Graph Representation Learning and Deep Generative Models On Graphs

Dr. Truong Son Hy

Assistant Professor
Department of Mathematics and Computer Science, Indiana State University

Introduction

Objective of my research

Advanced machine learning and deep learning for scientific problems.

For our talk today:

1. Graph representation learning
   - Message passing neural networks
   - Limitations of GNNs
   - Multiresolution Graph Transformers
   - Permutation equivariance & symmetry
2. Deep generative models on graphs
   - Equivariant graph/molecule generation
   - Graph Variational Autoencoder
   - Protein-binding Ligand generation
   - Multiresolution generative model

Motivation for graph learning

The Anatomy of a Large-Scale Hypertextual Web Search Engine

Sergey Brin and Lawrence Page

Computer Science Department,
Stanford University, Stanford, CA 94305, USA
sergey@cs.stanford.edu and page@cs.stanford.edu

Abstract

In this paper, we present Google, a prototype of a large-scale search engine which makes heavy use of the structure present in hypertext. Google is designed to crawl and index the Web efficiently and produce much more satisfying search results than existing systems. The prototype with a full text and hyperlink database of at least 24 million pages is available at http://google.stanford.edu/
To engineer a search engine is a challenging task. Search engines index tens to hundreds of millions of web pages involving a comparable number of distinct terms. They answer tens of millions of queries every day. Despite the importance of large-scale search engines on the web, very little academic research has been done on them. Furthermore, due to rapid advance in technology and web proliferation, creating a web search engine today is very different from three years ago. This paper provides an in-depth description of our large-scale web search engine -- the first such detailed public description we know of to date. Apart from the problems of scaling traditional search techniques to data of this magnitude, there are new technical challenges involved with using the additional information present in hypertext to produce better search results. This paper addresses this question of how to build a practical large-scale system which can exploit the additional information present in hypertext. Also we look at the problem of how to effectively deal with uncontrolled hypertext collections where anyone can publish anything they want.

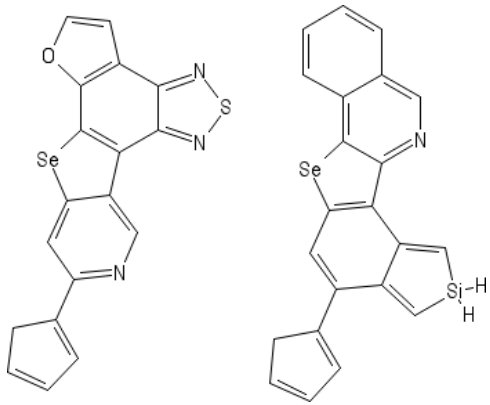
Keywords

World Wide Web, Search Engines, Information Retrieval, PageRank, Google

1. Introduction

(Note: There are two versions of this paper -- a longer full version and a shorter printed version. The full version is available on the web and the conference CD-ROM.)
The web creates new challenges for information retrieval. The amount of information on the web is growing rapidly, as well as the number of new users inexperienced in the art of web research. People are likely to surf the web using its link graph, often starting with high quality human maintained indices such as Yahoo! or with search engines. Human maintained lists cover popular topics effectively but are subjective, expensive to build and maintain, slow to improve, and cannot cover all esoteric topics. Automated search engines that rely on keyword matching usually return too many low quality matches. To make matters worse, some advertisers attempt to gain people's attention by taking measures meant to mislead automated search engines. We have built a large-scale search engine which addresses many of the problems of existing systems. It makes especially heavy use of the additional structure present in hypertext to provide much higher quality search results. We chose our system name, Google, because it is a common spelling of googol, or 10^{100} and fits well with our goal of building very large-scale search

(a) Citation network



(b) Molecules

(c) Knowledge graph

A form of Graph Neural Networks (GNNs)

