Hierarchical Generation of Molecular Graphs using Structural Motifs

Wengong Jin [1] **Regina Barzilay** [1] **Tommi Jaakkola** [1]

Abstract

Graph generation techniques are increasingly being adopted for drug discovery. Previous graph generation approaches have utilized relatively small molecular building blocks such as atoms or simple cycles, limiting their effectiveness to smaller molecules. Indeed, as we demonstrate, their performance degrades significantly for larger molecules. In this paper, we propose a new hierarchical graph encoder-decoder that employs significantly larger and more flexible graph motifs as basic building blocks. Our encoder produces a multi-resolution representation for each molecule in a fine-to-coarse fashion, from atoms to connected motifs. Each level integrates the encoding of constituents below with the graph at that level. Our autoregressive coarse-to-fine decoder adds one motif at a time, interleaving the decision of selecting a new motif with the process of resolving its attachments to the emerging molecule. We evaluate our model on multiple molecule generation tasks, including polymers, and show that our model significantly outperforms previous state-of-the-art baselines.

1. Introduction

Deep learning models for molecule property prediction and molecule generation are improving at a fast pace. Work to date has adopted primarily two types of building blocks for representing and building molecules: atom-by-atom strategies (Li et al., 2018; You et al., 2018a; Liu et al., 2018), or substructure based (either rings or bonds) (Jin et al., 2018; 2019). While these methods have been successful for small molecules, their performance degrades significantly for larger molecules such as polymers (see Figure 1). The failure is likely due to many generation steps required to realize larger molecules and the associated challenges with gradients across the iterative steps.

[1]MIT CSAIL. Correspondence to: Wengong Jin <wengong@csail.mit.edu>.

Proceedings of the 37th International Conference on Machine Learning, Vienna, Austria, PMLR 119, 2020. Copyright 2020 by the author(s).

Large molecules such as polymers exhibit clear hierarchical structure, being built from repeated structural motifs. We hypothesize that explicitly incorporating such motifs as building blocks in the generation process can significantly improve reconstruction and generation accuracy, as already illustrated in Figure 1. While different substructures as building blocks were considered in previous work (Jin et al., 2018), their approach could not scale to larger motifs. Indeed, their decoding process required each substructure neighborhood to be assembled in one go, making it combinatorially challenging to handle large components with many possible attachment points.

In this paper, we propose a motif-based hierarchical encoder-decoder for graph generation. The motifs themselves are extracted separately at the outset from frequently occurring substructures, regardless of size. During generation, molecules are built step by step by attaching motifs, large or small, to the emerging molecule. The decoder operates hierarchically, in a coarse-to-fine manner, and makes three key consecutive predictions in each pass: new motif selection, which part of it attaches, and the points of contact with the current molecule. These decisions are highly coupled and naturally modeled auto-regressively. Moreover, each decision is directly guided by the information explicated in the associated layer of the mirroring hierarchical encoder. The feed-forward fine-to-coarse encoding performs iterative graph convolutions at each level, conditioned on the results from layer below.

The proposed model is evaluated on various tasks ranging from polymer generative modeling to graph translation for molecule property optimization.[1] Our baselines include state-of-the-art graph generation methods (You et al., 2018a; Liu et al., 2018; Jin et al., 2019). On polymer generation, our model achieved state-of-the art results under various metrics, outperforming the best baselines with 20% absolute improvement in reconstruction accuracy. On graph translation tasks, our model outperformed all the baselines, yielding 3.3% and 8.1% improvement on QED and DRD2 optimization tasks. We further conduct ablation studies to validate the advantage of using larger motifs and the proposed hierarchical architecture.

[1]https://github.com/wengong-jin/hgraph2graph

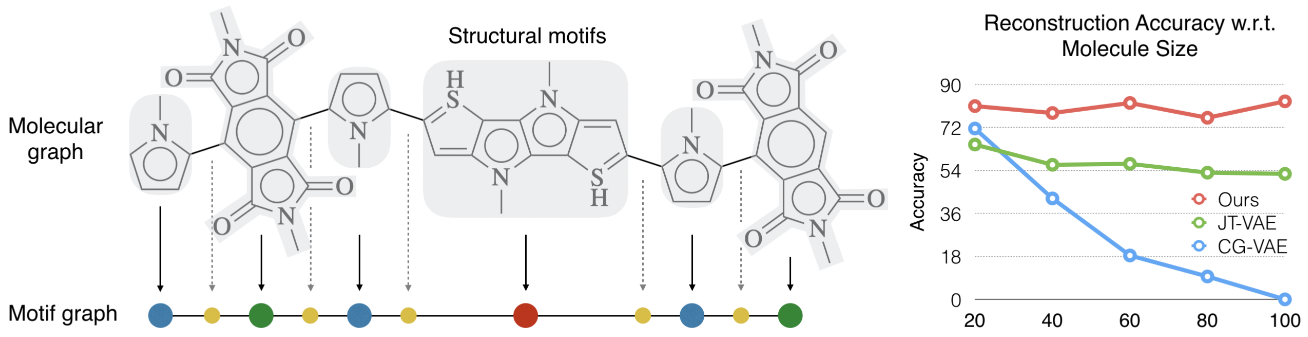

Figure 1. **Left**: Illustration of structural motifs in polymers. **Right**: Reconstruction accuracy for polymers with various sizes (number of atoms). Notably, the atom-based generative model CG-VAE (Liu et al., 2018) fails to reconstruct molecules over 80 atoms. In contrast, the proposed model maintains high accuracy for large molecules by utilizing motifs as building blocks for generation (red curve).

2. Background and Motivation

Molecules are represented as graphs $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with atoms $\mathcal{V}$ as nodes and bonds $\mathcal{E}$ as edges. Graphs are challenging objects to generate, especially for larger molecules such as polymers. For the polymer dataset used in our experiment, there are thousands of molecules with more than 80 atoms. To illustrate the challenge, we tested two state-of-the-art variational autoencoders (Liu et al., 2018; Jin et al., 2018) on this dataset and found these models often fail to reconstruct molecules from their latent embedding (see Figure 1).

The reason of this failure is that these methods generate molecules based on small building blocks. In terms of autoregressive models, previous work on molecular graph generation can be roughly divided in two categories:[2]

- Atom-based methods (Li et al., 2018; You et al., 2018a; Liu et al., 2018) generate molecules atom by atom.
- Substructure-based methods (Jin et al., 2018; 2019) generates molecules based on small substructures restricted to rings and bonds (often no more than six atoms).

As the building blocks are typically small, it requires many decoding steps for current models to reconstruct polymers. Therefore they are prone to make errors when generating large molecules. On the other hand, many of these molecules consist of structural motifs beyond simple substructures. The number of decoding steps can be significantly reduced if graphs are generated motif by motif. As shown in Figure 1, our motif-based method achieves a much higher reconstruction accuracy.

Motivation for New Architecture Current substructure-based method (Jin et al., 2018) requires a combinatorial enumeration to assemble substructures whose time complexity is exponential to substructure size. Their enumeration algorithm assumes the substructures to be of certain types (single cycles or bonds). In practice, their method often fails when handling rings with more than 10 atoms (e.g., memory error). Unlike substructures, motifs are typically much larger and can have flexible structures (see Figure 1). As a result, this method cannot be directly extended to utilize motifs in practice.

To this end, we propose a hierarchical encoder-decoder for graph generation. Our decoder allows arbitrary types of motifs and can assemble them efficiently without combinatorial explosion. Our encoder learns a hierarchical representation that allows the decoding process to depend on both coarse-grained motif and fine-grained atom connectivity.

2.1. Motif Extraction

We define a motif $\mathcal{S}_i = (\mathcal{V}_i, \mathcal{E}_i)$ as a subgraph of molecule $\mathcal{G}$ induced by atoms in $\mathcal{V}_i$ and bonds in $\mathcal{E}_i$. Given a molecule, we extract its motifs $\mathcal{S}_1, \cdots, \mathcal{S}_n$ such that their union covers the entire molecular graph: $\mathcal{V} = \bigcup_i \mathcal{V}_i$ and $\mathcal{E} = \bigcup_i \mathcal{E}_i$. To extract motifs, we decompose a molecule $\mathcal{G}$ into disconnected fragments by breaking all the bridge bonds that will not violate chemical validity (illustrations in the appendix).

1. Find all the *bridge* bonds $(u, v) \in \mathcal{E}$, where both u and v have degree $\Delta_u, \Delta_v \geq 2$ and either u or v is part of a ring. Detach all the bridge bonds from its neighbors.
2. Now the graph $\mathcal{G}$ becomes a set of disconnected subgraphs $\mathcal{G}_1, \cdots, \mathcal{G}_N$. Select $\mathcal{G}_i$ as motif in $\mathcal{G}$ if its occurrence in the training set is more than $\Delta = 100$.
3. If $\mathcal{G}_i$ is not selected as motif, further decompose it into rings and bonds and select them as motif in $\mathcal{G}$.

We apply the above procedure to all the molecules in the training set and construct a vocabulary of motifs $V_{\mathcal{S}}$. In the following section, we will describe how we encode and decode molecules using the extracted motifs.

[2]We restrict our discussion to molecule generation. You et al. (2018b); Liao et al. (2019) developed generative models for other types of graphs such as social networks. Their current implementations do not support the prediction of node and edge attributes and cannot be directly applied to molecules. Thus their methods are not tested here.

