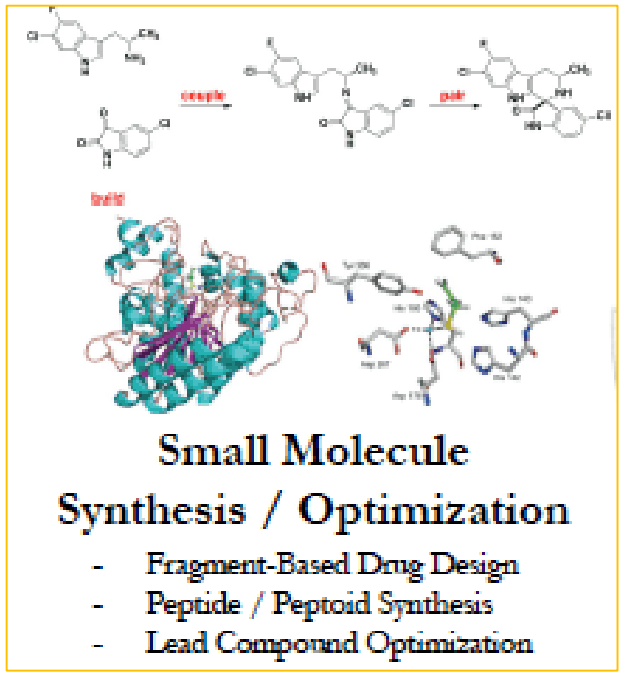
Hierarchical graph-to-graph translation for molecule

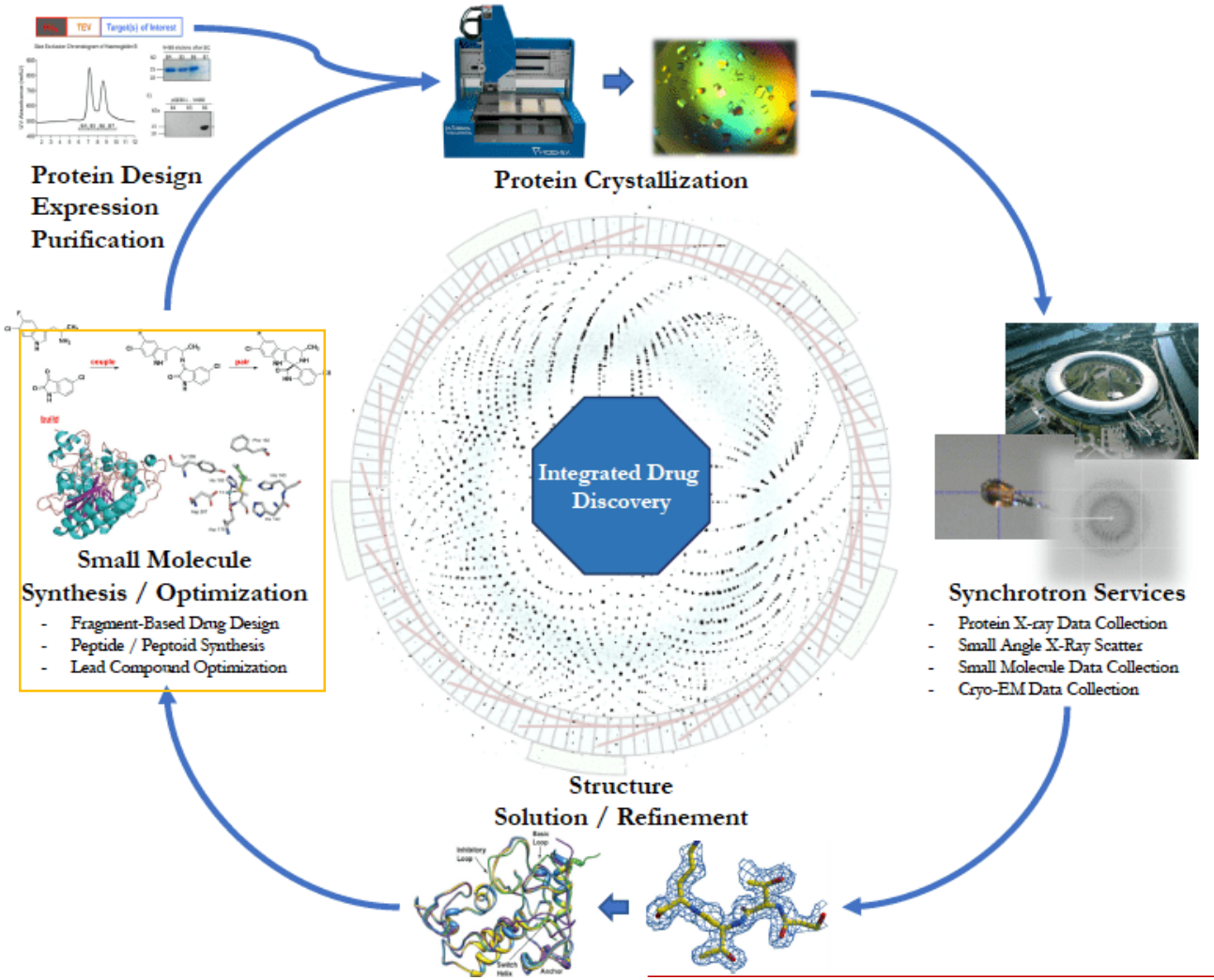
GNN Study Group 2020.03.15

Cecile

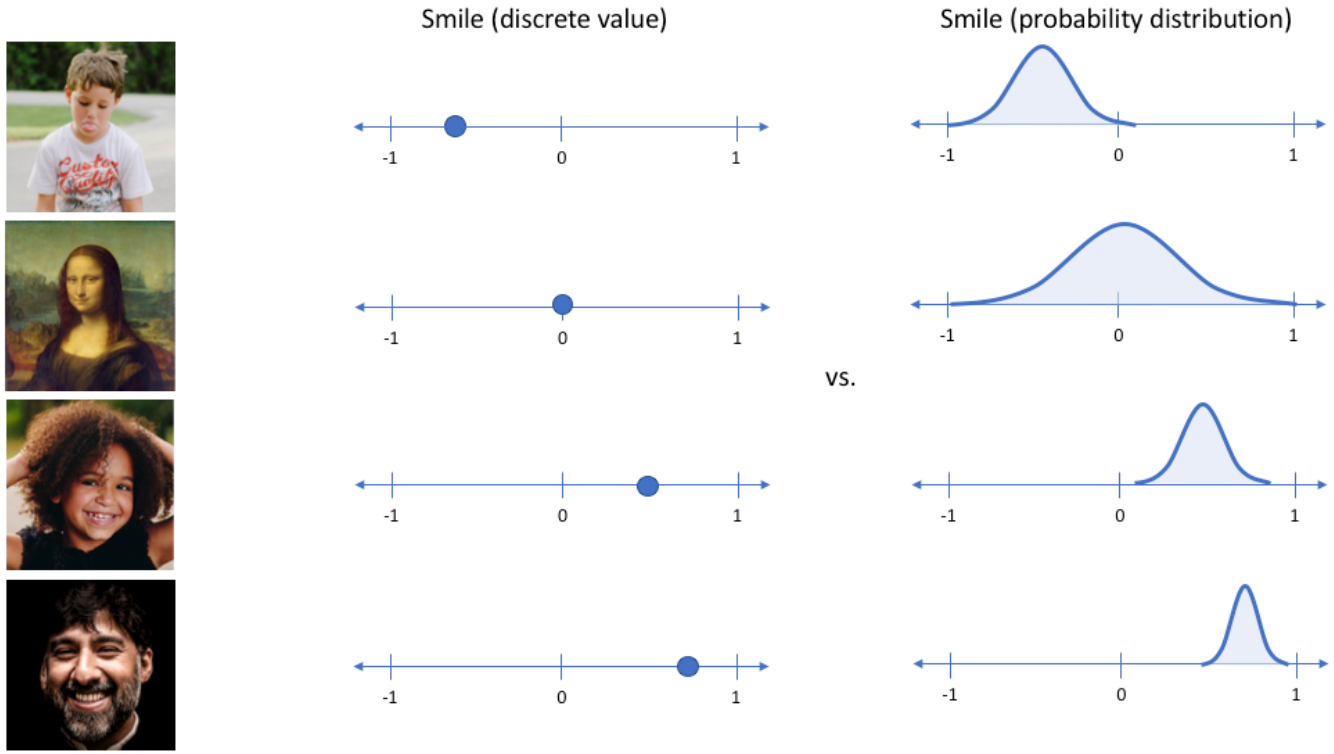
Drug design

The purpose of a drug discovery and development project is to find new therapeutic agents that target a key enzyme, protein-protein interaction, receptor-ligand, or protein-nucleic acid interaction of relevance in a disease of interest in order to mitigate the course of the disease





