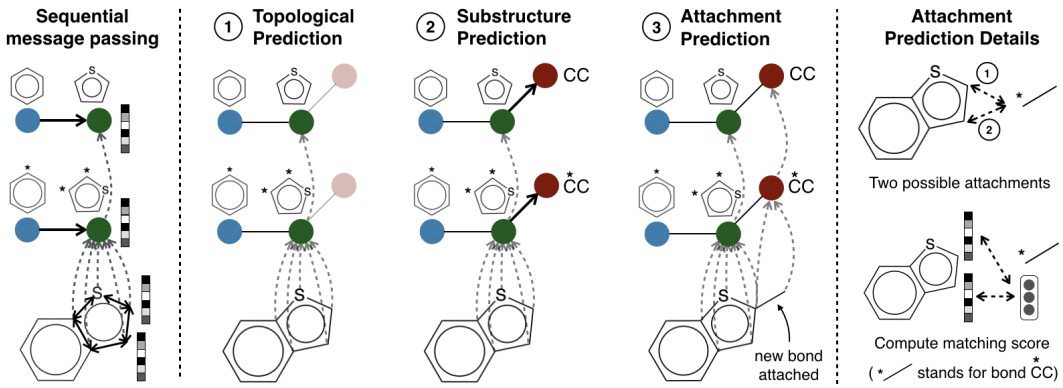

Figure 3: Illustration of the hierarchical graph decoder. **Left**: We first run message passing network over the nodes involved in the prediction. Computed messages are highlighted with arrows. **Middle**: We predict the next substructure and how it is attached to its parent substructure. **Right**: There are two ways to attach the predicted bond to its parent substructure. We compute their compatibility scores based on the representations of matching atoms.

3. **Attachment Prediction** $P(\mathcal{A}_{pa(k),k}|\mathcal{G}_k)$: Finally, the model needs to predict to which atoms in $\mathcal{G}_i$ the new substructure will be attached. This prediction only occurs when the parent node $\mathcal{G}_i$ has more than two atoms. When $\mathcal{G}_i$ is a bond, this step can be omitted because there is only one atom in $\mathcal{G}_i$ remain unattached. The other atom has been attached to $\mathcal{G}_{pa(i)}$ and cannot be used again.

Sequential Message Passing All of these predictors build on a hierarchical message passing network with the same architecture as the encoder but with a different message passing schedule. The encoder computes all the messages simultaneously as the input graph is fixed. In contrast, the graph input to the decoder is always being changed, and all the node representations need to be updated accordingly before any prediction is made. This requires us to recompute all the node vectors at every step if messages are propagated in the same way. We therefore compute messages and node vectors sequentially based on the order of nodes traversed during decoding, similar to recurrent neural networks. Specifically, suppose the decoder is currently at node i in the substructure layer. We only recompute the messages $\{\boldsymbol{\nu}_{uv}^{(\cdot)}|(u,v) \in \mathcal{G}_i\}, \boldsymbol{\mu}_{ji}^{(\cdot)}, \boldsymbol{m}_{ji}^{(\cdot)}$ for T' steps and compute node vectors $\{\boldsymbol{y}_u^{\mathcal{G}}|u \in \mathcal{G}_i\}, \boldsymbol{y}_i^{\mathcal{S}}, \boldsymbol{y}_i^{\mathcal{T}}$ correspondingly (see Figure 3). All the other messages are used based on their current value and remain unchanged. In many cases, this reduces the number of message propagation from $O(n|\mathcal{E}|T')$ to $O(|\mathcal{E}|T')$. We refer to Figure 5 in the appendix for a step-by-step illustration.

Topological Prediction The probability of backtracking is computed based on the substructure vector $\boldsymbol{y}_i^{\mathcal{T}}$ and the context vector $\boldsymbol{\alpha}_i^d$ which is a weight average of substructure representations $\{\boldsymbol{x}_j^{\mathcal{T}}\}$ of input molecule X via attention mechanism (recall that we are translating molecule X into Y).

$$\boldsymbol{p}_i = \sigma\left(\boldsymbol{V}^d \cdot \mathrm{ReLU}(\boldsymbol{W}^d \boldsymbol{y}_i^{\mathcal{T}} + \boldsymbol{U}^d \boldsymbol{\alpha}_i^d)\right) \quad \boldsymbol{\alpha}_i^d = \mathrm{attention}_{\theta_d}\left(\boldsymbol{y}_i^{\mathcal{T}}, \{\boldsymbol{x}_j^{\mathcal{T}}\}\right) \tag{8}$$

Substructure Prediction When a new node k is expanded from node i, we compute its inward message $\boldsymbol{m}_{ik}^{(\cdot)}$ and predict its substructure type in two steps. We first predict $\mathcal{C}(\mathcal{G}_k)$ as follows:

$$\boldsymbol{q}_k = \mathrm{softmax}\left(\boldsymbol{V}^l \cdot \mathrm{ReLU}(\boldsymbol{W}^l \boldsymbol{m}_{ik}^{(T')} + \boldsymbol{U}^l \boldsymbol{\alpha}_{ik}^l)\right) \quad \boldsymbol{\alpha}_{ik}^l = \mathrm{attention}_{\theta_l}\left(\boldsymbol{m}_{ik}^{(T')}, \{\boldsymbol{x}_j^{\mathcal{T}}\}\right) \tag{9}$$