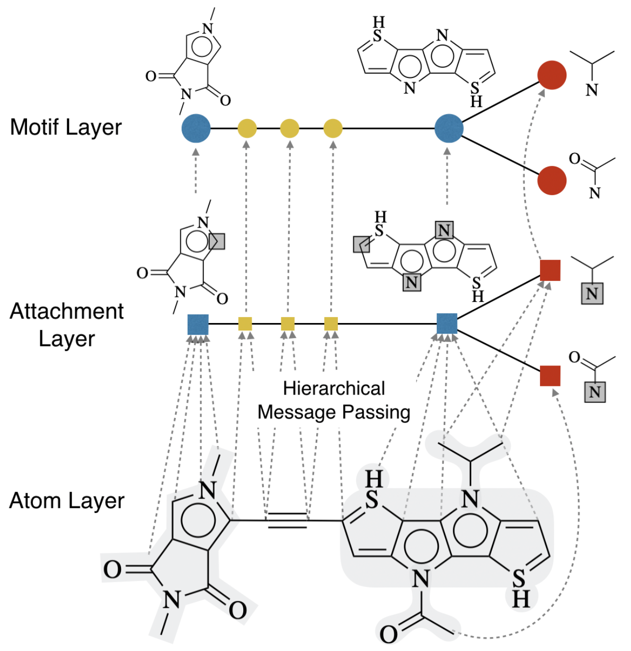

Figure 2. Hierarchical graph encoder. Dashed arrows connect each atom to the motifs it belongs. In the attachment layer, each node $\mathcal{A}_i$ is a particular attachment configuration of motif $\mathcal{S}_i$. The atoms in the intersection between each motif and its neighbors are highlighted in faded block.

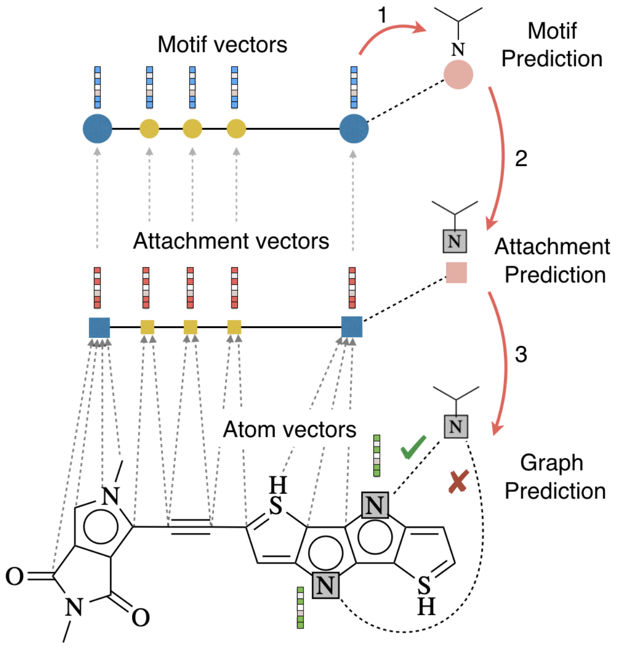

Figure 3. Hierarchical graph decoder. In each step, the decoder first runs hierarchical message passing to compute motif, attachment and atom vectors. Then it performs motif and attachment prediction for the next motif node. Finally, it decides how the new motif should be attached to the current graph via graph prediction.

3. Hierarchical Graph Generation

Our approach extends the variational autoencoder (Kingma & Welling, 2013) to molecular graphs by introducing a hierarchical decoder and a matching encoder. In our framework, the probability of a graph $\mathcal{G}$ is modeled as a joint distribution over structural motifs $\mathcal{S}_1, \cdots, \mathcal{S}_n$ constituting $\mathcal{G}$, together with their attachments $\mathcal{A}_1, \cdots, \mathcal{A}_n$. Each attachment $\mathcal{A}_i = \{v_j \mid v_j \in \bigcup_k \mathcal{S}_i \cap \mathcal{S}_k\}$ indicates the intersecting atoms between $\mathcal{S}_i$ and its neighbor motifs. To capture complex dependencies involved in the joint distribution of motifs and their attachments, we propose an auto-regressive factorization of $P(\mathcal{G})$:

$$P(\mathcal{G}) = \int_{\boldsymbol{z}} P(\boldsymbol{z}) \prod\nolimits_k P(\mathcal{S}_k, \mathcal{A}_k \mid \mathcal{S}_{<k}, \mathcal{A}_{<k}, \boldsymbol{z}) d\boldsymbol{z} \quad (1)$$

As illustrated in Figure 3, in each generation step, our decoder adds a new motif $\mathcal{S}_k$ (*motif prediction*) and its attachment configuration $\mathcal{A}_k$ (*attachment prediction*). Then it decides how the new motif should be attached to the current graph (*graph prediction*).

To support the above hierarchical generation, we need to design a matching encoder representing molecules at multiple resolutions in order to provide necessary information for each decoding step. Therefore, we propose to represent a molecule $\mathcal{G}$ by a hierarchical graph $\mathcal{H}_\mathcal{G}$ with three layers (see Figure 2):

1. **Motif layer**: This layer represents how the motifs are coarsely connected in the graph. This layer provides essential information for the motif prediction in the decoding process. Specifically, this layer contains n nodes $\mathcal{S}_1, \cdots, \mathcal{S}_n$ and m edges $\{(\mathcal{S}_i, \mathcal{S}_j) \mid \mathcal{S}_i \cap \mathcal{S}_j \neq \emptyset\}$ for all intersecting motifs $\mathcal{S}_i, \mathcal{S}_j$. This layer is tree-structured due to our way of constructing motifs.

2. **Attachment layer**: This layer encodes the connectivity between motifs at a fine-grained level. Each node $\mathcal{A}_i = (\mathcal{S}_i, \{v_j\})$ in this layer represents a particular attachment configuration of motif $\mathcal{S}_i$, where $\{v_j\}$ are atoms in the intersection between $\mathcal{S}_i$ and one of its neighbor motifs (see Figure 2). This layer provides crucial information for the attachment prediction step during decoding, which helps reducing the space of candidate attachments between $\mathcal{S}_i$ and its neighbor motifs. Just like the motif vocabulary $V_\mathcal{S}$, all the attachment configurations of $\mathcal{S}_i$ form a motif-specific vocabulary $V_\mathcal{A}(\mathcal{S}_i)$, which is computed from the training set.[3]

3. **Atom layer**: The atom layer is the molecular graph $\mathcal{G}$ representing how its atoms are connected. Each atom node v is associated with a label a_v indicating its atom type and charge. Each edge (u, v) in the atom layer is

[3]In our experiments, the average size of attachment vocabulary $|V_\mathcal{A}(\mathcal{S}_i)| \leq 10$ and the size of motif vocabulary $|V_\mathcal{S}| < 500$.

labeled with b_{uv} indicating its bond type. This layer provides necessary information for the graph prediction step during decoding.

We further introduce edges that connect the atoms and motifs between different layers in order to propagate information in between. In particular, we draw a directed edge from atom v in the atom layer to node $\mathcal{A}_i$ in the attachment layer if $v \in \mathcal{S}_i$. We also draw edges from $\mathcal{A}_i$ to $\mathcal{S}_i$ in the motif layer. This gives us the hierarchical graph $\mathcal{H}_\mathcal{G}$ for molecule $\mathcal{G}$, which will be encoded by a hierarchical message passing network (MPN). During encoding, each node $\mathcal{S}_i$ is represented as a one-hot encoding in the motif vocabulary $V_\mathcal{S}$. Likewise, each node $\mathcal{A}_i$ is represented as a one-hot encoding in the attachment vocabulary $V_\mathcal{A}(\mathcal{S}_i)$.

3.1. Hierarchical Graph Encoder

Our encoder contains three MPNs that encode each of the three layers in the hierarchical graph. For simplicity, we denote the MPN encoding process as $\mathrm{MPN}_\psi(\cdot)$ with parameter ψ, and denote $\mathrm{MLP}(\boldsymbol{x}, \boldsymbol{y})$ as a multi-layer neural network whose input is the concatenation of $\boldsymbol{x}$ and $\boldsymbol{y}$. The details of MPN architecture is listed in the appendix.

Atom Layer MPN We first encode the atom layer of $\mathcal{H}_\mathcal{G}$ (denoted as $\mathcal{H}_\mathcal{G}^g$). The inputs to this MPN are the embedding vectors $\{\boldsymbol{e}(a_u)\}, \{\boldsymbol{e}(b_{uv})\}$ of all the atoms and bonds in $\mathcal{G}$. During encoding, the network propagates the message vectors between different atoms for T iterations and then outputs the atom representation $\boldsymbol{h}_v$ for each atom v:

$$\boldsymbol{c}_\mathcal{G}^g = \{\boldsymbol{h}_v\} = \mathrm{MPN}_{\psi_1}\left(\mathcal{H}_\mathcal{G}^g, \{\boldsymbol{e}(a_u)\}, \{\boldsymbol{e}(b_{uv})\}\right) \quad (2)$$

Attachment Layer MPN The input feature of each node $\mathcal{A}_i$ in the attachment layer $\mathcal{H}_\mathcal{G}^a$ is an concatenation of the embedding $\boldsymbol{e}(\mathcal{A}_i)$ and the sum of its atom vectors $\{\boldsymbol{h}_v \mid v \in \mathcal{S}_i\}$:

$$\boldsymbol{f}_{\mathcal{A}_i} = \mathrm{MLP}\left(\boldsymbol{e}(\mathcal{A}_i), \sum\nolimits_{v \in \mathcal{S}_i} \boldsymbol{h}_v\right) \quad (3)$$