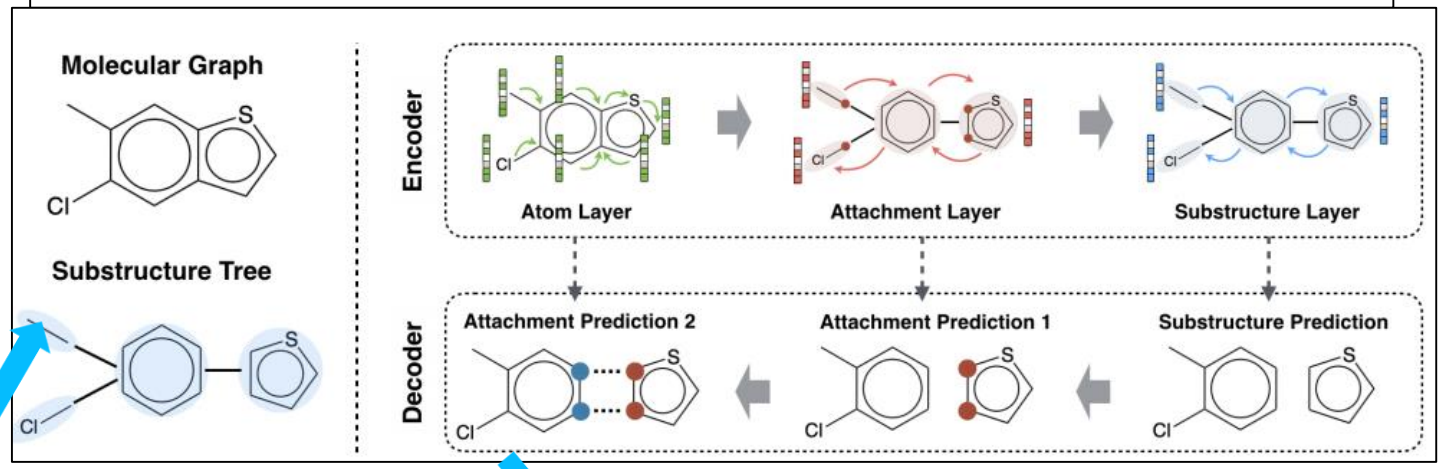
Autoencoder & Variational Autoencoder



Autoencoders learn a “compressed representation” of input (could be image,text sequence etc.) automatically by first compressing the input (encoder) and decompressing it back (decoder) to match the original input. The learning is aided by using distance function that quantifies the information loss that occurs from the lossy compression.

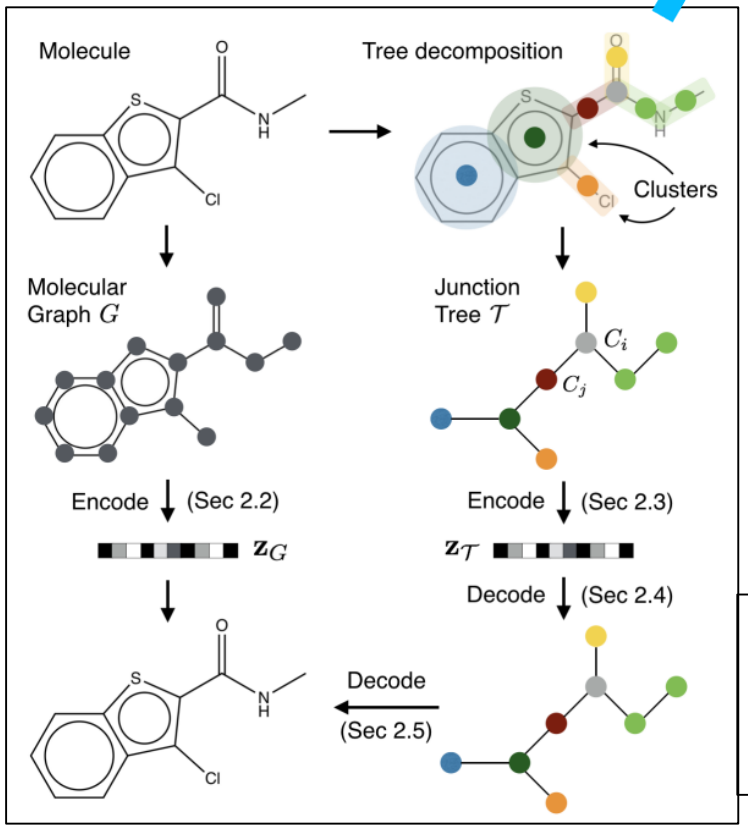
Variational autoencoders learn the parameters of a probability distribution representing the data. Since it learns to model the data, we can sample from the distribution and generate new input data samples.

(ICLR 2020 reject) 2019 Hierarchical Graph-to-graph Translation for molecule



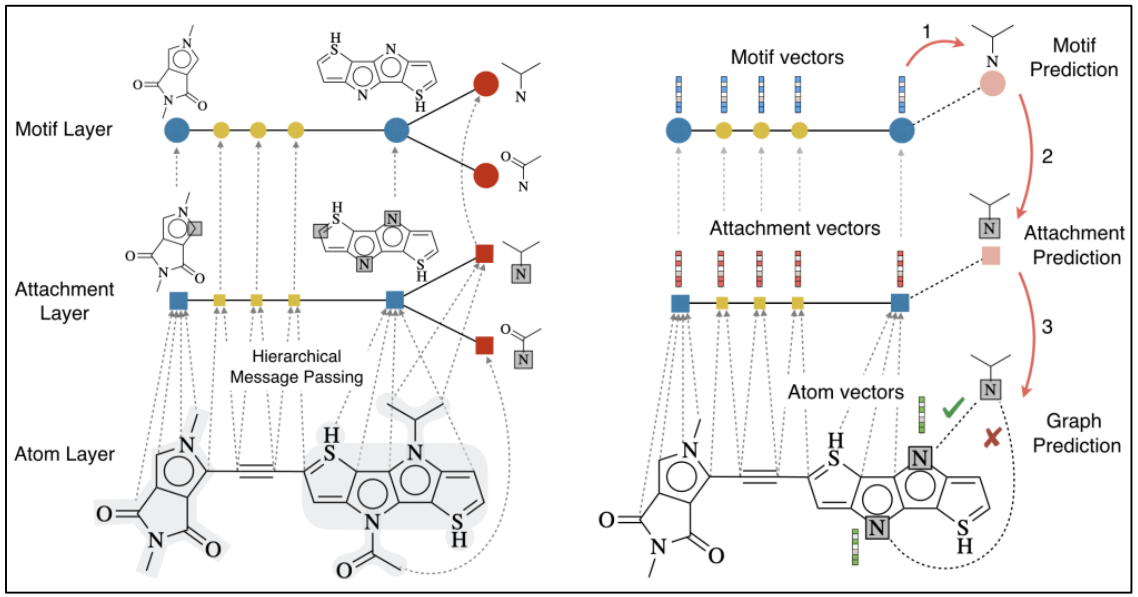
Brings autoregressive concept into decoding process