The contextualized structure type $\mathcal{C}_{\mathcal{G}}(\mathcal{G}_k)$ is predicted similarly based on the message vector $\boldsymbol{m}_{ik}^{(T')}$ and context vector $\boldsymbol{\alpha}_{ik}^l$ but with different parameters. We only consider labels that are compatible with the predicted substructure $\mathcal{C}(\mathcal{G}_k)$ (i.e., $\mathcal{C}_{\mathcal{G}}(\mathcal{G}_k) = \mathcal{C}(\mathcal{G}_k)$ with $*$ ignored).

Attachment Prediction Let $\{v_1, \cdots, v_m\}$ be the atoms marked with $*$ in $\mathcal{C}_{\mathcal{G}}(\mathcal{G}_k)$. We seek to find the correct matching $\widehat{\mathcal{M}}_{ik} = \{(\widehat{u}_j, v_j)|\widehat{u}_j \in \mathcal{G}_i\}$ where v_j is attached to $\widehat{u}_j$. Note that this prediction is required only when $\mathcal{G}_i$ is a ring. Thus there exist at most $2|\mathcal{G}_i|$ different matchings as attaching points must be consecutive. Each candidate matching $\mathcal{M}_{ik} = \{(u_j, v_j)\}$ is scored based on the atom representations $\boldsymbol{y}_{u_j}^{\mathcal{G}}$ and $\boldsymbol{e}_{\mathcal{G}_k}(v_j)$ describing their local environments in the current graph and $\mathcal{G}_k$:

$$s(\mathcal{M}_{ik}) = \boldsymbol{h}_{\mathcal{M}_{ik}} \cdot \mathrm{attention}_{\theta_a}(\boldsymbol{h}_{\mathcal{M}_{ik}}, \{\boldsymbol{x}_j^{\mathcal{G}}\}) \quad \boldsymbol{h}_{\mathcal{M}_{ik}} = \sum\nolimits_j \mathrm{ReLU}(\boldsymbol{W}^a \boldsymbol{y}_{u_j}^{\mathcal{G}} + \boldsymbol{U}^a \boldsymbol{e}_{\mathcal{G}_k}(v_j)) \tag{10}$$

The scores are then normalized by softmax over all possible matchings. As the structure $\mathcal{G}_k$ is always fixed, we represent each atom v_j with an embedding vector $\boldsymbol{e}_{\mathcal{G}_k}(v_j)$ indicating its position in $\mathcal{G}_k$ instead of running another graph convolutional network on $\mathcal{G}_k$. We ensure two atoms receive the same embedding if they are isomorphic in $\mathcal{G}_k$.

3.3 Variational Graph-to-Graph Translation

We now apply our hierarchical graph encoder-decoder to the molecular translation task. The training set contains molecular pairs (X, Y) and their meta information $\boldsymbol{c}_{X,Y}$ indicating what properties of X have been improved. During testing, we can specify in $\boldsymbol{c}_{X,Y}$ what properties we want to improve and feed it to the model to adapt its translation outcome (see multi-property optimization in § 5). It is important to note that each compound X can be associated with multiple outputs Y since there are many ways to modify X to improve its properties. Following [22], we extend our model to a variational translation model that learns a mapping $\mathcal{F} : (X, \boldsymbol{z}) \rightarrow Y$. The latent vector $\boldsymbol{z}$ indicates the intended mode of translation which is sampled from a prior distribution $P(\boldsymbol{z})$ at test time.

The latent vectors are learned through variational inference [26]. Given a training example (X, Y), we compute $\boldsymbol{z}$ by first encoding molecules X and Y into their representations $\{\boldsymbol{x}_*^{\mathcal{T}}\}, \{\boldsymbol{x}_*^{\mathcal{G}}\}; \{\boldsymbol{y}_*^{\mathcal{T}}\}, \{\boldsymbol{y}_*^{\mathcal{G}}\}$ using our hierarchical encoder. Then we compute difference vectors $\boldsymbol{\delta}_{X,Y}$ that summarize the structural changes occurred from molecule X to Y at both atom and substructure level:

$$\boldsymbol{\delta}_{X,Y}^{\mathcal{T}} = \sum\nolimits_i \boldsymbol{y}_i^{\mathcal{T}} - \sum\nolimits_i \boldsymbol{x}_i^{\mathcal{T}} \qquad \boldsymbol{\delta}_{X,Y}^{\mathcal{G}} = \sum\nolimits_i \boldsymbol{y}_i^{\mathcal{G}} - \sum\nolimits_i \boldsymbol{x}_i^{\mathcal{G}} \tag{11}$$