The input feature for each edge $(\mathcal{A}_i, \mathcal{A}_j)$ in this layer is an embedding vector $\boldsymbol{e}(d_{ij})$, where d_{ij} describes the relative ordering between node $\mathcal{A}_i$ and $\mathcal{A}_j$ during decoding. Specifically, we set $d_{ij} = k$ if node $\mathcal{A}_i$ is the k-th child of node $\mathcal{A}_j$ and $d_{ij} = 0$ if $\mathcal{A}_i$ is the parent. We then run T iterations of message passing over $\mathcal{H}_\mathcal{G}^a$ to compute the motif representations:

$$\boldsymbol{c}_\mathcal{G}^a = \{\boldsymbol{h}_{\mathcal{A}_i}\} = \mathrm{MPN}_{\psi_2}\left(\mathcal{H}_\mathcal{G}^a, \{\boldsymbol{f}_{\mathcal{A}_i}\}, \{\boldsymbol{e}(d_{ij})\}\right) \quad (4)$$

Motif Layer MPN Similarly, the input feature of node $\mathcal{S}_i$ in this layer is computed as the concatenation of embedding $\boldsymbol{e}(\mathcal{S}_i)$ and the node vector $\boldsymbol{h}_{\mathcal{A}_i}$ from the previous layer. Finally, we run message passing over the motif layer $\mathcal{H}_\mathcal{G}^s$ to obtain the motif representations:

$$\boldsymbol{f}_{\mathcal{S}_i} = \mathrm{MLP}\left(\boldsymbol{e}(\mathcal{S}_i), \boldsymbol{h}_{\mathcal{A}_i}\right) \quad (5)$$

$$\boldsymbol{c}_\mathcal{G}^s = \{\boldsymbol{h}_{\mathcal{S}_i}\} = \mathrm{MPN}_{\psi_3}\left(\mathcal{H}_\mathcal{G}^s, \{\boldsymbol{f}_{\mathcal{S}_i}\}, \{\boldsymbol{e}(d_{ij})\}\right) \quad (6)$$

Finally, we represent a molecule $\mathcal{G}$ by a latent vector $\boldsymbol{z}_\mathcal{G}$ sampled through reparameterization trick with mean $\boldsymbol{\mu}(\boldsymbol{h}_{\mathcal{S}_1})$ and log variance $\boldsymbol{\Sigma}(\boldsymbol{h}_{\mathcal{S}_1})$:

$$\boldsymbol{z}_\mathcal{G} = \boldsymbol{\mu}(\boldsymbol{h}_{\mathcal{S}_1}) + \exp(\boldsymbol{\Sigma}(\boldsymbol{h}_{\mathcal{S}_1})) \cdot \epsilon; \quad \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \quad (7)$$

where $\mathcal{S}_1$ is the root motif (i.e., the first motif to be generated during reconstruction).

3.2. Hierarchical Graph Decoder

As illustrated in Figure 3, our graph decoder generates a molecule $\mathcal{G}$ by incrementally expanding its hierarchical graph. In t^{th} generation step, we first use the same hierarchical MPN architecture to encode all the motifs and atoms in $\mathcal{H}_\mathcal{G}^{(t)}$, the (partial) hierarchical graph generated till step t. This gives us motif vectors $\boldsymbol{h}_{\mathcal{S}_k}$ and atom vectors $\boldsymbol{h}_{v_j}$ for the existing motifs and atoms.

During decoding, the model maintains a set of frontier nodes $\mathcal{F}$ where each node $\mathcal{S}_k \in \mathcal{F}$ is a motif that still has neighbors to be generated. $\mathcal{F}$ is implemented as a stack because motifs are generated in their depth-first order. Suppose $\mathcal{S}_k$ is at the top of stack $\mathcal{F}$ in step t, the model makes the following predictions conditioned on latent representation $\boldsymbol{z}_\mathcal{G}$:

1. **Motif Prediction**: The model predicts the next motif $\mathcal{S}_t$ to be attached to $\mathcal{S}_k$. This is cast as a classification task over the motif vocabulary $V_\mathcal{S}$:

$$\boldsymbol{p}_{\mathcal{S}_t} = \mathrm{softmax}(\mathrm{MLP}(\boldsymbol{h}_{\mathcal{S}_k}, \boldsymbol{z}_\mathcal{G})) \quad (8)$$

2. **Attachment Prediction**: Now the model needs to predict the attachment configuration $\mathcal{A}_t$ of motif $\mathcal{S}_t$ (i.e., what atoms $v_j \in \mathcal{S}_t$ belong to the intersection of $\mathcal{S}_t$ and its neighbor motifs). This is also cast as a classification task over the attachment vocabulary $V_\mathcal{A}(\mathcal{S}_t)$:

$$\boldsymbol{p}_{\mathcal{A}_t} = \mathrm{softmax}(\mathrm{MLP}(\boldsymbol{h}_{\mathcal{S}_k}, \boldsymbol{z}_\mathcal{G})) \quad (9)$$

   This prediction step is crucial because it significantly reduces the space of possible attachments between $\mathcal{S}_t$ and its neighbor motifs.

3. **Graph Prediction**: Finally, the model must decide how $\mathcal{S}_t$ should be attached to $\mathcal{S}_k$. The attachment between $\mathcal{S}_t$ and $\mathcal{S}_k$ is defined as atom pairs $\mathcal{M}_{tk} = \{(u_j, v_j) \mid u_j \in \mathcal{A}_k, v_j \in \mathcal{A}_t\}$ where atom u_j and v_j are attached together. The probability of a candidate attachment M is computed based on the atom vectors $\boldsymbol{h}_{u_j}$ and $\boldsymbol{h}_{v_j}$:

$$\boldsymbol{p}_M = \mathrm{softmax}\left(\boldsymbol{h}_M \cdot \boldsymbol{z}_\mathcal{G}\right) \quad (10)$$

$$\boldsymbol{h}_M = \sum\nolimits_j \mathrm{MLP}(\boldsymbol{h}_{u_j}, \boldsymbol{h}_{v_j}) \quad (11)$$

The number of possible attachments are limited because the number of attaching atoms between two motifs is small and the attaching points must be consecutive.[4]

The above three predictions together give an autoregressive factorization of the distribution over the next motif and its attachment. Each of the three decoding steps depends on the outcome of previous step, and predicted attachments will in turn affect the prediction of subsequent motifs.

Training During training, we apply teacher forcing to the above generation process, where the generation order is determined by a depth-first traversal over the ground truth molecule. Given a training set of molecules, we seek to minimize the negative ELBO:

$$-\mathbb{E}_{\boldsymbol{z}\sim Q}[\log P(\mathcal{G}|\boldsymbol{z})] + \lambda_{\text{KL}}\mathcal{D}_{\text{KL}}[Q(\boldsymbol{z}|\mathcal{G})||P(\boldsymbol{z})] \quad (12)$$

3.3. Extension to Graph-to-Graph Translation

The proposed architecture can be naturally extended to graph-to-graph translation (Jin et al., 2019) for molecular optimization, which seeks to modify compounds in order to improve their biochemical properties. Given a corpus of molecular pairs $\{(X, Y)\}$, where Y is a structural analog of X with better chemical properties, the model is trained to translate an input molecular graph into its better form. In this case, we seek to learn a translation model $P(Y|X)$ parameterized by our encoder-decoder architecture. We also introduce attention layers into our model, which is crucial for translation performance (Bahdanau et al., 2014).

Training In graph translation, a compound X can be associated with multiple outputs Y since there are many ways to modify X to improve its properties. In order to generate diverse outputs, we follow previous work (Zhu et al., 2017; Jin et al., 2019) and incorporate latent variables $\boldsymbol{z}$ to the translation model:

$$P(Y|X) = \int_{\boldsymbol{z}} P(Y|X, \boldsymbol{z})P(\boldsymbol{z})d\boldsymbol{z} \quad (13)$$

where the latent vector $\boldsymbol{z}$ indicates the intended mode of translation, sampled from a prior $P(\boldsymbol{z})$ during testing.

The model is trained as a conditional variational autoencoder. Given a training example (X, Y), we sample $\boldsymbol{z}$ from the approximate posterior $Q(\boldsymbol{z}|X, Y) = \mathcal{N}(\boldsymbol{\mu}_{X,Y}, \boldsymbol{\sigma}_{X,Y})$. To compute $Q(\boldsymbol{z}|X, Y)$, we first encode X and Y into their representations $\boldsymbol{c}_X$ and $\boldsymbol{c}_Y$ and then compute difference vector $\boldsymbol{\delta}_{X,Y}$ that summarizes the structural changes from molecule X to Y at both atom and motif level:

$$\boldsymbol{\delta}^s_{X,Y} = \sum \boldsymbol{c}^s_Y - \sum \boldsymbol{c}^s_X \quad \boldsymbol{\delta}^g_{X,Y} = \sum \boldsymbol{c}^g_Y - \sum \boldsymbol{c}^g_X$$

Finally, we compute $[\boldsymbol{\mu}_{X,Y}, \boldsymbol{\sigma}_{X,Y}] = \text{MLP}(\boldsymbol{\delta}^{\mathcal{S}}_{X,Y}, \boldsymbol{\delta}^{\mathcal{G}}_{X,Y})$ and sample $\boldsymbol{z}$ using reparameterization trick. The latent code $\boldsymbol{z}$ is passed to the decoder along with the input representation $\boldsymbol{c}_X$ to reconstruct output Y. The training objective is to minimize negative ELBO similar to Eq.(12).