Improve the decision of selecting a new motif during decoding



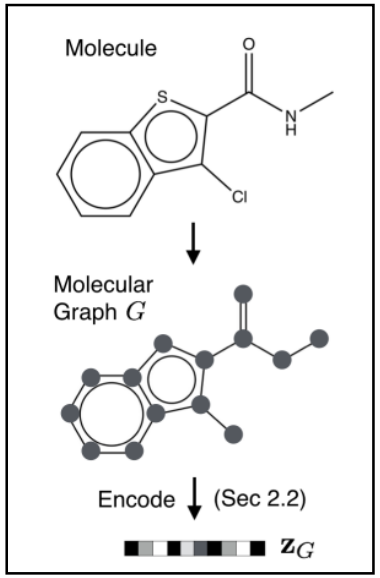
(ICML 2018)
2018 Junction Tree Variational Autoencoder for Molecular Graph Generation

(ICML 2020 submission) Hierarchical Generation of Molecular Graphs using Structural Motifs

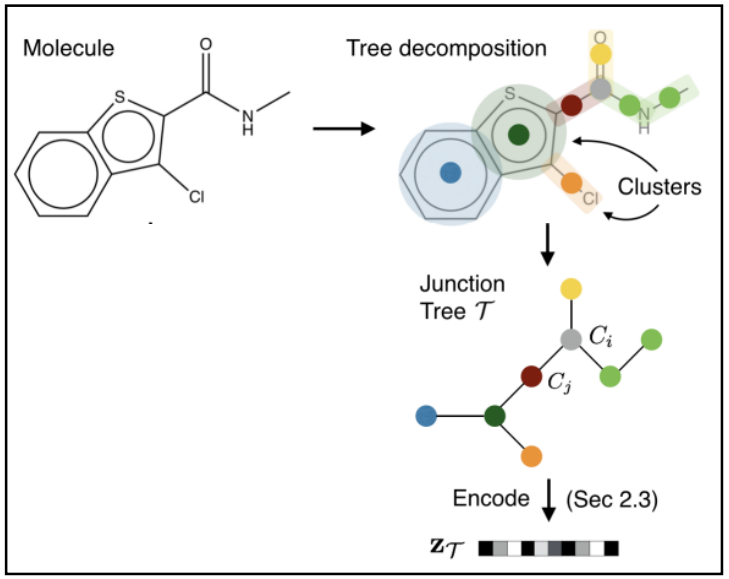


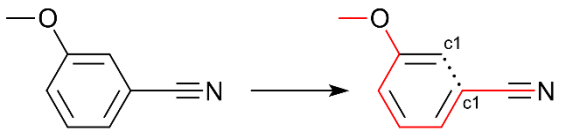
Encoding

Graph encoder



Tree encoder



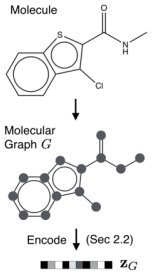


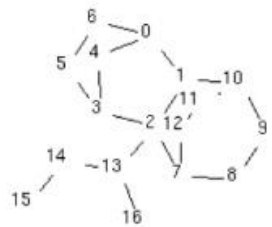
SMILES: COc(c1)cccc1C#N

Graph-to-graph translation：the model is trained to translate an input molecular graph into its better form

https://www.youtube.com/watch?v=QFRv_lOWeKI

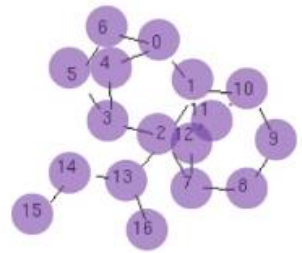
Encoding: graph encoder





Molecular structure

1



Molecular Graph

- fatoms
- fbonds

Features — vertex, edge

- agraph
- bgraph

Graphs

Scope: tracks atoms per molecule
In the whole dataset

Index	Symbol	Number of bonds	Formal Charge	Chirality	Part of aromatic ring
0	C	3	0	0	FALSE
1	C	4	0	0	FALSE
2	C	3	0	0	FALSE
3	C	2	0	0	FALSE
4	C	2	0	0	FALSE
5	C	2	0	0	FALSE
6	C	3	0	0	FALSE
7	C	2	0	0	FALSE
8	C	2	0	0	FALSE
9	C	3	0	0	FALSE
10	C	2	0	0	FALSE
11	C	2	0	0	FALSE
12	C	3	0	0	FALSE
13	C	2	0	0	FALSE
14	C	1	0	0	FALSE
15	C	1	0	0	FALSE

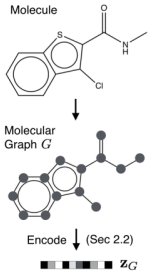
Mapping

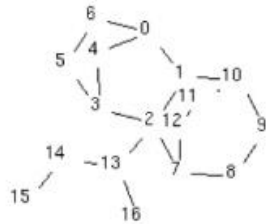
```
tensor([ 1.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,
         0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,
         0.,  0.,  1.,  0.,  0.,  0.,  0.,  0.,  0.,  1.,  1.,  0.,
         0.,  0.,  0.])
```

torch.Size([1,39])

Perform for all atoms → **Atom features (fatoms)**

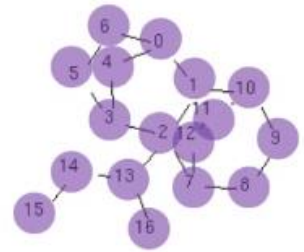
Encoding: graph encoder





Molecular structure

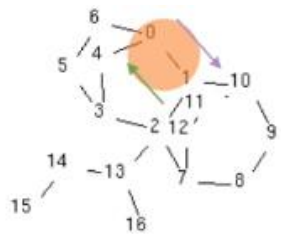
1



Molecular Graph

- fatoms
- fbonds
- **Features** → vertex, edge
- agraph
- bgraph
- **Graphs**

Scope: tracks atoms per molecule
In the whole dataset



Bond Atoms	Type	Cis/Trans, E/Z
[0, 1]	SINGLE	0
[1, 2]	SINGLE	0
[2, 3]	SINGLE	0
[3, 4]	SINGLE	0
[3, 5]	SINGLE	0
[5, 6]	SINGLE	0
[2, 7]	SINGLE	0
[7, 8]	SINGLE	0
[8, 9]	SINGLE	0
[9, 10]	SINGLE	0
[10, 11]	SINGLE	0
[11, 12]	SINGLE	0
[2, 13]	SINGLE	0
[13, 14]	SINGLE	0
[14, 15]	SINGLE	0

Mapping

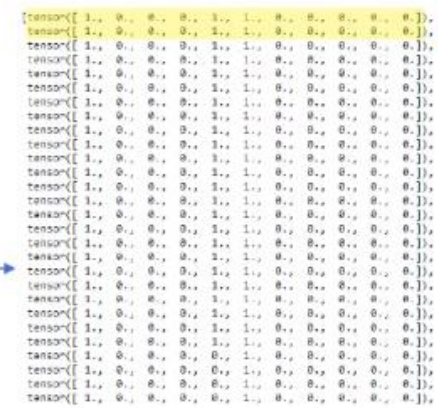
torch.Size([2,11])

```
0 1
0 tensor([ 1.,  0.,  0.,  0.,  1.,  1.,  0.,  0.,  0.,  0.,  0.])
1 tensor([ 1.,  0.,  0.,  0.,  1.,  1.,  0.,  0.,  0.,  0.,  0.])
```