The approximate posterior $Q(\boldsymbol{z}|X, Y)$ is modeled as a normal distribution whose mean and log variance are computed from $\boldsymbol{\delta}_{X,Y} = [\boldsymbol{\delta}_{X,Y}^{\mathcal{T}}, \boldsymbol{\delta}_{X,Y}^{\mathcal{G}}, \boldsymbol{c}_{X,Y}]$ with two separate affine layers $\boldsymbol{\mu}(\cdot)$ and $\boldsymbol{\Sigma}(\cdot)$. We sample latent code $\boldsymbol{z} = [\boldsymbol{z}^{\mathcal{T}}, \boldsymbol{z}^{\mathcal{G}}]$ from $Q(\boldsymbol{z}|X, Y)$ via reparameterization trick, and combine latent codes $\boldsymbol{z}^{\mathcal{T}}, \boldsymbol{z}^{\mathcal{G}}$ with the atom and substructure representations of X:

$$\tilde{\boldsymbol{x}}_i^{\mathcal{T}} = \mathrm{ReLU}\left(\boldsymbol{W}_e^{\mathcal{T}}[\boldsymbol{x}_i^{\mathcal{T}}, \boldsymbol{z}^{\mathcal{T}}, \boldsymbol{c}_{X,Y}]\right) \qquad \tilde{\boldsymbol{x}}_i^{\mathcal{G}} = \mathrm{ReLU}\left(\boldsymbol{W}_e^{\mathcal{G}}[\boldsymbol{x}_i^{\mathcal{G}}, \boldsymbol{z}^{\mathcal{G}}, \boldsymbol{c}_{X,Y}]\right); \tag{12}$$

The combined representation $\{\tilde{\boldsymbol{x}}_*^{\mathcal{T}}\}$ and $\{\tilde{\boldsymbol{x}}_*^{\mathcal{G}}\}$ are then fed into the decoder to generate the target molecule Y. The training objective follows a standard conditional variational autoencoder: $\mathcal{L}(X, Y) = -\mathbb{E}_{\boldsymbol{z}\sim Q}[\log P(Y|\boldsymbol{z}, X)] + \lambda_{\mathrm{KL}}\mathcal{D}_{\mathrm{KL}}[Q(\boldsymbol{z}|X, Y)||P(\boldsymbol{z})]$.

4 Related Work

Molecular Graph Generation Previous work have adopted various approaches for generating molecular graphs. Methods [13, 44, 29, 6, 15, 38, 39, 24] generate molecules based on their SMILES strings [47]. Simonovsky and Komodakis [45], De Cao and Kipf [8], Ma et al. [36] developed generative models which output the adjacency matrices and node labels of the graphs at once. Li et al. [34], Samanta et al. [41], Li et al. [32], Liu et al. [35], Assouel et al. [1] proposed generative models generating molecular graphs sequentially node by node. You et al. [51], Zhou et al. [53] adopted similar node-by-node approaches in the context of reinforcement learning. Kajino [23] developed a hypergraph grammar based method for molecule generation. Our work is most closely related to Jin et al. [21, 22] that generate molecules structure by structure, but our approach jointly predicts the structures and their connectivity with an autoregressive decoder.

Graph Encoders and Decoders Our work is related to encoder and decoder architectures for general graphs. You et al. [52], Bojchevski et al. [3], Grover et al. [14] developed graph decoders for modeling social networks. Previous work on graph encoders include convolutional networks [42, 4, 18, 37, 27, 16, 49, 28], recurrent networks [33, 5, 31] and attention based architectures [46]. Graph encoders have been widely applied to encoding molecular structures [10, 25, 12, 43, 20]. We refer to Hamilton et al. [17], Wu et al. [48] for a more comprehensive survey on graph neural networks.

Hierarchical Graph Representation Our work is closely related to [9, 50, 11] that learn to represent graphs in a hierarchical manner. Defferrard et al. [9] utilized graph coarsening approaches to represent graphs at multiple resolutions. Ying et al. [50], Gao and Ji [11] proposed to learn the graph pooling operations jointly with graph convolution to compute hierarchical representations of graphs. Our work learns hierarchical representation of molecules but our focus is on graph generation. Our method builds on a predefined graph pooling mechanism so that the vocabulary of substructures is fixed during graph decoding.

Table 1: Results on four single-property translation tasks where baselines are from Jin et al. [22]. "Div." stands for diversity. "Succ." stands for success rate. "Improve." stands for property improvement.

Method	logP (sim ≥ 0.6)		logP (sim ≥ 0.4)		QED		DRD2	
	Improve.	Div.	Improve.	Div.	Succ.	Div.	Succ.	Div.
JT-VAE	0.28 ± 0.79	-	1.03 ± 1.39	-	8.8%	-	3.4%	-
GCPN	0.79 ± 0.63	-	2.49 ± 1.30	-	9.4%	0.216	4.4%	0.152
MMPA	1.65 ± 1.44	0.329	3.29 ± 1.12	0.496	32.9%	0.236	46.4%	**0.275**
Seq2Seq	2.33 ± 1.17	0.331	3.37 ± 1.75	0.471	58.5%	0.331	75.9%	0.176
JTNN	2.33 ± 1.24	0.333	3.55 ± 1.67	0.480	59.9%	0.373	77.8%	0.156
HierG2G	$\mathbf{2.49 \pm 1.09}$	**0.381**	$\mathbf{3.81 \pm 1.52}$	**0.564**	**76.6%**	**0.477**	**85.9%**	0.192

Table 2: Results on three multi-property translation tasks. $\boldsymbol{c} = [1, *]$ means the output Y needs to be drug-like and $\boldsymbol{c} = [*, 1]$ means it needs to be DRD2-active.