Attention For graph translation, the input molecule X is embedded by our hierarchical encoder into a set of vectors $\boldsymbol{c}_X = \boldsymbol{c}^s_X \cup \boldsymbol{c}^a_X \cup \boldsymbol{c}^g_X$, representing the molecule at multiple resolutions. These vectors are fed into the decoder through attention mechanisms (Luong et al., 2015). Specifically, we modify the motif prediction (Eq. 8) into

$$\begin{aligned} \boldsymbol{p}_{\mathcal{S}_t} &= \text{softmax}(\text{MLP}(\boldsymbol{h}_{\mathcal{S}_k}, \boldsymbol{\alpha}^s_k, \boldsymbol{z})) & (14)\\ \boldsymbol{\alpha}^s_k &= \text{attention}\left(\boldsymbol{h}_{\mathcal{S}_k}, \boldsymbol{c}^s_X\right) & (15) \end{aligned}$$

where $\text{attention}(\boldsymbol{h}_*, \boldsymbol{c}^s_X)$ is a bilinear attention over vectors $\boldsymbol{c}^s_X$ with query vector $\boldsymbol{h}_{\mathcal{S}_k}$. The attachment prediction (Eq. 9) is modified similarly with its attention over $\boldsymbol{c}^a_X$. The graph prediction (Eq. 10) is modified into

$$\begin{aligned} \boldsymbol{p}_M &= \text{softmax}\left(\boldsymbol{h}_M \cdot \text{attention}(\boldsymbol{h}_M, \boldsymbol{c}^g_X)\right) & (16)\\ \boldsymbol{h}_M &= \sum\nolimits_j \text{MLP}(\boldsymbol{h}_{u_j}, \boldsymbol{h}_{v_j}, \boldsymbol{z}) & (17) \end{aligned}$$

4. Experiments

We evaluate our method on two application tasks. The first task is polymer generative modeling. This experiment validates our argument in section 2 that our model is advantageous when the molecules have large sizes. The second task is graph-to-graph translation for small molecules. Here we show the proposed architecture also brings benefits to small molecules compared to previous state-of-the-art graph generation methods.

4.1. Polymer Generative Modeling

Dataset Our method is evaluated on the polymer dataset from St. John et al. (2019), which contains 86K polymers in total (after removing duplicates). The dataset is divided into 76K, 5K and 5K for training, validation and testing. Using our motif extraction, we collected 436 different motifs (examples shown in Figure 4). On average, each motif has 5.24 different attachment configurations. The distribution of motif size and their frequencies are reported in Figure 5.

Evaluation Metrics Our evaluation effort measures various aspects of molecule generation proposed in Kusner et al. (2017); Polykovskiy et al. (2018). Besides basic metrics like chemical validity and diversity, we compare distributional statistics between generated and real compounds. A good generative model should generate molecules which present similar aggregate statistics to real compounds. Our metrics include (with details shown in the appendix):

- **Reconstruction accuracy**: We measure how often the model can completely reconstruct a given molecule from its latent embedding $\boldsymbol{z}$. The reconstruction accuracy is computed over 5K compounds in the test set.

[4]In our experiments, the number of possible attachments are usually less than 20 for polymers and small molecules.

Table 1. Results on polymer generative modeling. The first row reports the oracle performance using real data as generated samples. The last row (small motif) is a variant of our model where we restrict the motif vocabulary to contain only single rings and bonds (similar to JT-VAE). "Recon." means reconstruction accuracy; "Div." means diversity; SNN means nearest neighbor similarity; "Frag / Scaf" means fragment and scaffold similarity. Except property statistics, all metrics are the higher the better.

Method	Reconstruction / Sample Quality (↑)				Property Statistics (↓)				Structural Statistics (↑)		
	Recon.	Valid	Unique	Div.	logP	SA	QED	MW	SNN	Frag.	Scaf.
Real data	-	100%	100%	0.823	0.094	6.7e-5	1.7e-5	82.3	0.706	0.995	0.462
SMILES	21.5%	93.1%	**97.3%**	0.821	1.471	0.011	**5.4e-4**	4963	0.704	0.981	0.385
CG-VAE	42.4%	**100%**	96.2%	**0.879**	3.958	2.600	0.0030	3944	0.204	0.372	0.001
JT-VAE	58.5%	**100%**	94.1%	0.864	2.645	0.157	0.0075	10867	0.522	0.925	0.297
HierVAE	**79.9%**	**100%**	97.0%	0.817	**0.525**	**0.007**	5.7e-4	**1928**	**0.708**	**0.984**	**0.390**
· Small motif	71.0%	**100%**	97.2%	0.835	0.872	0.042	0.0019	5320	0.575	0.949	0.191

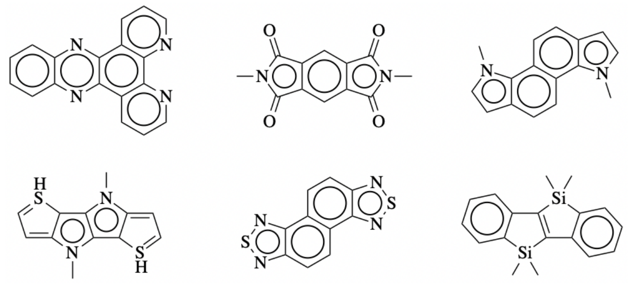

Figure 4. Examples of motif structures utilized by our model. These motifs consist of multiple rings and bonds, which are substantially more complex than previous methods (Jin et al., 2018).

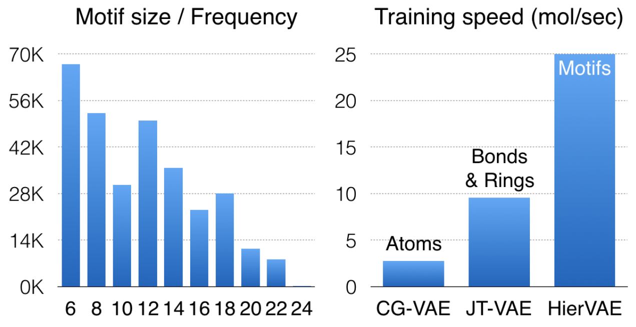

Figure 5. **Left**: Histogram of motif frequencies with respect to their sizes (i.e., number of atoms). **Right**: Training speed comparison between our method and baselines (on the same hardware).

- **Validity**: Percentage of chemically valid compounds.
- **Uniqueness**: Percentage of unique compounds.
- **Diversity**: We compute the pairwise molecular distance among generated compounds. The molecular distance $\text{dist}(X, Y)$ is defined as the Tanimoto distance over Morgan fingerprints (Rogers & Hahn, 2010) of two molecules.
- **Property statistics**: We compare property statistics between generated molecules and real data. Our properties include *partition coefficient* (logP), *synthetic accessibility* (SA), *drug-likeness* (QED) and *molecular weight* (MW). To quantitatively evaluate the distance between two distributions, we compute Frechet distance between property distributions of molecules in the generated and test sets (Polykovskiy et al., 2018).
- **Structural statistics**: We also compute structural statistics between generated molecules and real data. *Nearest neighbor similarity* (SNN) is the average similarity of generated molecules to the nearest molecule from the test set. *Fragment similarity* (Frag) and *scaffold similarity* (Scaf) are cosine distances between vectors of fragment or scaffold frequencies of the generated and the test set.

Baselines We compare our method against three state-of-the-art variational autoencoders for molecular graphs. SMILES VAE (Gómez-Bombarelli et al., 2018) is a sequence to sequence VAE that generates molecules based on their SMILES strings (Weininger, 1988). CG-VAE (Liu et al., 2018) is a graph-based VAE generating molecules atom by atom. JT-VAE (Jin et al., 2018) is also a graph-based VAE generating molecules based on simple substructures restricted to rings and bonds. Finally, we report the oracle performance of distributional statistics by using real molecules in the training set as our generated samples.

4.1.1. Results

The performance of different methods are summarized in Table 1, Our method (HierVAE) significantly outperforms all previous methods in terms of reconstruction accuracy (79.9% vs 58.5%). This validates the advantage of utilizing large structural motifs, which reduces the number of generation steps. In terms of distributional statistics, our method achieves state-of-the-art results on logP (0.525 vs 1.471), molecular weight Frechet distance (1928 vs 4863) and all the structural similarity metrics. Since our model requires fewer generation steps, our training speed is much faster than other graph-based methods (see Figure 5).

Ablation Study To validate the importance of utilizing large structural motifs, we further experiment a variant of our model ($\mathrm{small\ motif}$), which keeps the same architecture but replaces the large structural motifs with basic substructures such as rings and bonds (with less than ten atoms). As shown in Table 1, its performance is significantly worse than our full model even though it builds on the same hierarchical architecture.

4.2. Graph-to-Graph Translation

We follow the experimental design by Jin et al. (2019) and evaluate our model on their graph-to-graph translation tasks. Following their setup, we require the molecular similarity between X and output Y to be above certain threshold $\mathrm{sim}(X, Y) \geq \delta$ at test time. This is to prevent the model from ignoring input X and translating it into arbitrary compound. Here the molecular similarity is defined as $\mathrm{sim}(X, Y) = 1 - \mathrm{dist}(X, Y)$.

Dataset The dataset consists of four property optimization tasks. In each task, we train and evaluate our model on their provided training and test sets.

- **LogP**: The penalized logP score (Kusner et al., 2017) measures the solubility and synthetic accessibility of a compound. In this task, the model needs to translate input X into output Y such that $\mathrm{logP}(Y) > \mathrm{logP}(X)$. We experiment with two similarity thresholds $\delta = \{0.4, 0.6\}$.
- **QED**: The QED score (Bickerton et al., 2012) 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 $\mathrm{sim}(X, Y) \geq 0.4$.
- **DRD2**: This task involves the optimization of a compound's biological activity against dopamine type 2 receptor (DRD2). The model needs to translate inactive compounds ($p < 0.05$) into active compounds ($p \geq 0.5$), where the bioactivity is assessed by a property prediction model from Olivecrona et al. (2017). The similarity constraint is $\mathrm{sim}(X, Y) \geq 0.4$.