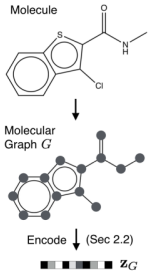
The same

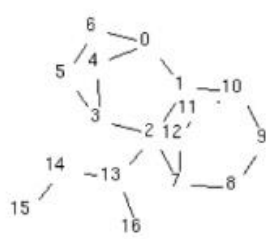
Perform for all bonds

For 0-1 bond



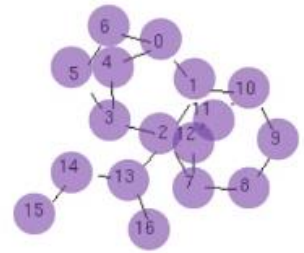
Encoding: graph encoder





Molecular structure

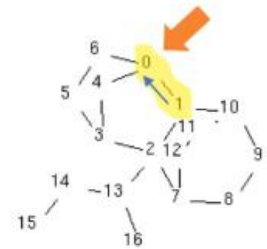
1



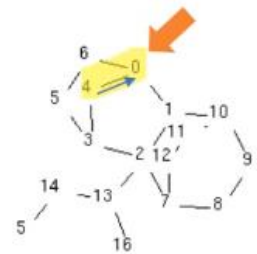
Molecular Graph

- Features
  - vertex
  - edge
- Graphs
  - fatoms
  - fbonds
  - agraph
  - bgraph

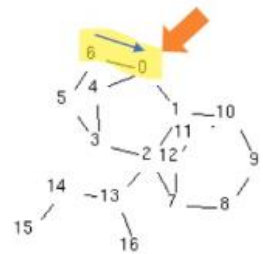
Scope: tracks atoms per molecule
In the whole dataset



Atom



Related bonds



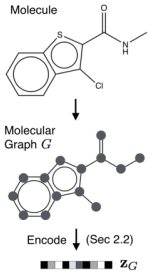
1 (0, 1)	21 (10, 11)
2 (1, 0)	22 (11, 10)
3 (1, 2)	23 (11, 12)
4 (2, 1)	24 (12, 11)
5 (2, 3)	25 (2, 13)
6 (3, 2)	26 (13, 2)
7 (3, 4)	27 (13, 14)
8 (4, 3)	28 (14, 13)
9 (3, 5)	29 (14, 15)
10 (5, 3)	30 (15, 14)
11 (5, 6)	31 (13, 16)
12 (6, 5)	32 (16, 13)
13 (2, 7)	33 (4, 0)
14 (7, 2)	34 (0, 4)
15 (7, 8)	35 (6, 0)
16 (8, 7)	36 (0, 6)
17 (8, 9)	37 (10, 1)
18 (9, 8)	38 (1, 10)
19 (9, 10)	39 (12, 7)
20 (10, 9)	40 (7, 12)

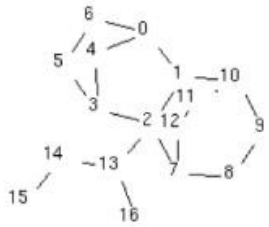
0 [2, 33, 35]
1 [1, 4, 37]
2 [3, 6, 14, 26]
3 [5, 8, 10]
4 [7, 34]
5 [9, 12]
6 [11, 36]
7 [13, 16, 39]
8 [15, 18]
9 [17, 20]
10 [19, 22, 38]
11 [21, 24]
12 [23, 40]
13 [25, 28, 32]
14 [27, 30]
15 [29]
16 [31]

```
tensor([[ 2, 33, 35,  0,  0,  0],
        [ 1,  4, 37,  0,  0,  0],
        [ 3,  6, 14, 26,  0,  0],
        [ 5,  8, 10,  0,  0,  0],
        [ 7, 34,  0,  0,  0,  0],
        [ 9, 12,  0,  0,  0,  0],
        [11, 36,  0,  0,  0,  0],
        [13, 16, 39,  0,  0,  0],
        [15, 18,  0,  0,  0,  0],
        [17, 20,  0,  0,  0,  0],
        [19, 22, 38,  0,  0,  0],
        [21, 24,  0,  0,  0,  0],
        [23, 40,  0,  0,  0,  0],
        [25, 28, 32,  0,  0,  0],
        [27, 30,  0,  0,  0,  0],
        [29,  0,  0,  0,  0,  0],
        [31,  0,  0,  0,  0,  0]])
```

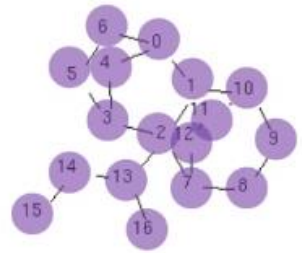
For atom 0

Encoding: graph encoder



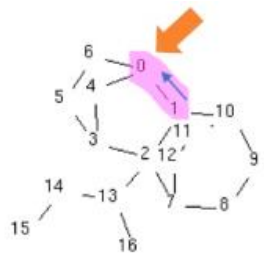


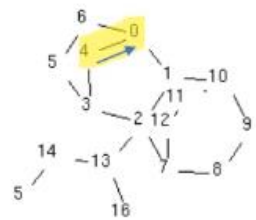
Molecular structure

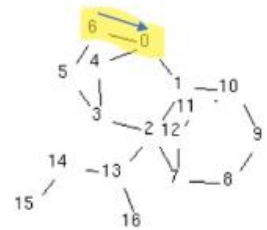


Molecular Graph

- fatoms, fbonds → Features → vertex, edge
- agraph, bgraph → Graphs
- Scope: tracks atoms per molecule In the whole dataset







1 (0, 1)
2 (1, 0)
3 (1, 2)
4 (2, 1)
5 (2, 3)
6 (3, 2)
7 (3, 4)
8 (4, 3)
9 (3, 5)
10 (5, 3)
11 (5, 6)
12 (6, 5)
13 (2, 7)
14 (7, 2)
15 (7, 8)
16 (8, 7)
17 (8, 9)
18 (9, 8)
19 (9, 10)
20 (10, 9)
21 (10, 11)
22 (11, 10)
23 (11, 12)
24 (12, 11)
25 (2, 13)
26 (13, 2)
27 (13, 14)
28 (14, 13)
29 (14, 15)
30 (15, 14)
31 (13, 16)
32 (16, 13)
33 (4, 0)
34 (0, 4)
35 (6, 0)
36 (0, 6)
37 (10, 1)
38 (1, 10)
39 (12, 7)
40 (7, 12)

(1, 0) Is self

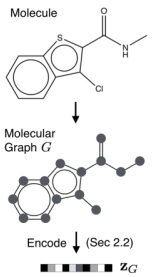
0 [2, 33, 35]
1 [1, 4, 37]
2 [3, 6, 14, 26]
3 [5, 8, 10]
4 [7, 34]
5 [9, 12]
6 [11, 36]
7 [13, 16, 39]
8 [15, 18]
9 [17, 20]
10 [19, 22, 38]
11 [21, 24]
12 [23, 40]
13 [25, 28, 32]
14 [27, 30]
15 [29]
16 [31]

Max number of neighbors, n=6

```
tensor([[ 0,  0,  0,  0,  0,  0],
        [ 0, 33, 35,  0,  0,  0],
        [ 0,  4, 37,  0,  0,  0],
        [ 1,  0, 37,  0,  0,  0],
        [ 0,  6, 14, 26,  0,  0],
        [ 3,  0, 14, 26,  0,  0],
        [ 0,  8, 10,  0,  0,  0],
        [ 5,  0, 10,  0,  0,  0],
        [ 0, 34,  0,  0,  0,  0],
        [ 5,  8,  0,  0,  0,  0],
        [ 0, 12,  0,  0,  0,  0],
        [ 9,  0,  0,  0,  0,  0],
        [ 0, 36,  0,  0,  0,  0],
        [ 3,  6,  0, 26,  0,  0],
        [ 0, 16, 39,  0,  0,  0],
        [13,  0, 39,  0,  0,  0],
        [ 0, 18,  0,  0,  0,  0],
        [15,  0,  0,  0,  0,  0],
        [ 0, 20,  0,  0,  0,  0],
        [17,  0,  0,  0,  0,  0],
        [ 0, 22, 38,  0,  0,  0],
        [19,  0, 38,  0,  0,  0],
        [ 0, 24,  0,  0,  0,  0],
        [21,  0,  0,  0,  0,  0],
        [ 0, 40,  0,  0,  0,  0],
        [ 3,  6, 14,  0,  0,  0],
        [ 0, 28, 32,  0,  0,  0],
        [25,  0, 32,  0,  0,  0],
        [ 0, 30,  0,  0,  0,  0],
        [27,  0,  0,  0,  0,  0],
        [ 0,  0,  0,  0,  0,  0],
        [25, 28,  0,  0,  0,  0],
        [ 0,  0,  0,  0,  0,  0],
        [ 7,  0,  0,  0,  0,  0],
        [ 2,  0, 35,  0,  0,  0],
        [11,  0,  0,  0,  0,  0],
        [ 2, 33,  0,  0,  0,  0],
        [19, 22,  0,  0,  0,  0],
        [ 1,  4,  0,  0,  0,  0],
        [23,  0,  0,  0,  0,  0],
        [13, 16,  0,  0,  0,  0]])
```