Method	$\boldsymbol{c} = [1, 1]$		$\boldsymbol{c} = [1, 0]$		$\boldsymbol{c} = [0, 1]$	
	Success	Diversity	Success	Diversity	Success	Diversity
Seq2Seq	14.6%	0.113	81.5%	0.413	61.9%	0.280
JTNN	24.6%	**0.158**	80.2%	0.440	72.7%	0.271
HierG2G	**28.4%**	0.133	**84.2%**	**0.520**	**75.9%**	**0.312**

5 Experiments

We follow the experimental design by Jin et al. [22] and evaluate our translation model on their single-property optimization tasks. As molecular optimization in real-world setting often involves multiple properties, we further construct a novel multi-property optimization task where the desired property combination is fed as input to the translation process. To avoid the model from ignoring input X and translating it into arbitrary compounds, we require the molecular similarity $sim(X, Y)$ between input X and output Y to be above certain threshold δ across all the tasks. The molecular similarity is defined as the Tanimoto similarity over Morgan fingerprints [40] of two molecules.

Single-property optimization This dataset consists of four different tasks. For each task, we train and evaluate our model on their provided training and test sets. For single-property optimization, we set $\boldsymbol{c}_{X,Y} = \mathbf{0}$ during training and testing as previous works do not utilize the meta information.

- **LogP optimization**: The penalized logP score [29] measures the solubility and synthetic accessibility of a compound. In this task, the model needs to translate input compound X into compound Y such that logP$(Y) >$ logP(X). We experiment with two similarity thresholds $\delta = \{0.4, 0.6\}$.
- **QED optimization**: The QED score [2] quantifies a compound's drug-likeness. In this task, the model is required to translate molecules with QED scores from the lower range $[0.7, 0.8]$ into the higher range $[0.9, 1.0]$. The similarity constraint is $sim(X, Y) \geq 0.4$.
- **DRD2 optimization**: This task involves the optimization of a compound's biological activity against dopamine type 2 receptor (DRD2). The model needs to translate inactive compounds (DRD2 < 0.05) into active compounds (DRD2 $\geq$ 0.5), where the bioactivity is assessed by a property prediction model from Olivecrona et al. [38]. The similarity constraint is $sim(X, Y) \geq 0.4$.

Multi-property optimization This new task requires the model to translate input X to output Y satisfying multiple constraints over its QED and DRD2 scores. It is important to note that different users may be interested in different combination of these constraints. For example, when DRD2 is considered as an on-target, the output molecule needs to be both drug-like and active to DRD2. In other applications, DRD2 may become an off-target and the model must generate compounds inactive to DRD2 to avoid side effects. In order to handle different property combinations, we encode the desired criteria as meta information $\boldsymbol{c}$ and feed it to the decoder (directed translation).

In particular, our model is trained to handle three different criteria over target molecule Y: 1) Y is drug-like and DRD2-active; 2) Y is drug-like but DRD2-inactive; 3) Y is not drug-like but DRD2-

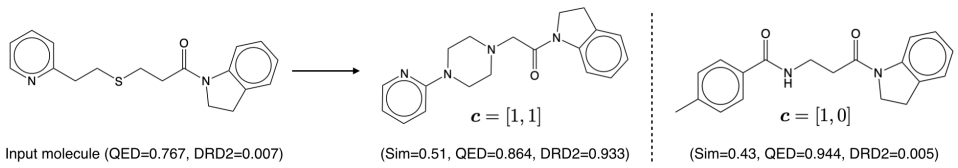

Figure 4: Illustration of multi-property translation. Our model generates different molecules when the translation criteria changes. When $\boldsymbol{c} = [1, 1]$, the model indeed generates a compound with high QED and DRD2 scores. When $\boldsymbol{c} = [1, 0]$, the model predicts another compound inactive to DRD2.

active. We define a molecule Y as drug-like if QED$(Y) \geq 0.85$ and DRD2-active if its predicted bioactivity DRD2$(Y) \geq 0.5$. These settings share the same similarity constraint $sim(X, Y) \geq 0.4$. Our training set contains 120K molecular pairs and the test set has 780 compounds with QED < 0.85 and DRD2 < 0.5. For each pair (X, Y), we set its meta information $\boldsymbol{c}_{X,Y} = (\mathbb{I}[\text{QED}(Y) \geq 0.85], \mathbb{I}[\text{DRD2}(Y) \geq 0.5])$ to indicate its satisfied constraints. During testing, we translate each compound with $\boldsymbol{c} = [1, 1], [1, 0], [0, 1]$ for each of the three translation criteria. We note that $\boldsymbol{c} = [1, 1]$ is the most challenging as there are only 6K pairs with target Y being both drug-like and DRD2-active (5% of the training set). The model must learn to transfer the knowledge from other pairs with $\boldsymbol{c}_{X,Y} = [1, 0], [0, 1]$ to this criteria to achieve good performance.