Evaluation Metrics Our evaluation metrics include translation accuracy and diversity. 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 $\mathrm{sim}(X_i, Y_i) \geq \delta$. We then report the average property improvement $\frac{1}{\mathcal{D}} \sum_i \mathrm{logP}(Y_i) - \mathrm{logP}(X_i)$ 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.

Baselines We compare our method against the baselines including GCPN (You et al., 2018a), MMPA (Dalke et al., 2018) and translation based methods Seq2Seq and JTNN (Jin et al., 2019). Seq2Seq is a sequence-to-sequence model that generates molecules by their SMILES strings. JTNN is a graph-to-graph architecture that generates molecules structure by structure, but its decoder is not fully autoregressive.

To make a direct comparison possible between our method and atom-based generation, we further developed an atom-based translation model (AtomG2G) as baseline. It makes three predictions in each generation step. First, it predicts whether the decoding process has completed (no more new atoms). If not, it creates a new atom a_t and predicts its atom type. Lastly, it predicts the bond type between a_t and other atoms autoregressively to fully capture edge dependencies (You et al., 2018b). The encoder of AtomG2G encodes only the atom-layer graph and the decoder attention only sees the atom vectors $\boldsymbol{c}_X^{\mathcal{G}}$. All translation models are trained under the same variational objective. Details of baseline architectures are in the appendix.

4.2.1. Results

As shown in Table 2, our model (HierG2G) achieves the new state-of-the-art on the four translation tasks. In particular, our model significantly outperforms JTNN in both translation accuracy (e.g., 76.9% versus 59.9% on the QED task) and output diversity (e.g., 0.564 versus 0.480 on the logP task). While both methods generate molecules by structures, our decoder is autoregressive which can learn more expressive mappings. In addition, our model runs 6.3 times faster than JTNN during decoding. Our model also outperforms AtomG2G on three datasets, with over 10% improvement on the DRD2 task. This shows the advantage of our hierarchical model.

Ablation Study To understand the importance of different architecture choices, we report ablation studies over the QED and DRD2 tasks in Table 3. We first replace our hierarchical decoder with the atom-based decoder of AtomG2G to see how much the motif-based decoding benefits us. We keep the same hierarchical encoder but modified the input of the decoder attention to include both atom and motif vectors. Using this setup, the model performance decreases by 0.8% and 10.9% on the two tasks. We suspect the DRD2 task benefits more from motif-based decoding because biological target binding often depends on the presence of specific functional groups.

Our second experiment reduces the number of hierarchies in our encoder and decoder MPN, while keeping the same hierarchical decoding process. When the top motif layer is removed, the translation accuracy drops slightly by 0.8%

Table 2. Results on graph translation tasks from Jin et al. (2019). We report average improvement for continuous properties (logP), and success rate for binary properties (e.g., DRD2).

Method	logP (sim ≥ 0.6)		logP (sim ≥ 0.4)		Drug likeness		DRD2	
	Improvement	Diversity	Improvement	Diversity	Success	Diversity	Success	Diversity
JT-VAE	0.28 ± 0.79	-	1.03 ± 1.39	-	8.8%	-	3.4%	-
CG-VAE	0.25 ± 0.74	-	0.61 ± 1.09	-	4.8%	-	2.3%	-
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
AtomG2G	2.41 ± 1.19	0.379	$\mathbf{3.98 \pm 1.54}$	0.563	73.6%	0.421	75.8%	0.128
HierG2G	$\mathbf{2.49 \pm 1.09}$	**0.381**	$\mathbf{3.98 \pm 1.46}$	**0.564**	**76.9%**	**0.477**	**85.9%**	0.192

Table 3. Ablation study: the importance of hierarchical graph encoding, LSTM MPN architecture and structure-based decoding.

Method	QED	DRD2
HierG2G	**76.9%**	**85.9%**
· atom-based decoder	76.1%	75.0%
· two-layer encoder	75.8%	83.5%
· one-layer encoder	67.8%	74.1%

and 2.4%. When we further remove the attachment layer (*one-layer encoder*), the performance degrades significantly on both datasets. This is because all the motif information is lost and the model needs to infer what motifs are and how motif layers are constructed for each molecule. This shows the importance of the hierarchical representation.

5. Related Work

Graph Generation Previous work have adopted various approaches for generating molecular graphs. Gómez-Bombarelli et al. (2018); Segler et al. (2017); Kusner et al. (2017); Dai et al. (2018); Guimaraes et al. (2017); Olivecrona et al. (2017); Popova et al. (2018); Kang & Cho (2018) generated molecules based on their SMILES strings (Weininger, 1988). Simonovsky & Komodakis (2018); De Cao & Kipf (2018); Ma et al. (2018) developed generative models which output the adjacency matrices and node labels of the graphs at once. You et al. (2018b); Li et al. (2018); Samanta et al. (2018); Liu et al. (2018); Zhou et al. (2018) proposed generative models which decode molecules sequentially node by node. Seff et al. (2019) developed a edit-based model which generates molecules based on insertions and deletions.

Our model is closely related to Liao et al. (2019) which generate graphs one block of nodes and edges at a time. While their encoder operates on original graphs, our encoder operates on multiple hierarchies and learns multi-resolution representations of input graphs. Our work is also closely related to Jin et al. (2018; 2019) that generate molecules based on substructures. Their decoder first generates a junction tree with substructures as nodes, and then predicts how the substructures should be attached to each other. Their substructure attachment process involves combinatorial enumeration and therefore their model cannot scale to substructures more complex than simple rings and bonds. In contrast, our model allows the motif to have flexible structures.

Graph Encoders Graph neural networks have been extensively studied for graph encoding (Scarselli et al., 2009; Bruna et al., 2013; Li et al., 2015; Niepert et al., 2016; Kipf & Welling, 2017; Hamilton et al., 2017; Lei et al., 2017; Velickovic et al., 2017; Xu et al., 2018). Our method is related to graph encoders for molecules (Duvenaud et al., 2015; Kearnes et al., 2016; Dai et al., 2016; Gilmer et al., 2017; Schütt et al., 2017). Different to these approaches, our method represents molecules as hierarchical graphs spanning from atom-level to motif-level graphs.

Our work is most closely related to (Defferrard et al., 2016; Ying et al., 2018; Gao & Ji, 2019) that learn to represent graphs in a hierarchical manner. In particular, Defferrard et al. (2016) utilized graph coarsening algorithms to construct multiple layers of graph hierarchy and Ying et al. (2018); Gao & Ji (2019) proposed to learn the graph hierarchy jointly with the encoding process. Despite some differences, all of these methods learns the hierarchy for regression or classification tasks. In contrast, our hierarchy is constructed for efficient graph generation.

6. Conclusion

In this paper, we developed a hierarchical encoder-decoder architecture generating molecular graphs using structural motifs as building blocks. The experimental results show our model outperforms prior atom and substructure based methods in both small molecule and polymer domains.

Acknowledgements

We would like to thank Jiaming Luo, Adam Yala, Benson Chen, Adam Fisch, Octavian Ganea, Yujia Bao and Rachel Wu for their insightful comments. We also thank the anonymous reviewers for their helpful feedback. This work was supported by MLPDS, DARPA AMD project and J-Clinic.

References

Bahdanau, D., Cho, K., and Bengio, Y. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

Bickerton, G. R., Paolini, G. V., Besnard, J., Muresan, S., and Hopkins, A. L. Quantifying the chemical beauty of drugs. *Nature chemistry*, 4(2):90, 2012.

Bruna, J., Zaremba, W., Szlam, A., and LeCun, Y. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*, 2013.

Dai, H., Dai, B., and Song, L. Discriminative embeddings of latent variable models for structured data. In *International Conference on Machine Learning*, pp. 2702–2711, 2016.

Dai, H., Tian, Y., Dai, B., Skiena, S., and Song, L. Syntax-directed variational autoencoder for structured data. *arXiv preprint arXiv:1802.08786*, 2018.

Dalke, A., Hert, J., and Kramer, C. mmpdb: An open-source matched molecular pair platform for large multiproperty data sets. *Journal of chemical information and modeling*, 2018.

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

Defferrard, M., Bresson, X., and Vandergheynst, P. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems*, pp. 3844–3852, 2016.

Duvenaud, D. K., Maclaurin, D., Iparraguirre, J., Bombarell, R., Hirzel, T., Aspuru-Guzik, A., and Adams, R. P. Convolutional networks on graphs for learning molecular fingerprints. In *Advances in neural information processing systems*, pp. 2224–2232, 2015.

Gao, H. and Ji, S. Graph u-net. *International Conference on Machine Learning*, 2019.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. Neural message passing for quantum chemistry. *arXiv preprint arXiv:1704.01212*, 2017.

Gómez-Bombarelli, R., Wei, J. N., Duvenaud, D., Hernández-Lobato, J. M., Sánchez-Lengeling, B., Sheberla, D., Aguilera-Iparraguirre, J., Hirzel, T. D., Adams, R. P., and Aspuru-Guzik, A. Automatic chemical design using a data-driven continuous representation of molecules. *ACS Central Science*, 2018. doi: 10.1021/acscentsci.7b00572.