For bond 0-1

Encoding: graph encoder



graph

- fatoms
- fbonds

Features — vertex, edge

- agraph
- bgraph

Graphs

Scope: tracks atoms per molecule
In the whole dataset

concatenate

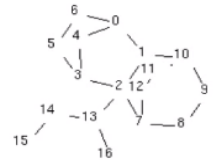
Linear layer
+
Relu

Message

Loo up table

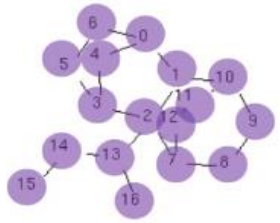
Bond graph

(Same way for agraph)



Molecular structure

1



Molecular Graph

2

Encode

Graph Message Passing Network

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

Message — Atom Features — Bond Features — Old messages from neighbors

$$\mathbf{h}_u = \tau(\mathbf{U}_1^g \mathbf{x}_u + \sum_{v \in N(u)} \mathbf{U}_2^g \nu_{vu}^{(T)})$$

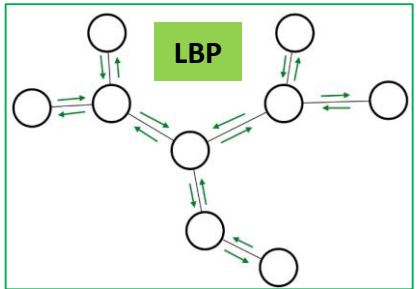
vertex — Atom Features — final messages from neighbors