DFT = Density Functional Theory

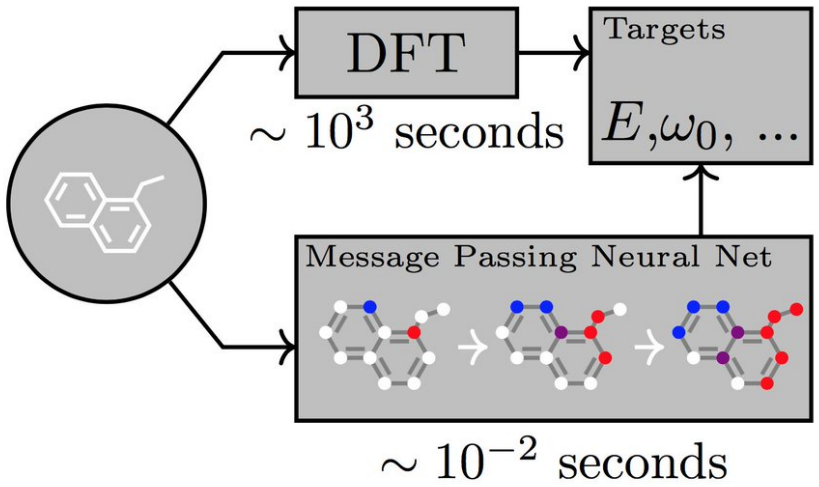


Figure 1. A Message Passing Neural Network predicts quantum properties of an organic molecule by modeling a computationally expensive DFT calculation.

Gilmer et al., *Neural Message Passing for Quantum Chemistry*, ICML 2017

Message Passing Scheme (1)

Given an input graph / network $G = (V, E)$:

1. Initially, each vertex v of the graph is associated with a feature representation ℓ_v (label) or f_v^0. This feature representation can also be called as a *message*.

Message Passing Scheme (1)

Given an input graph / network $G = (V, E)$:

1. Initially, each vertex v of the graph is associated with a feature representation ℓ_v (label) or f_v^0. This feature representation can also be called as a *message*.
2. Iteratively, at iteration t, each vertex collects / aggregates all messages of the previous iteration $\{f_{v_1}^{t-1}, .., f_{v_k}^{t-1}\}$ from other vertices in its neighborhood $\mathcal{N}(v) = \{v_1, .., v_k\}$, and then produces a new message f_v^t via some *hashing function* $\psi(.)$.

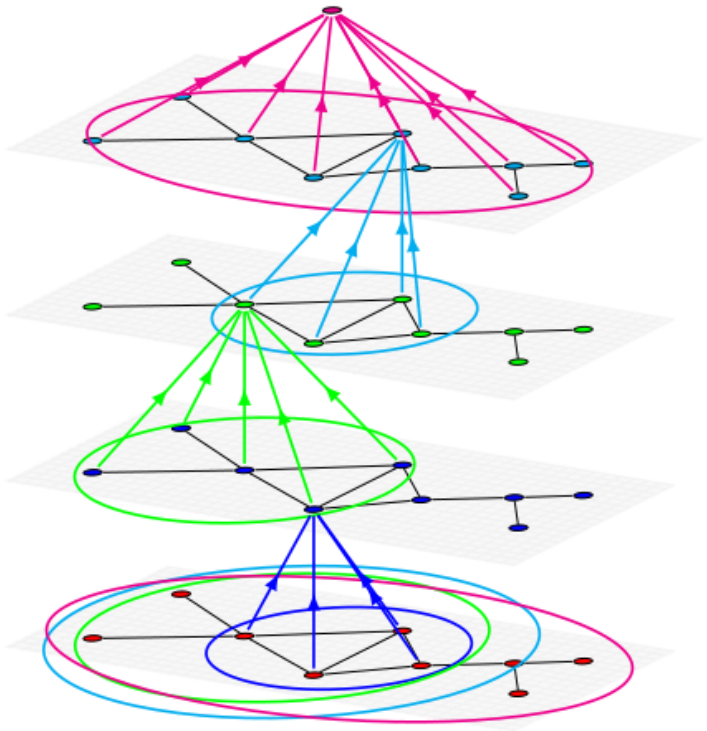
Message Passing Scheme (1)

Given an input graph / network $G = (V, E)$:

1. Initially, each vertex v of the graph is associated with a feature representation ℓ_v (label) or f_v^0. This feature representation can also be called as a *message*.
2. Iteratively, at iteration t, each vertex collects / aggregates all messages of the previous iteration $\{f_{v_1}^{t-1}, .., f_{v_k}^{t-1}\}$ from other vertices in its neighborhood $\mathcal{N}(v) = \{v_1, .., v_k\}$, and then produces a new message f_v^t via some *hashing function* $\psi(.)$.
3. The graph representation $\phi(G)$ is obtained by aggregating all messages in the last iteration of every vertex. $\phi(G)$ is then used for downstream application.