Evaluation Metrics Our evaluation metrics include translation accuracy and diversity. Following Jin et al. [22], each test molecule X_i is translated $K = 20$ times with different latent codes sampled from the prior distribution. On the logP optimization, we select compound Y_i as the final translation of X_i that gives the highest property improvement and satisfies $sim(X_i, Y_i) \geq \delta$. We then report the average property improvement $\frac{1}{\mathcal{D}} \sum_i \text{logP}(Y_i) - \text{logP}(X_i)$ over test set $\mathcal{D}$. For other tasks, we report the translation success rate. A compound is successfully translated if one of its K translation candidates satisfies all the similarity and property constraints of the task. To measure the diversity, for each molecule we compute the average pairwise Tanimoto distance between all its successfully translated compounds. Here the Tanimoto distance is defined as $dist(X, Y) = 1 - sim(X, Y)$.

Baselines Our baselines include generative models (JT-VAE [21]), reinforcement learning (GCPN [51]) and translation based approaches (MMPA, Seq2Seq and JTNN). MMPA [7] is a template based translation method. Seq2Seq is a variational sequence-to-sequence translation model that generates molecules by their SMILES strings. JTNN [22] is a variational junction tree encoder-decoder that generates molecular graphs structure by structure, but its decoder is not fully autoregressive. Similar to our method, the latent code $\boldsymbol{z}$ in Seq2Seq and JTNN is learned from the difference vector $\boldsymbol{\delta}_{X,Y}$ computed from their internal molecular representations. For multi-property translation, we feed the meta information $\boldsymbol{c}_{X,Y}$ as input to Seq2Seq and JTNN similarly to our model.

Results Table 1 summarizes the results on the single-property optimization tasks. Our model greatly outperforms previous state-of-the-art JTNN in terms of both translation accuracy (e.g., 76.6% versus 59.9% on the QED task) and diversity (e.g., 0.564 versus 0.480 on the logP task). This clearly demonstrates the benefits of our autoregressive decoder which can learn more expressive mappings. Our model also outperforms Seq2Seq though its decoder is autoregressive. This shows the benefit of learning hierarchical representations of molecules at both atom and substructure level.

Results on the multi-property optimization are shown in Table 2. Across three translation criteria, our model significantly outperforms other models in terms of translation accuracy. For all models, the success rate on $\boldsymbol{c} = [1, 1]$ is the lowest because it has the strongest constraints and there are only 6K training pairs satisfying this criteria. In fact, training our model on the 6K examples only achieved 12.4% translation success rate as compared to 28.4% when trained with other pairs. This shows our directed translation model can transfer the knowledge from other pairs with $\boldsymbol{c}_{X,Y} = [1, 0], [0, 1]$ to this criteria. Figure 4 illustrates how the input criteria affects the translation outcome.

6 Conclusion

In this paper, we developed a hierarchical graph-to-graph translation model that generates molecular graphs using chemical substructures as building blocks. In contrast to previous work, our model is fully autoregressive and learns coherent multi-resolution representations. The experimental results show that our method significantly outperforms previous models under various settings.

References

[1] Rim Assouel, Mohamed Ahmed, Marwin H Segler, Amir Saffari, and Yoshua Bengio. Defactor: Differentiable edge factorization-based probabilistic graph generation. *arXiv preprint arXiv:1811.09766*, 2018.

[2] G Richard Bickerton, Gaia V Paolini, Jérémy Besnard, Sorel Muresan, and Andrew L Hopkins. Quantifying the chemical beauty of drugs. *Nature chemistry*, 4(2):90, 2012.

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

[4] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*, 2013.

[5] Hanjun Dai, Bo Dai, and Le Song. Discriminative embeddings of latent variable models for structured data. In *International Conference on Machine Learning*, pages 2702–2711, 2016.

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

[7] Andrew Dalke, Jerome Hert, and Christian Kramer. mmpdb: An open-source matched molecular pair platform for large multiproperty data sets. *Journal of chemical information and modeling*, 2018.

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

[9] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems*, pages 3844–3852, 2016.

[10] David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. Convolutional networks on graphs for learning molecular fingerprints. In *Advances in neural information processing systems*, pages 2224–2232, 2015.

[11] Hongyang Gao and Shuiwang Ji. Graph u-net. *International Conference on Machine Learning*, 2019.

[12] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. *arXiv preprint arXiv:1704.01212*, 2017.