$$\mathbf{h}_G = \sum_i \mathbf{h}_i / |V|$$

Loopy belief propagation

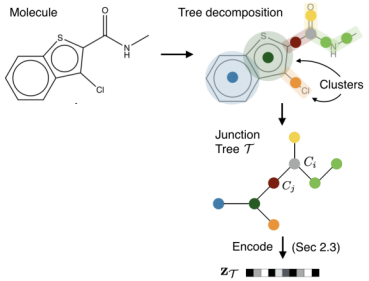
ear layer

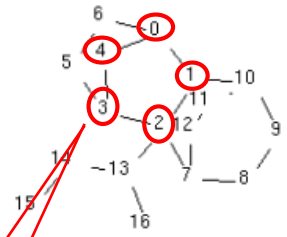
New message

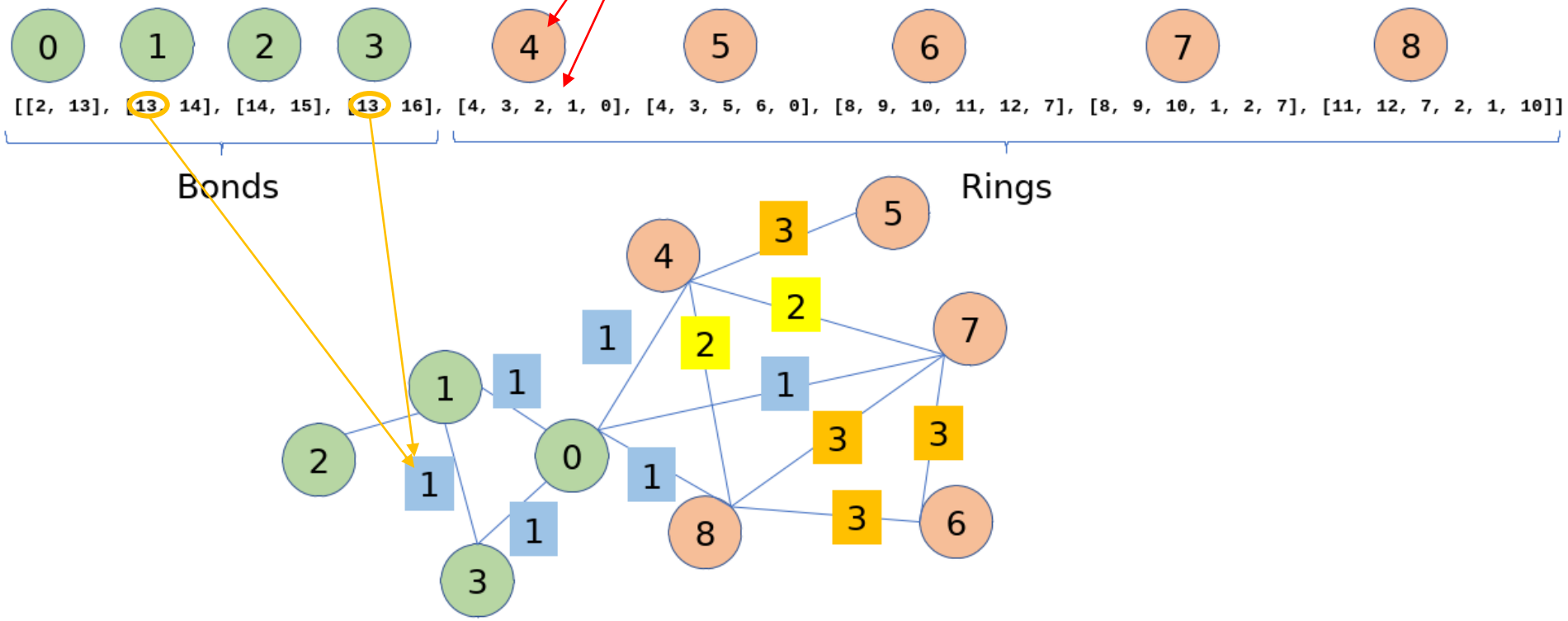


https://www.youtube.com/watch?v=meBWAboEWQk

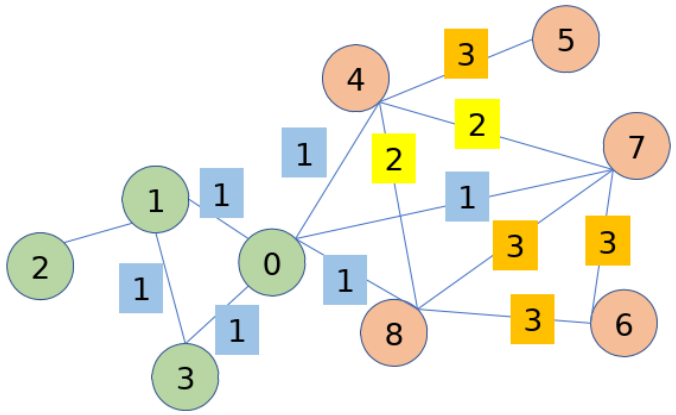
Encoding: Tree encoder

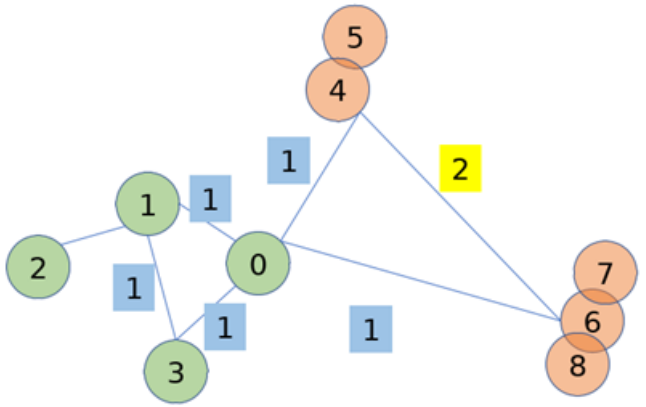


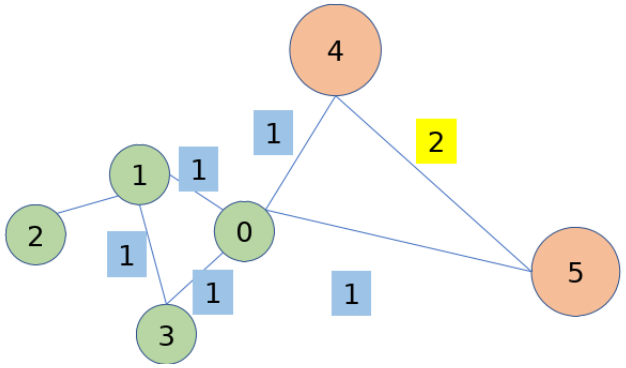


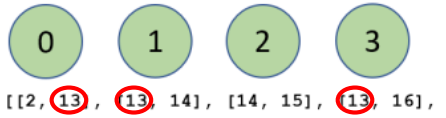


Encoding: Tree encoder

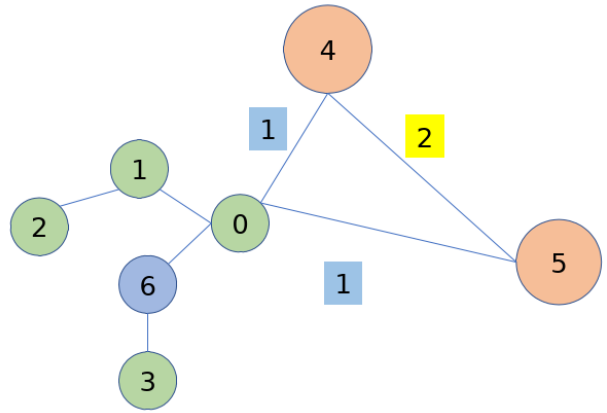


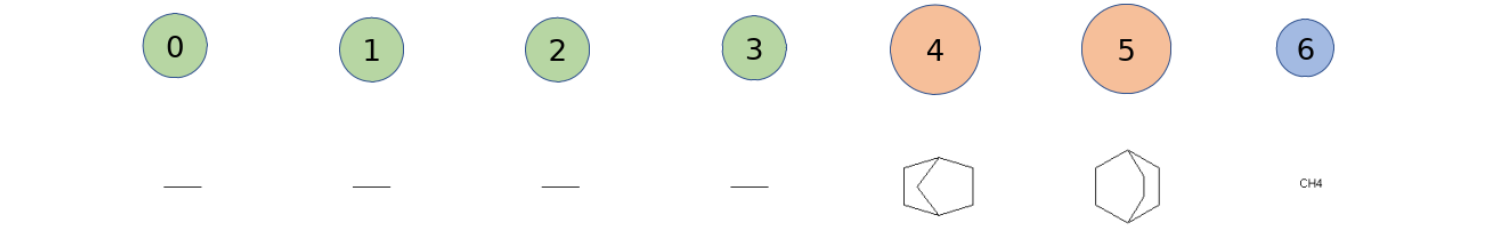




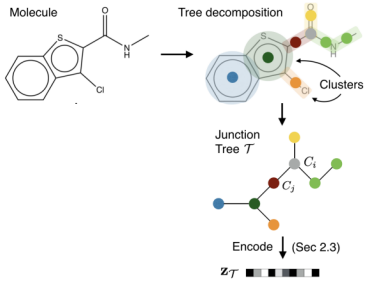


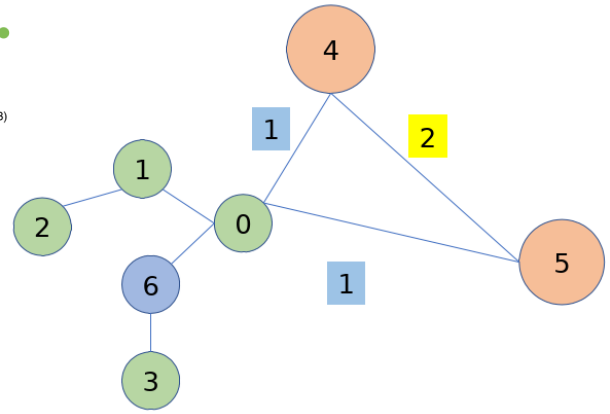
Bonds

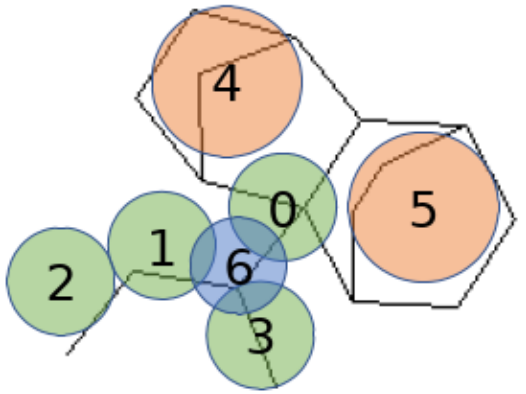


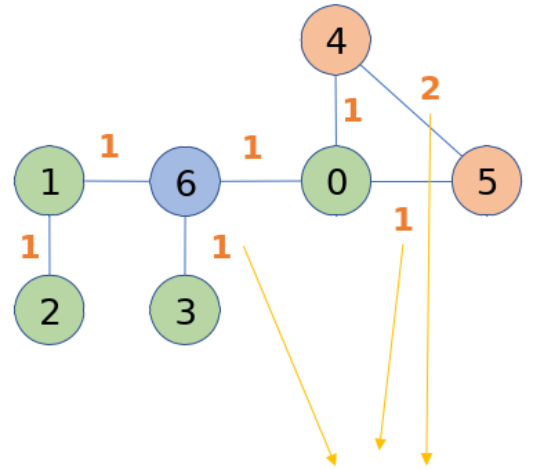


Encoding: Tree encoder

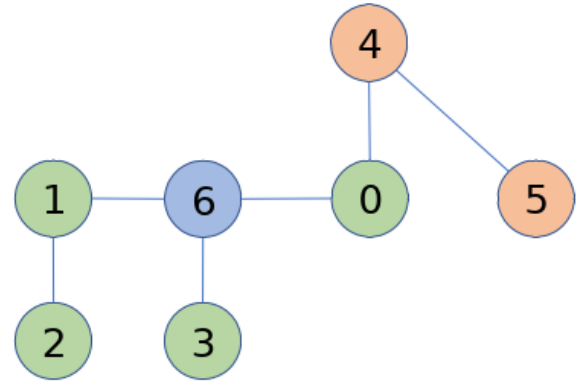


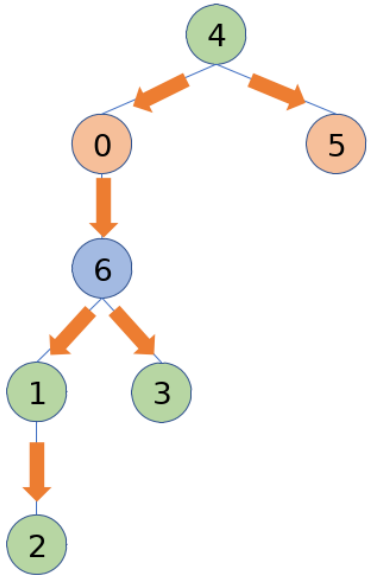


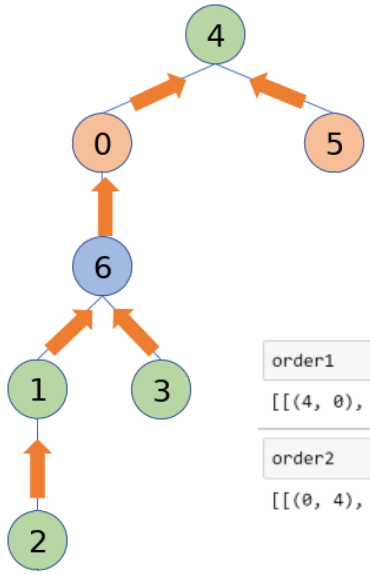




Common atoms between cliques (intersect)








```
order1
[[(4, 0), (4, 5)], [(0, 6)], [(6, 1), (6, 3)], [(1, 2)]]
order2
[[(0, 4), (5, 4)], [(6, 0)], [(1, 6), (3, 6)], [(2, 1)]]
```