Guimaraes, G. L., Sanchez-Lengeling, B., Farias, P. L. C., and Aspuru-Guzik, A. Objective-reinforced generative adversarial networks (organ) for sequence generation models. *arXiv preprint arXiv:1705.10843*, 2017.

Hamilton, W. L., Ying, R., and Leskovec, J. Inductive representation learning on large graphs. *arXiv preprint arXiv:1706.02216*, 2017.

Jin, W., Barzilay, R., and Jaakkola, T. Junction tree variational autoencoder for molecular graph generation. *International Conference on Machine Learning*, 2018.

Jin, W., Yang, K., Barzilay, R., and Jaakkola, T. Learning multimodal graph-to-graph translation for molecular optimization. *International Conference on Learning Representations*, 2019.

Kang, S. and Cho, K. Conditional molecular design with deep generative models. *Journal of chemical information and modeling*, 59(1):43–52, 2018.

Kearnes, S., McCloskey, K., Berndl, M., Pande, V., and Riley, P. Molecular graph convolutions: moving beyond fingerprints. *Journal of computer-aided molecular design*, 30(8):595–608, 2016.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. *International Conference on Learning Representations*, 2017.

Kusner, M. J., Paige, B., and Hernández-Lobato, J. M. Grammar variational autoencoder. *arXiv preprint arXiv:1703.01925*, 2017.

Lei, T., Jin, W., Barzilay, R., and Jaakkola, T. Deriving neural architectures from sequence and graph kernels. *International Conference on Machine Learning*, 2017.

Li, Y., Tarlow, D., Brockschmidt, M., and Zemel, R. Gated graph sequence neural networks. *arXiv preprint arXiv:1511.05493*, 2015.

Li, Y., Vinyals, O., Dyer, C., Pascanu, R., and Battaglia, P. Learning deep generative models of graphs. *arXiv preprint arXiv:1803.03324*, 2018.

Liao, R., Li, Y., Song, Y., Wang, S., Hamilton, W., Duvenaud, D. K., Urtasun, R., and Zemel, R. Efficient graph generation with graph recurrent attention networks. In *Advances in Neural Information Processing Systems*, pp. 4257–4267, 2019.

Liu, Q., Allamanis, M., Brockschmidt, M., and Gaunt, A. L. Constrained graph variational autoencoders for molecule design. *Neural Information Processing Systems*, 2018.

Luong, M.-T., Pham, H., and Manning, C. D. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015.

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

Niepert, M., Ahmed, M., and Kutzkov, K. Learning convolutional neural networks for graphs. In *International Conference on Machine Learning*, pp. 2014–2023, 2016.

Olivecrona, M., Blaschke, T., Engkvist, O., and Chen, H. Molecular de-novo design through deep reinforcement learning. *Journal of cheminformatics*, 9(1):48, 2017.

Polykovskiy, D., Zhebrak, A., Sanchez-Lengeling, B., Golovanov, S., Tatanov, O., Belyaev, S., Kurbanov, R., Artamonov, A., Aladinskiy, V., Veselov, M., Kadurin, A., Nikolenko, S., Aspuru-Guzik, A., and Zhavoronkov, A. Molecular Sets (MOSES): A Benchmarking Platform for Molecular Generation Models. *arXiv preprint arXiv:1811.12823*, 2018.

Popova, M., Isayev, O., and Tropsha, A. Deep reinforcement learning for de novo drug design. *Science advances*, 4(7): eaap7885, 2018.

Rogers, D. and Hahn, M. Extended-connectivity fingerprints. *Journal of chemical information and modeling*, 50 (5):742–754, 2010.

Samanta, B., De, A., Jana, G., Chattaraj, P. K., Ganguly, N., and Gomez-Rodriguez, M. Nevae: A deep generative model for molecular graphs. *arXiv preprint arXiv:1802.05283*, 2018.

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2009.

Schütt, K., Kindermans, P.-J., Felix, H. E. S., Chmiela, S., Tkatchenko, A., and Müller, K.-R. Schnet: A continuous-filter convolutional neural network for modeling quantum interactions. In *Advances in Neural Information Processing Systems*, pp. 992–1002, 2017.

Seff, A., Zhou, W., Damani, F., Doyle, A., and Adams, R. P. Discrete object generation with reversible inductive construction. In *Advances in Neural Information Processing Systems*, pp. 10353–10363, 2019.

Segler, M. H., Kogej, T., Tyrchan, C., and Waller, M. P. Generating focussed molecule libraries for drug discovery with recurrent neural networks. *arXiv preprint arXiv:1701.01329*, 2017.

Simonovsky, M. and Komodakis, N. Graphvae: Towards generation of small graphs using variational autoencoders. *arXiv preprint arXiv:1802.03480*, 2018.

St. John, P. C., Phillips, C., Kemper, T. W., Wilson, A. N., Guan, Y., Crowley, M. F., Nimlos, M. R., and Larsen, R. E. Message-passing neural networks for high-throughput polymer screening. *The Journal of chemical physics*, 150 (23):234111, 2019.

Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., and Bengio, Y. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

Weininger, D. 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.

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

Ying, Z., You, J., Morris, C., Ren, X., Hamilton, W., and Leskovec, J. Hierarchical graph representation learning with differentiable pooling. In *Advances in Neural Information Processing Systems*, pp. 4800–4810, 2018.

You, J., Liu, B., Ying, R., Pande, V., and Leskovec, J. Graph convolutional policy network for goal-directed molecular graph generation. *arXiv preprint arXiv:1806.02473*, 2018a.

You, J., Ying, R., Ren, X., Hamilton, W. L., and Leskovec, J. Graphrnn: A deep generative model for graphs. *arXiv preprint arXiv:1802.08773*, 2018b.

Zhou, Z., Kearnes, S., Li, L., Zare, R. N., and Riley, P. Optimization of molecules via deep reinforcement learning. *arXiv preprint arXiv:1810.08678*, 2018.

Zhu, J.-Y., Zhang, R., Pathak, D., Darrell, T., Efros, A. A., Wang, O., and Shechtman, E. Toward multimodal image-to-image translation. In *Advances in Neural Information Processing Systems*, pp. 465–476, 2017.

A. Motif Construction

To extract motifs, we decompose a molecule $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ into disconnected fragments by breaking all the bridge bonds that will not violate chemical validity. Our motif extraction consists of three steps (see Figure 6):

1. Find all the *bridge* bonds $(u, v) \in \mathcal{E}$, where both u and v have degree $\Delta_u, \Delta_v \geq 2$ and either u or v is part of a ring.
2. Detach all the bridge bonds from its neighbors. Now the graph $\mathcal{G}$ becomes a set of disconnected subgraphs $\mathcal{G}_1, \cdots, \mathcal{G}_N$.
3. Select $\mathcal{G}_i$ as motif in $\mathcal{G}$ if its occurrence in the training set is more than $\Delta = 100$. If $\mathcal{G}_i$ is not selected as motif, further decompose it into rings and bonds and put them into the motif vocabulary $V_\mathcal{S}$.

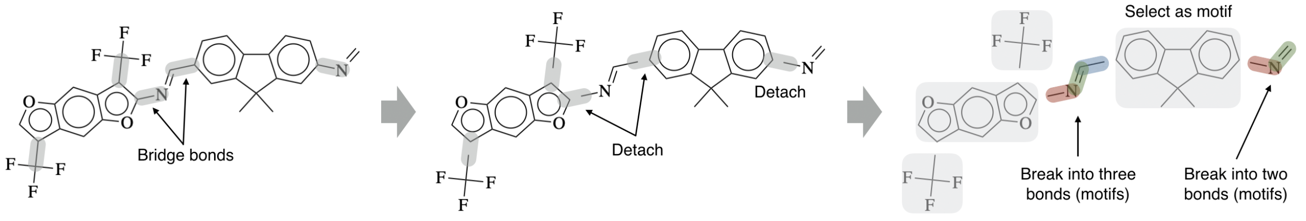
Bridge bonds
Detach
Detach
Select as motif
Break into three bonds (motifs)
Break into two bonds (motifs)

Figure 6. Illustration of motif extraction procedure.

B. Network Architecture

MPN Architecture Our message passing network $\text{MPN}_\psi(\mathcal{H}, \{\boldsymbol{x}_u\}, \{\boldsymbol{x}_{uv}\})$ is a slight modification from the MPN architecture used in Dai et al. (2016); Jin et al. (2019). Let $N(v)$ be the neighbors of node v, $\boldsymbol{x}_v$ the node feature of v and $\boldsymbol{x}_{uv}$ be the feature of edge (u, v). During encoding, each edge (u, v) is associated with two messages $\boldsymbol{\nu}_{uv}$ and $\boldsymbol{\nu}_{vu}$, representing the message from u to v and vice versa. The messages are updated by an LSTM cell with parameters $\psi = \{\boldsymbol{W}_\psi^z, \boldsymbol{W}_\psi^o, \boldsymbol{W}_\psi^r, \boldsymbol{W}_\psi\}$ defined as follows:

Algorithm 1 LSTM Message Passing

```
**function** $\text{LSTM}_\psi\left(\boldsymbol{x}_u, \boldsymbol{x}_{uv}, \{\boldsymbol{\nu}_{wu}^{(t)}, \boldsymbol{c}_{wu}^{(t)}\}_{w \in N(u) \setminus v}\right)$
    $\boldsymbol{i}_{uv} = \sigma\left(\boldsymbol{W}_\psi^z\left[\boldsymbol{x}_u, \boldsymbol{x}_{uv}, \sum_w \boldsymbol{\nu}_{wu}^{(t)}\right] + \boldsymbol{b}^z\right)$        $\boldsymbol{o}_{uv} = \sigma\left(\boldsymbol{W}_\psi^o\left[\boldsymbol{x}_u, \boldsymbol{x}_{uv}, \sum_w \boldsymbol{\nu}_{wu}^{(t)}\right] + \boldsymbol{b}^o\right)$
    $\boldsymbol{f}_{wu} = \sigma\left(\boldsymbol{W}_\psi^r\left[\boldsymbol{x}_u, \boldsymbol{x}_{uv}, \boldsymbol{\nu}_{wu}^{(t)}\right] + \boldsymbol{b}^r\right)$        $\tilde{\boldsymbol{c}}_{uv}^{(t+1)} = \tanh\left(\boldsymbol{W}_\psi\left[\boldsymbol{x}_u, \boldsymbol{x}_{uv}, \sum_w \boldsymbol{\nu}_{wu}^{(t)}\right] + \boldsymbol{b}\right)$
    $\boldsymbol{c}_{uv}^{(t+1)} = \boldsymbol{i}_{uv} \odot \tilde{\boldsymbol{c}}_{uv}^{(t+1)} + \sum_w \boldsymbol{f}_{wu} \odot \boldsymbol{c}_{wu}^{(t)}$        $\boldsymbol{\nu}_{uv}^{(t+1)} = \boldsymbol{o}_{uv} \odot \tanh\left(\boldsymbol{c}_{uv}^{(t+1)}\right)$
    **Return** $\boldsymbol{\nu}_{uv}^{(t+1)}, \boldsymbol{c}_{uv}^{(t+1)}$
**end function**

**function** $\text{MPN}_\psi(\mathcal{H}, \{\boldsymbol{x}_v\}, \{\boldsymbol{x}_{uv}\})$
    Initialize messages: $\boldsymbol{\nu}_{uv}^0 = \mathbf{0}, \boldsymbol{c}_{uv}^0 = \mathbf{0}$
    **for** $t = 0$ **to** $T - 1$ **do**
        Compute messages $\boldsymbol{\nu}_{uv}^{(t+1)}, \boldsymbol{c}_{uv}^{(t+1)} = \text{LSTM}_\psi\left(\boldsymbol{x}_u, \boldsymbol{x}_{uv}, \{\boldsymbol{\nu}_{wu}^{(t)}, \boldsymbol{c}_{wu}^{(t)}\}_{w \in N(u) \setminus v}\right)$ for all edges $(u, v) \in \mathcal{H}$.
    **end for**
    **Return** node representations $\boldsymbol{h}_v = \text{MLP}\left(\boldsymbol{x}_v, \sum_{u \in N(v)} \boldsymbol{\nu}_{uv}^{(T)}\right)$
**end function**
```

AtomG2G Architecture AtomG2G is an atom-based translation method that is directly comparable to HierG2G. Here molecules are represented solely as molecular graphs rather than a hierarchical graph with motifs. The encoder of AtomG2G uses the same LSTM MPN to encode molecular graph. This gives us a set of atom vectors $\boldsymbol{c}_X^\mathcal{G}$ representing molecule X only at the atom level. The decoder of AtomG2G is illustrated in Figure 7. Following You et al. (2018b); Liu et al. (2018), the model generates molecule $\mathcal{G}$ atom by atom following their breadth-first order. During generation, it maintains a FIFO queue

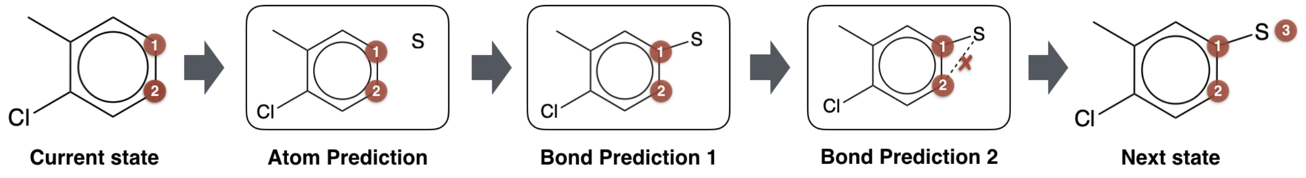

Figure 7. Illustration of AtomG2G decoding process. Atoms marked with red circles are frontier nodes in the queue $\mathcal{Q}$. In each step, the model picks the first node v_t from $\mathcal{Q}$ and predict whether there will be new atoms attached to v_t. If so, it predicts the atom type of new node u_t (atom prediction). Then the model predicts the bond type between u_t and other nodes in $\mathcal{Q}$ sequentially for $|\mathcal{Q}|$ steps (bond prediction, $|\mathcal{Q}| = 2$). Finally, it adds the new atom to the queue $\mathcal{Q}$.

$\mathcal{Q}$ that contains the frontier nodes in the graph (i.e., nodes who still have neighbors to be generated). Let v_t be the first node in $\mathcal{Q}$ and $\mathcal{G}_t$ be the current graph at step t. In each step, the model makes three predictions to expand the graph $\mathcal{G}_t$:

1. It predicts whether there will be new atoms attached to v_t. If not, the model discards v and move on to the next node in $\mathcal{Q}$. The generation stops if $\mathcal{Q}$ is empty.
2. Otherwise, it creates a new atom u_t and predicts its atom type.
3. Lastly, it predicts the bond type between u_t and other frontier nodes in $\mathcal{Q}$ autoregressively to fully capture edge dependencies. Since nodes are generated in breadth-first order, there will be no edges between u_t and nodes outside of $\mathcal{Q}$.

To make those predictions, we use the same LSTM MPN to encode the current graph $\mathcal{G}_t$. Let $\boldsymbol{h}_{v_t}$ be the atom representation of v_t. We represent $\mathcal{G}_t$ as the sum of all its atom vectors $\boldsymbol{h}_{\mathcal{G}_t} = \sum_{v \in \mathcal{G}_t} \boldsymbol{h}_v$. In the first step, we model the probability of expanding a new node from v_t as:

$$\boldsymbol{p}_t = \sigma(\mathrm{MLP}(\boldsymbol{h}_{v_t}, \boldsymbol{h}_{\mathcal{G}_t}, \boldsymbol{\alpha}_t^d)) \qquad \boldsymbol{\alpha}_t^d = \mathrm{attention}_d\left([\boldsymbol{h}_{v_t}, \boldsymbol{h}_{\mathcal{G}_t}], \boldsymbol{c}_X^{\mathcal{G}}\right) \tag{18}$$

In the second step, the atom type of the new node u_t is predicted using another MLP:

$$\boldsymbol{q}_t = \mathrm{softmax}(\mathrm{MLP}(\boldsymbol{h}_{v_t}, \boldsymbol{h}_{\mathcal{G}_t}, \boldsymbol{\alpha}_t^s)) \qquad \boldsymbol{\alpha}_t^s = \mathrm{attention}_s\left([\boldsymbol{h}_{v_t}, \boldsymbol{h}_{\mathcal{G}_t}], \boldsymbol{c}_X^{\mathcal{G}}\right) \tag{19}$$

In the last step, we predict the bonds between u_t and nodes in $\mathcal{Q} = a_1, \cdots, a_n$ sequentially starting with $a_1 = v_t$. Specifically, for each atom pair (u_t, a_k), we predict their bond type (single, double, triple or none) as the following:

$$\boldsymbol{b}_{u_t, a_k} = \mathrm{softmax}(\mathrm{MLP}(\boldsymbol{h}_{\mathcal{G}_t}, \boldsymbol{h}_{u_t}^k, \boldsymbol{h}_{a_k}, \boldsymbol{\alpha}_t^b)) \qquad \boldsymbol{\alpha}_t^b = \mathrm{attention}_b\left([\boldsymbol{h}_{\mathcal{G}_t}, \boldsymbol{h}_{u_t}^k, \boldsymbol{h}_{a_k}], \boldsymbol{c}_X^{\mathcal{G}}\right) \tag{20}$$

where $\boldsymbol{h}_{a_k}$ is the atom representation of node a_k and $\boldsymbol{h}_{u_t}^k$ is the representation of node u_t at the k^{th} bond prediction. Let $N_k(u_t)$ be node u_t's current neighbor predicted in the first k steps. $\boldsymbol{h}_{u_t}^k$ is computed as follows to reflect its local graph structure after k^{th} bond prediction:

$$\boldsymbol{h}_{u_t}^k = \mathrm{MLP}\left(\boldsymbol{x}_{u_t}, \sum\nolimits_{w \in N_k(u_t)} \boldsymbol{\nu}_{w, u_t}\right) \qquad \boldsymbol{\nu}_{w, u_t} = \mathrm{MLP}(\boldsymbol{h}_w, \boldsymbol{x}_{w, u_t}) \tag{21}$$

where $\boldsymbol{x}_{u_t}$ is the atom feature of u_t (i.e., predicted atom type) and $\boldsymbol{x}_{w,u_t}$ is the bond feature between w and u_t (i.e., predicted bond type). Intuitively, this can be viewed as running one-step message passing at each bond prediction step (i.e., passing the message $\boldsymbol{\nu}_{w,u_t}$ from w to u_t). AtomG2G is trained under the same variational objective as HierG2G, with the latent code $\boldsymbol{z}$ sampled from the posterior $Q(\boldsymbol{z}|X, Y) = \mathcal{N}(\boldsymbol{\mu}_{X,Y}, \boldsymbol{\sigma}_{X,Y})$ and $[\boldsymbol{\mu}_{X,Y}, \boldsymbol{\sigma}_{X,Y}] = \mathrm{MLP}(\sum \boldsymbol{c}_Y^{\mathcal{G}} - \sum \boldsymbol{c}_X^{\mathcal{G}})$.