[13] Rafael Gómez-Bombarelli, Jennifer N Wei, David Duvenaud, José Miguel Hernández-Lobato, Benjamín Sánchez-Lengeling, Dennis Sheberla, Jorge Aguilera-Iparraguirre, Timothy D Hirzel, Ryan P Adams, and Alán Aspuru-Guzik. Automatic chemical design using a data-driven continuous representation of molecules. *ACS Central Science*, 2018. doi: 10.1021/acscentsci.7b00572.

[14] Aditya Grover, Aaron Zweig, and Stefano Ermon. Graphite: Iterative generative modeling of graphs. *arXiv preprint arXiv:1803.10459*, 2018.

[15] Gabriel Lima Guimaraes, Benjamin Sanchez-Lengeling, Pedro Luis Cunha Farias, and Alán Aspuru-Guzik. Objective-reinforced generative adversarial networks (organ) for sequence generation models. *arXiv preprint arXiv:1705.10843*, 2017.

[16] William L Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. *arXiv preprint arXiv:1706.02216*, 2017.

[17] William L Hamilton, Rex Ying, and Jure Leskovec. Representation learning on graphs: Methods and applications. *arXiv preprint arXiv:1709.05584*, 2017.

[18] Mikael Henaff, Joan Bruna, and Yann LeCun. Deep convolutional networks on graph-structured data. *arXiv preprint arXiv:1506.05163*, 2015.

[19] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

[20] Wengong Jin, Connor Coley, Regina Barzilay, and Tommi Jaakkola. Predicting organic reaction outcomes with weisfeiler-lehman network. In *Advances in Neural Information Processing Systems*, pages 2604–2613, 2017.

[21] Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Junction tree variational autoencoder for molecular graph generation. *International Conference on Machine Learning*, 2018.

[22] Wengong Jin, Kevin Yang, Regina Barzilay, and Tommi Jaakkola. Learning multimodal graph-to-graph translation for molecular optimization. *International Conference on Learning Representations*, 2019.

[23] Hiroshi Kajino. Molecular hypergraph grammar with its application to molecular optimization. *arXiv preprint arXiv:1809.02745*, 2018.

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

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

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

[27] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *International Conference on Learning Representations*, 2017.

[28] Risi Kondor, Hy Truong Son, Horace Pan, Brandon Anderson, and Shubhendu Trivedi. Covariant compositional networks for learning graphs. *arXiv preprint arXiv:1801.02144*, 2018.

[29] Matt J Kusner, Brooks Paige, and José Miguel Hernández-Lobato. Grammar variational autoencoder. *arXiv preprint arXiv:1703.01925*, 2017.

[30] Greg Landrum. Rdkit: Open-source cheminformatics. *Online). http://www. rdkit. org. Accessed*, 3(04):2012, 2006.

[31] Tao Lei, Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Deriving neural architectures from sequence and graph kernels. *International Conference on Machine Learning*, 2017.

[32] Yibo Li, Liangren Zhang, and Zhenming Liu. Multi-objective de novo drug design with conditional graph generative model. *arXiv preprint arXiv:1801.07299*, 2018.

[33] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. Gated graph sequence neural networks. *arXiv preprint arXiv:1511.05493*, 2015.

[34] Yujia Li, Oriol Vinyals, Chris Dyer, Razvan Pascanu, and Peter Battaglia. Learning deep generative models of graphs. *arXiv preprint arXiv:1803.03324*, 2018.

[35] Qi Liu, Miltiadis Allamanis, Marc Brockschmidt, and Alexander L Gaunt. Constrained graph variational autoencoders for molecule design. *Neural Information Processing Systems*, 2018.

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

[37] Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. Learning convolutional neural networks for graphs. In *International Conference on Machine Learning*, pages 2014–2023, 2016.

[38] Marcus Olivecrona, Thomas Blaschke, Ola Engkvist, and Hongming Chen. Molecular de-novo design through deep reinforcement learning. *Journal of cheminformatics*, 9(1):48, 2017.

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

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

[41] Bidisha Samanta, Abir De, Gourhari Jana, Pratim Kumar Chattaraj, Niloy Ganguly, and Manuel Gomez-Rodriguez. Nevae: A deep generative model for molecular graphs. *arXiv preprint arXiv:1802.05283*, 2018.

[42] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2009.

[43] Kristof Schütt, Pieter-Jan Kindermans, Huziel Enoc Sauceda Felix, Stefan Chmiela, Alexandre Tkatchenko, and Klaus-Robert Müller. Schnet: A continuous-filter convolutional neural network for modeling quantum interactions. In *Advances in Neural Information Processing Systems*, pages 992–1002, 2017.

[44] Marwin HS Segler, Thierry Kogej, Christian Tyrchan, and Mark P Waller. Generating focussed molecule libraries for drug discovery with recurrent neural networks. *arXiv preprint arXiv:1701.01329*, 2017.

[45] Martin Simonovsky and Nikos Komodakis. Graphvae: Towards generation of small graphs using variational autoencoders. *arXiv preprint arXiv:1802.03480*, 2018.

[46] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

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

[48] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S Yu. A comprehensive survey on graph neural networks. *arXiv preprint arXiv:1901.00596*, 2019.