Encoding: Tree encoder

$$\mathbf{m}_{ij} = \text{GRU}(\mathbf{x}_i, \{\mathbf{m}_{ki}\}_{k \in N(i) \setminus j}) \Rightarrow \mathbf{h}_i = \tau(\mathbf{W}^o \mathbf{x}_i + \sum_{k \in N(i)} \mathbf{U}^o \mathbf{m}_{ki}) \Rightarrow \mathbf{h}_{\mathcal{T}_G} = \mathbf{h}_{root}$$

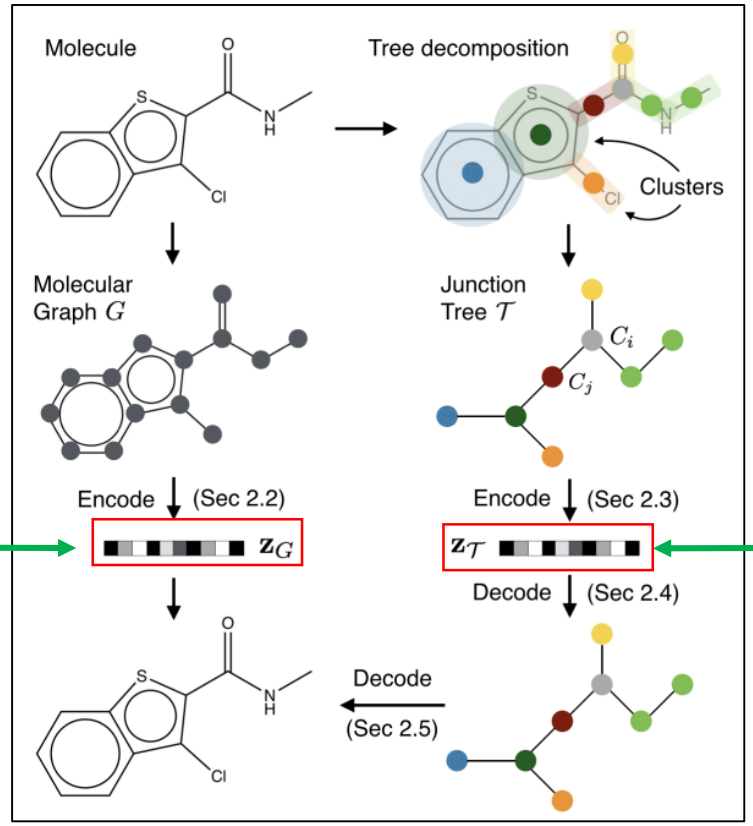
embedding

Memory (hidden_state)

```
order2
[[(0, 4), (5, 4)], [(6, 0)], [(1, 6), (3, 6)], [(2, 1)]]
```

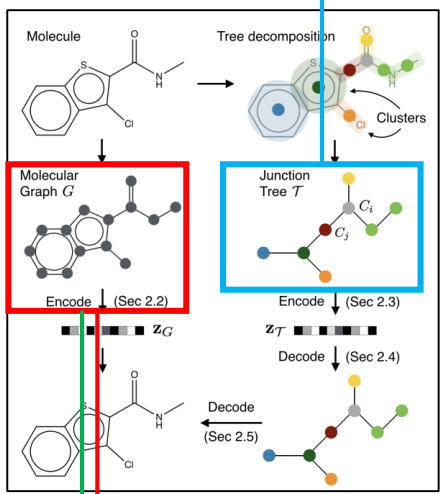
$\mathbf{z}_G$ is sampled from a Gaussian $\mathcal{N}(\boldsymbol{\mu}_G, \boldsymbol{\sigma}_G)$

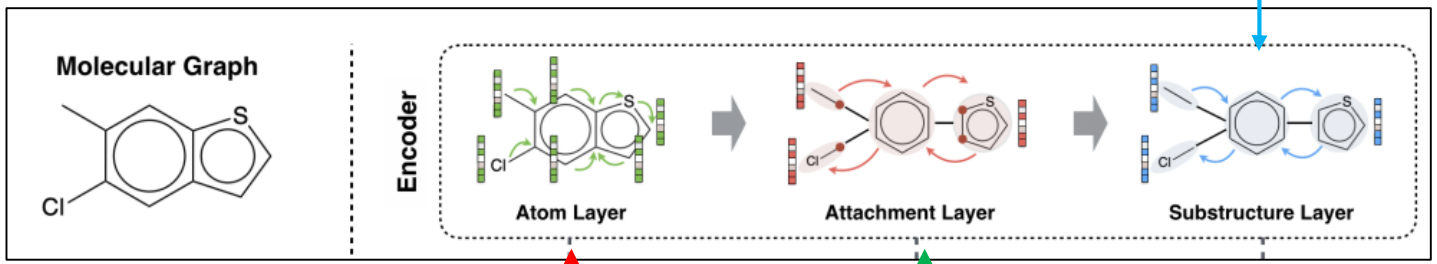
$\mathbf{h}_G$

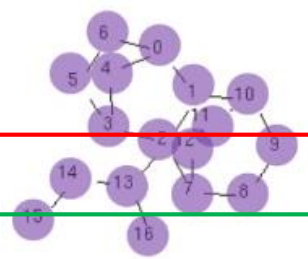


$\mathbf{z}_\mathbf{T}$ is sampled from a Gaussian $\mathcal{N}(\boldsymbol{\mu}_\mathbf{T}, \boldsymbol{\sigma}_\mathbf{T})$

$\mathbf{h}_{\mathcal{T}_G}$





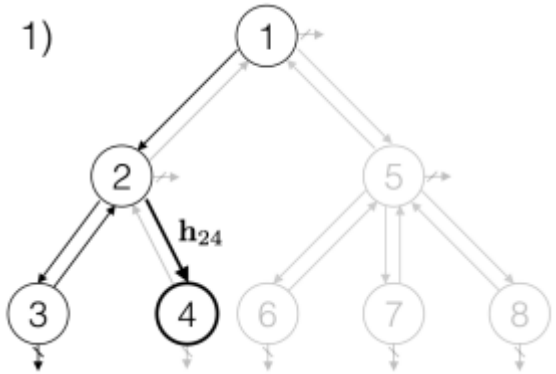


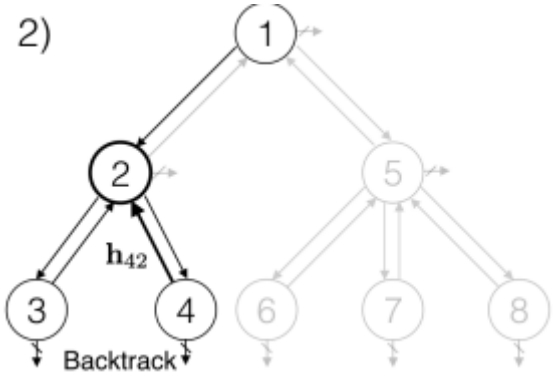
Molecular Graph

fatoms
fbonds
Features
vertex
edge
agraph
bgraph
Graphs

Scope: tracks atoms per molecule
In the whole dataset

Decoding: Tree decoder

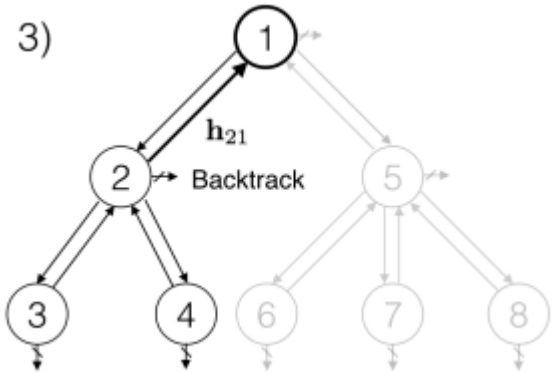




makes a binary prediction on whether it still has children

$$p_t = \sigma(\mathbf{u}^d \cdot \tau(\mathbf{W}_1^d \mathbf{x}_{i_t} + \mathbf{W}_2^d \mathbf{z}_{\mathcal{T}} + \mathbf{W}_3^d \sum_{(k,i_t)\in\tilde{\mathcal{E}}_t} \mathbf{h}_{k,i_t})$$