C. Experimental Details

C.1. Polymer Generation

Data The polymer dataset (St. John et al., 2019) is downloaded from `https://cscdata.nrel.gov/#/datasets/ad5d2c9a-af0a-4d72-b943-1e433d5750d6`. The dataset and motif vocabulary sizes are listed in Table 4.

Hyperparameters For HierVAE, we set the hidden layer dimension to be 400 and latent code dimension $|\boldsymbol{z}| = 20$. For CG-VAE and JT-VAE, we used their official implementations for our experiments with their suggested hyperparameters. We set the KL regularization weight $\lambda_{\mathrm{KL}} = 0.1$ for all models. Each model has around 5M parameters.

	Polymer	logP ($\delta = 0.6$)	logP ($\delta = 0.4$)	QED	DRD2
Training set size	76K	75K	99K	88K	34K
Test set size	5000	800	800	800	1000
Motif vocabulary size $\lvert\mathcal{S}\rvert$	436	478	462	307	307
Attachment vocabulary size (avg.) $\lvert\mathcal{A}(\mathcal{S}_t)\rvert$	5.24	3.68	3.50	3.62	3.30

Table 4. Training set size and motif vocabulary size for each dataset.

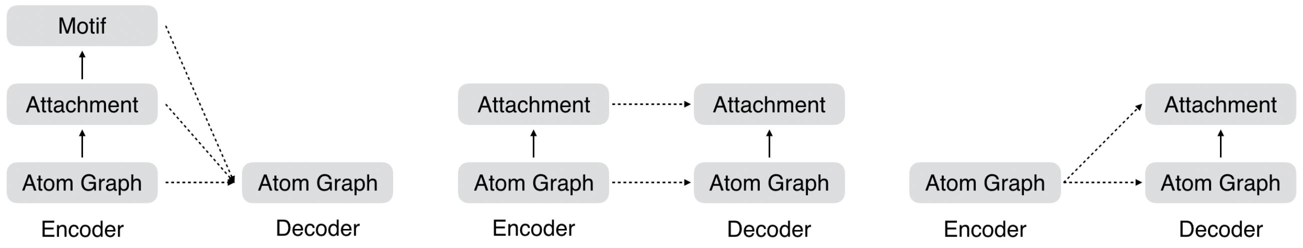

Figure 8. Ablation studies in graph translation tasks. **Left**: Atom-based decoder; **Middle**: Two-layer encoder; **Right**: One-layer encoder.

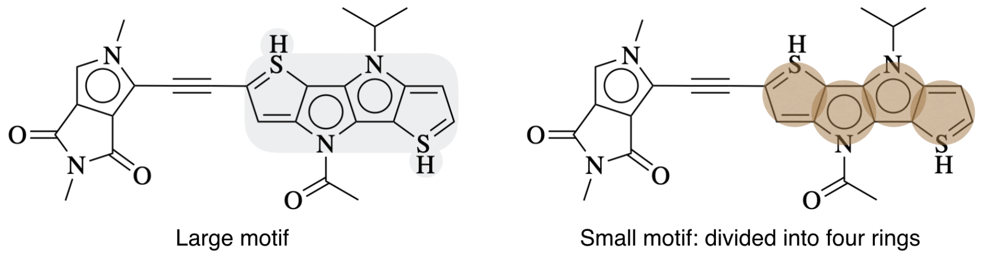

Figure 9. Ablation study of HierVAE (coined as small motif), where motifs are restricted to single rings and bonds.

Metrics and Samples Our metrics are computed using the implementation from Polykovskiy et al. (2018) (`https://github.com/molecularsets/moses`). Samples from our models are shown in Figure 10.

Ablation Study Our small motif baseline builds on the same hierarchical architecture as our model, but it generates molecules based on small motifs restricted to single rings or bonds (see Figure 9).

C.2. Graph-to-Graph Translation

Data The graph translation datasets are directly downloaded from the link provided in Jin et al. (2019). The dataset and motif vocabulary size for each dataset is listed in Table 4.

Hyperparameters For HierG2G, we set the hidden layer dimension to be 270 and the embedding layer dimension 200. We set the latent code dimension $|\boldsymbol{z}| = 8$ and KL regularization weight $\lambda_{\text{KL}} = 0.3$. We run $T = 20$ iterations of message passing in each layer of the encoder. For AtomG2G, we set the hidden layer and embedding layer dimension to be 400 so that both models have roughly the same number of parameters. We also set $\lambda_{\text{KL}} = 0.3$ and number of message passing iterations to be $T = 20$. We train both models with Adam optimizer with default parameters.

Ablation Study Our ablation studies are illustrated in Figure 8. In our first experiment, we changed our decoder to the atom-based decoder of AtomG2G. As the encoder is still hierarchical, we modified the input of the decoder attention to include both atom and motif vectors. We set the hidden layer and embedding layer dimension to be 300 to match the original model size. Our next two experiments reduces the number of hierarchies in both our encoder and decoder MPN. In the two-layer model, molecules are represented by $\boldsymbol{c}_X = \boldsymbol{c}_X^{\mathcal{G}} \cup \boldsymbol{c}_X^{\mathcal{A}}$. We make motif predictions based on hidden vector $\boldsymbol{h}_{\mathcal{A}_k}$ instead of $\boldsymbol{h}_{\mathcal{S}_k}$ because the motif layer is removed. In the one-layer model, molecules are represented by $\boldsymbol{c}_X = \boldsymbol{c}_X^{\mathcal{G}}$ and we make motif predictions based on atom vectors $\sum_{v \in \mathcal{S}_k} \boldsymbol{h}_v$. The hidden layer dimension is adjusted accordingly to match the original model size.

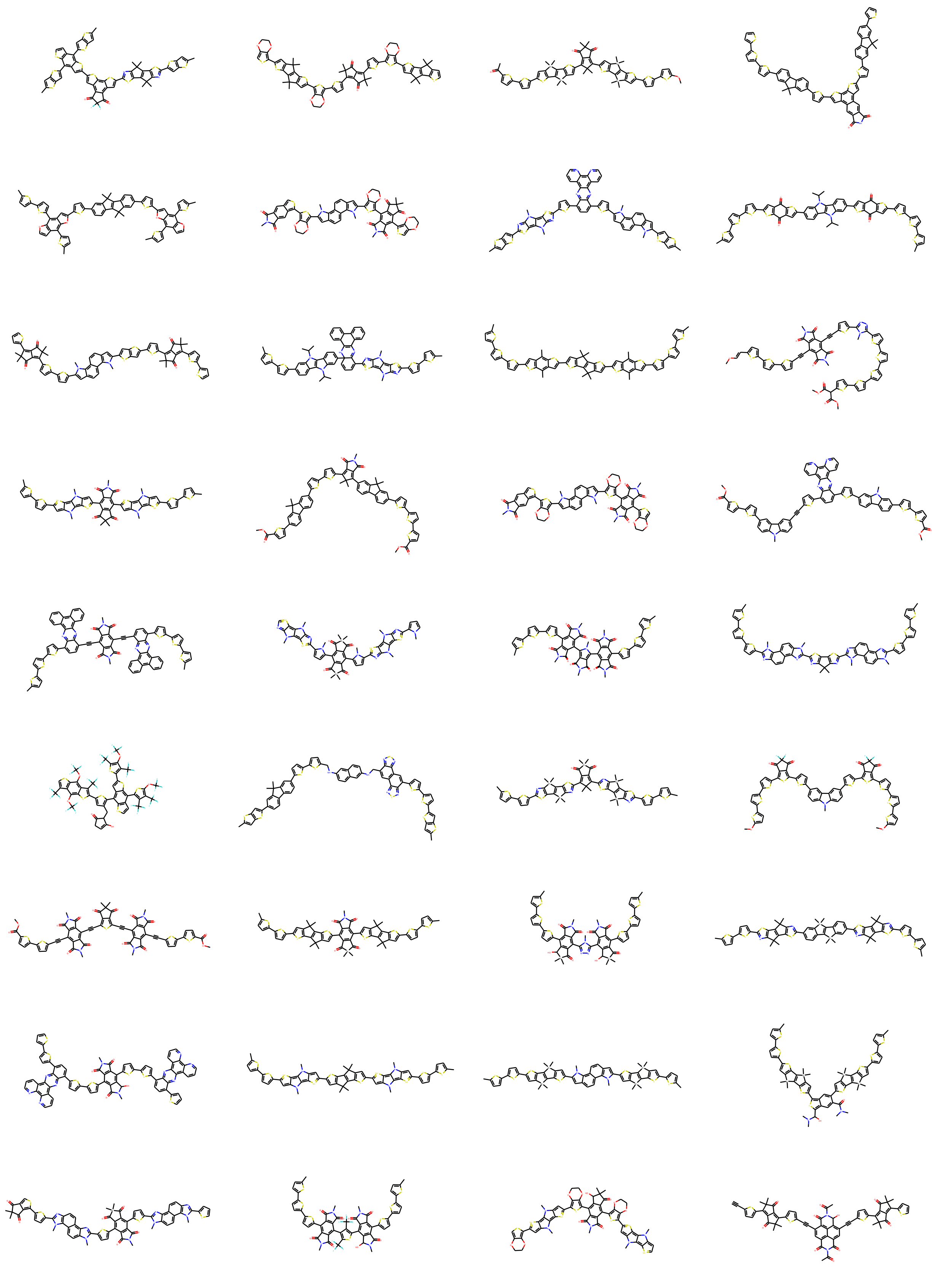

Figure 10. Sampled polymers from HierVAE.