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

[50] Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, Will Hamilton, and Jure Leskovec. Hierarchical graph representation learning with differentiable pooling. In *Advances in Neural Information Processing Systems*, pages 4800–4810, 2018.

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

[52] Jiaxuan You, Rex Ying, Xiang Ren, William L Hamilton, and Jure Leskovec. Graphrnn: A deep generative model for graphs. *arXiv preprint arXiv:1802.08773*, 2018.

[53] Zhenpeng Zhou, Steven Kearnes, Li Li, Richard N Zare, and Patrick Riley. Optimization of molecules via deep reinforcement learning. *arXiv preprint arXiv:1810.08678*, 2018.

A Network Architecture Details

Node Ordering Here we describe the procedure for aligning the node orderings between two molecules X and Y. First, we assign an global index to each substructure type $C(\mathcal{G}_k)$ which will be used to remove ambiguities during substructure layer traversal. Let $\{u_1, \cdots, u_n\}, \{v_1, \cdots, v_m\}$ be the set of leaf nodes (with degree one) in the substructure layer of hierarchical graphs $\mathcal{H}_X$ and $\mathcal{H}_Y$. Starting from each node u_i, we traverse the substructure layer of X in its depth-first order. This gives us an ordered list of substructure nodes $\mathcal{O}_X(u_i)$ for molecule X. When encountering multiple children during this traversal, we visit them in the order of their substructure type indices.

To find the best alignment, we enumerate all possible pairs of root nodes (u_i, v_j) of the two molecules. For each pair, we compute the edit distance between $\mathcal{O}_X(u_i)$ and $\mathcal{O}_Y(v_j)$ and select the pair yielding the smallest edit distance. While this requires us to compute $n \cdot m$ edit distances, this approach is tractable because most of the pairs have $n \cdot m < 50$. All the node orderings are preprocessed before training, and it takes roughly 15 minutes to align a training set with 100K pairs with single thread.

LSTM Cell The LSTM cell used to compute neural messages in our hierarchical encoder and decoder is defined as follows:

$$\begin{aligned}
\boldsymbol{s}_{uv} &= \sum\nolimits_{w \in N(u) \setminus v} \boldsymbol{h}_{wu}^{(t)} && (13)\\
\boldsymbol{i}_{uv} &= \sigma(\boldsymbol{W}_{\psi}^{z} [\boldsymbol{f}_u, \boldsymbol{f}_{uv}, \boldsymbol{s}_{uv}] + \boldsymbol{b}^{z}) && (14)\\
\boldsymbol{o}_{uv} &= \sigma(\boldsymbol{W}_{\psi}^{o} [\boldsymbol{f}_u, \boldsymbol{f}_{uv}, \boldsymbol{s}_{uv}] + \boldsymbol{b}^{o}) && (15)\\
\boldsymbol{r}_{wu} &= \sigma(\boldsymbol{W}_{\psi}^{r} \left[\boldsymbol{f}_u, \boldsymbol{f}_{uv}, \boldsymbol{h}_{wu}^{(t)}\right] + \boldsymbol{b}^{r}) && (16)\\
\tilde{\boldsymbol{c}}_{uv} &= \tanh\left(\boldsymbol{W}_{\psi} [\boldsymbol{f}_u, \boldsymbol{f}_{uv}, \boldsymbol{s}_{uv}] + \boldsymbol{b}\right) && (17)\\
\boldsymbol{c}_{uv}^{(t+1)} &= \boldsymbol{i}_{uv} \odot \tilde{\boldsymbol{c}}_{uv} + \sum_{w \in N(u) \setminus v} \boldsymbol{r}_{wu} \odot \boldsymbol{c}_{wu}^{(t)} && (18)\\
\boldsymbol{h}_{uv}^{(t+1)} &= \boldsymbol{o}_{uv} \odot \tanh\left(\boldsymbol{c}_{uv}^{(t+1)}\right) && (19)
\end{aligned}$$

For notational simplicity, throughout this paper we denote the above message update rule as $\boldsymbol{h}_{uv}^{(t+1)} = \mathrm{LSTM}_{\psi}\left(\boldsymbol{f}_u, \boldsymbol{f}_{uv}, \{\boldsymbol{h}_{wu}^{(t)}\}_{w \in N(u) \setminus v}\right)$, where $\psi = \{\boldsymbol{W}_{\psi}^{z}, \boldsymbol{W}_{\psi}^{o}, \boldsymbol{W}_{\psi}^{r}, \boldsymbol{W}_{\psi}\}$. During the message update, the LSTM cell also updates its internal cell state from $\boldsymbol{c}_{uv}^{(t)}$ to $\boldsymbol{c}_{uv}^{(t+1)}$.