$$\mathbf{h}_{i_t,j_t} = \mathrm{GRU}(\mathbf{x}_{i_t}, \{\mathbf{h}_{k,i_t}\}_{(k,i_t)\in\tilde{\mathcal{E}}_t, k\neq j_t})$$



When a child node j is generated from its parent, we predict its node label with:

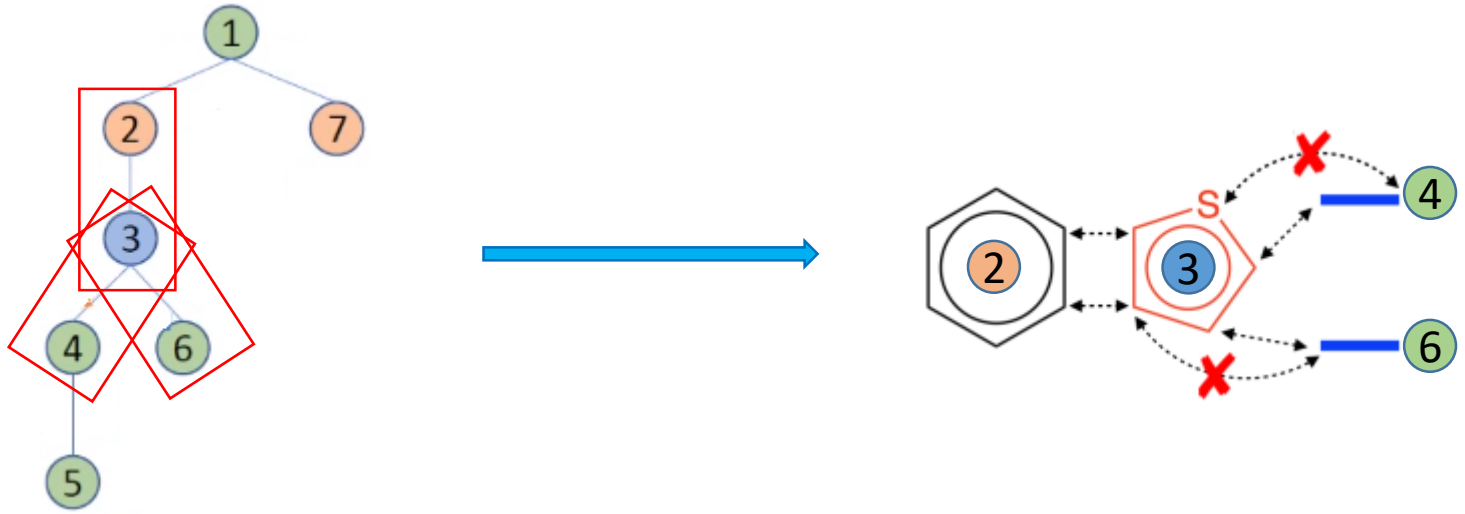
$$\mathbf{q}_j = \mathrm{softmax}(\mathbf{U}^l \tau(\mathbf{W}_1^l \mathbf{z}_{\mathcal{T}} + \mathbf{W}_2^l \mathbf{h}_{ij}))$$

The tree decoder aims to maximize the likelihood $p(\mathcal{T}|\mathbf{z}_{\mathcal{T}})$

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

Decoding: Graph decoder

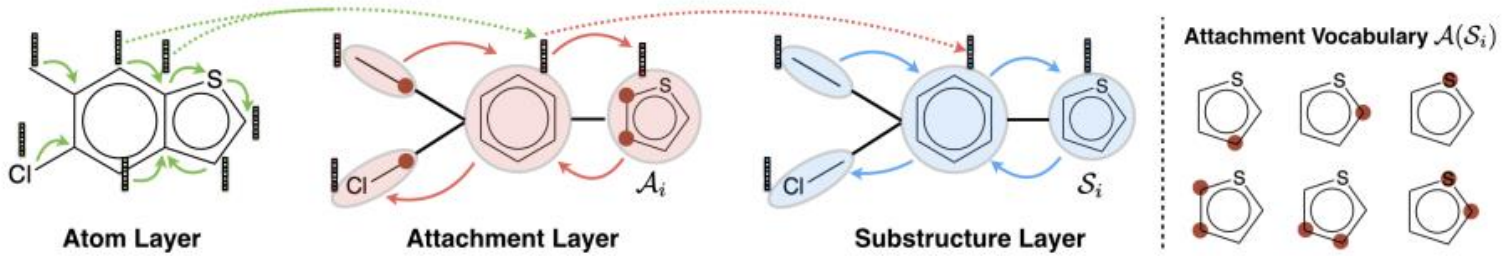
Our goal here is to assemble the subgraphs (nodes in the tree) together into the correct molecular graph.



- We score G_i as a candidate subgraph by first deriving a vector representation $\mathbf{h}_{G_i}$ and then $f_i^a(G_i) = \mathbf{h}_{G_i} \cdot \mathbf{z}_G$ as the subgraph score.

For each node, get all the possible combination with neighbors → Enumerate subgraphs → Score subgraphs

Decoding: Graph decoder



In graph encoder:

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

- $\boldsymbol{\nu}_{uv}^{(t)}$: Message
- $\mathbf{x}_u$: Atom Features
- $\mathbf{x}_{uv}$: Bond Features
- $\boldsymbol{\nu}_{wu}^{(t-1)}$: Old messages from neighbors

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

- $\mathbf{h}_u$: vertex
- $\mathbf{x}_u$: Atom Features
- $\boldsymbol{\nu}_{vu}^{(T)}$: final messages from neighbors

$$\mathbf{h}_G = \sum_i \mathbf{h}_i / |V|$$

In graph decoder:

$$\boldsymbol{\mu}_{uv}^{(t)} = \tau(\mathbf{W}_1^a \mathbf{x}_u + \mathbf{W}_2^a \mathbf{x}_{uv} + \mathbf{W}_3^a \widetilde{\boldsymbol{\mu}}_{uv}^{(t-1)})$$

- $\boldsymbol{\mu}_{uv}^{(t)}$: Atoms of subgraph u-v bond (edge)
- $\mathbf{x}_u$: Atom Features
- $\mathbf{x}_{uv}$: Bond Features
- $\widetilde{\boldsymbol{\mu}}_{uv}^{(t-1)}$: Old message from neighbors

$$\widetilde{\boldsymbol{\mu}}_{uv}^{(t-1)} = \begin{cases} \sum_{w \in N(u) \setminus v} \boldsymbol{\mu}_{wu}^{(t-1)} & \alpha_u = \alpha_v \\ \widehat{\mathbf{m}}_{\alpha_u \alpha_v} + \sum_{w \in N(u) \setminus v} \boldsymbol{\mu}_{wu}^{(t-1)} & \alpha_u \neq \alpha_v \end{cases}$$

- $\widehat{\mathbf{m}}_{\alpha_u \alpha_v}$: Tree message

Generated graph representation($\mathbf{h}_{G_i}$) = average[atom features* weight + sum(messages from neighbors)]

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