Attention Layer Our attention layer is a bilinear attention function with parameter $\theta = \{\boldsymbol{A}_{\theta}\}$:

$$\mathrm{attention}_{\theta}(\boldsymbol{v}, \{\boldsymbol{x}_i\}) = \frac{\exp(\boldsymbol{v}^T \boldsymbol{A}_{\theta} \boldsymbol{x}_i)}{\sum_j \exp(\boldsymbol{v}^T \boldsymbol{A}_{\theta} \boldsymbol{x}_j)} \cdot \boldsymbol{x}_i \tag{20}$$

Sequential Message Passing As illustrated in Figure 5, we compute messages and node vectors sequentially based on the order of nodes traversed during decoding. Suppose the decoder is currently at node i in the substructure layer. We reinitialize the messages $\boldsymbol{\mu}_{ji}^{(\cdot)}, \boldsymbol{m}_{ji}^{(\cdot)}, \{\boldsymbol{\nu}_{uv}^{(\cdot)} | (u, v) \in \mathcal{G}_i\}$ as zero and unroll these messages for T' iterations. The other messages are reused based on their current value. Then we compute nodes vectors $\{\boldsymbol{y}_u^{\mathcal{G}} | u \in \mathcal{G}_i\}, \boldsymbol{y}_i^{\mathcal{S}}, \boldsymbol{y}_i^{\mathcal{T}}$ for predictions.

Substructure Prediction We predict the next substructure in two steps, by first predicting its structure type $\mathcal{C}(\mathcal{G}_k)$ based on Eq.(9) and then its contextualized substructure type $\mathcal{C}_{\mathcal{G}}(\mathcal{G}_k)$ that are compatible with $\mathcal{C}(\mathcal{G}_k)$:

$$\mathrm{softmax}\left(\boldsymbol{V}^c \cdot \mathrm{ReLU}(\boldsymbol{W}^c \boldsymbol{m}_{ik}^{(T')} + \boldsymbol{U}^c \boldsymbol{\alpha}_{ik}^{l})\right) \tag{21}$$

The context vector $\boldsymbol{\alpha}_{ik}^{l}$ is computed in Eq.(9) and shared across these two predictions. As the vocabulary of structure types is much smaller than contextualized structure types, this two-layer hierarchical prediction can achieve higher accuracy than directly predicting the contextualized type.

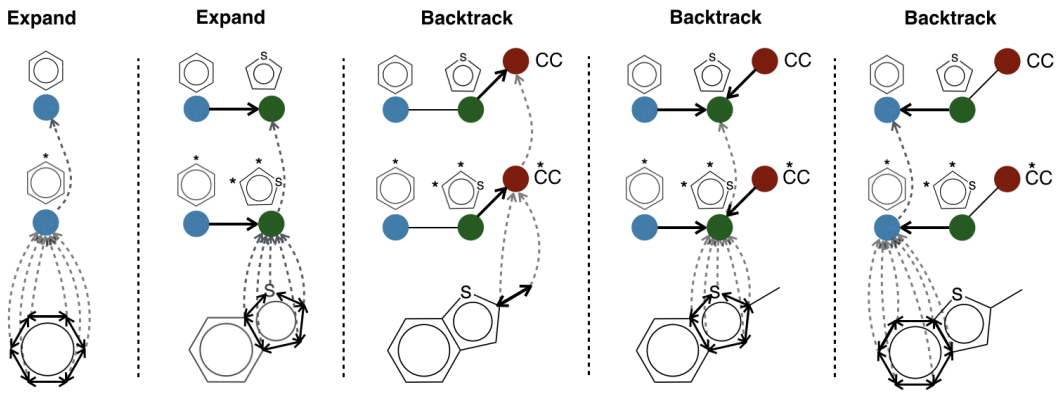

Figure 5: Illustration of the sequential message passing network used by our decoder. The topological prediction is shown on top of each step. The computed messages are highlighted with arrows.

B Experimental Details

Substructure Vocabulary Our substructure vocabulary is automatically constructed by extracting rings and bonds from the molecules using RDKit [30]. This gives us two sets of vocabularies, one over the substructure labels and the other cover the contextualized labels. Across different datasets, the vocabulary of contextualized substructures contains around 1000 to 1800 different labels. The non-contextualized vocabulary contains around 300 to 400 different labels.

Training Details We elaborate on the hyperparameters used in our experiments. The hidden layer dimension is 270 and the embedding layer dimension is 200. We set the latent code dimension $|z| = 8$ and KL regularization weight $\lambda_{KL} = 0.3$. We run $T = 20$ iterations of message passing in each layer of the encoder. As our graph decoder updates the messages in a sequential manner, we run $T' = 3$ iterations of message passing at each decoding step. We train our model with Adam optimizer with initial learning rate 0.001 and anneal the learning rate by 0.9 after every epoch. Our models are trained on single Titan X GPU with a 16-core CPU and 128G RAM. For all the tasks, we evaluate our model with a single run.

Data The single-property optimization datasets are directly downloaded from the github link provided in Jin et al. [22]. The training set size of logP (sim=0.6 and 0.4), QED and DRD2 optimization tasks are 75K, 99K, 88K and 34K respectively. We constructed the multi-property optimization by combining the training set of QED and DRD2 optimization task from Jin et al. [22]. The test set contains all the compounds in the test set of DRD2 optimization task with QED < 0.85. The training and test set is attached as part of the supplementary material.