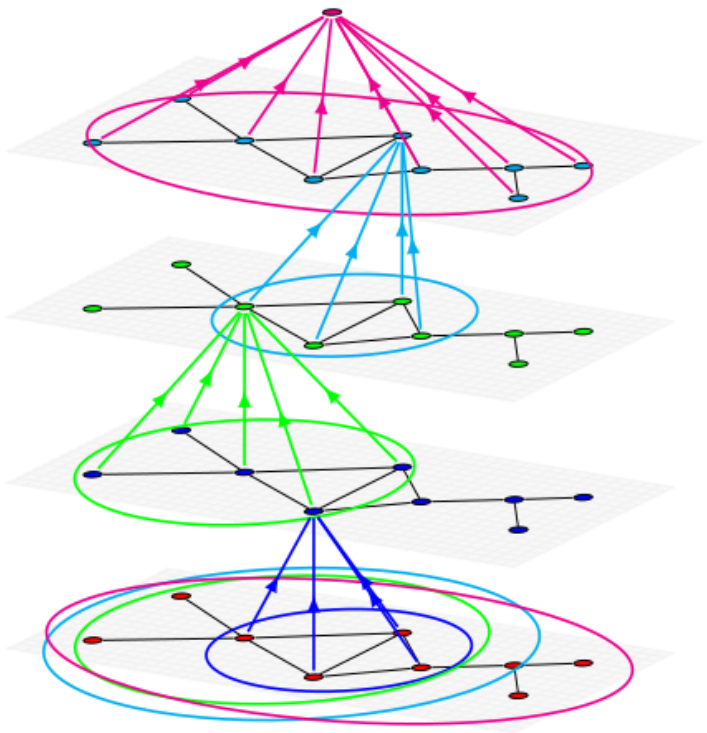
Message Passing Scheme (2)

1: **for** $v \in V$ **do**
2: $\quad f_v^0 \leftarrow \ell_v$
3: **end for**
4: **for** $t = 1 \rightarrow T$ **do**
5: $\quad$ **for** $v \in V$ **do**
6: $\quad\quad f_v^t \leftarrow \psi\left(\{f_i^{t-1}\}_{i \in \mathcal{N}(v)}\right)$
7: $\quad$ **end for**
8: **end for**
9: $\phi(G) \leftarrow \psi\left(\{f_v^T\}_{v \in V}\right)$



Message Passing Scheme (2)

1: **for** $v \in V$ **do**
2: $\quad f_v^0 \leftarrow \ell_v$
3: **end for**
4: **for** $t = 1 \rightarrow T$ **do**
5: $\quad$ **for** $v \in V$ **do**
6: $\quad\quad f_v^t \leftarrow \psi\left(\{f_i^{t-1}\}_{i \in \mathcal{N}(v)}\right)$
7: $\quad$ **end for**
8: **end for**
9: $\phi(G) \leftarrow \psi\left(\{f_v^T\}_{v \in V}\right)$



Note

This procedure is used in Weisfeiler–Lehman **graph isomorphism** test (NP-complete problem).

Message Passing Scheme (3)

With learnable parameters:

for $v \in V$ **do**
  $f_v^0 \leftarrow \ell_v$
end for
for $t = 1 \rightarrow T$ **do**
  for $v \in V$ **do**
    $f_v^t \leftarrow \psi\left(\{f_i^{t-1}\}_{i \in \mathcal{N}(v)}; W^t\right)$
  end for
end for
$\phi(G) \leftarrow \psi\left(\{f_v^T\}_{v \in V}; W^{T+1}\right)$

Message Passing Scheme (3)

With learnable parameters:

1: **for** $v \in V$ **do**
2: $\quad f_v^0 \leftarrow \ell_v$
3: **end for**
4: **for** $t = 1 \rightarrow T$ **do**
5: $\quad$ **for** $v \in V$ **do**
6: $\quad\quad f_v^t \leftarrow \psi\left(\{f_i^{t-1}\}_{i \in \mathcal{N}(v)}; W^t\right)$
7: $\quad$ **end for**
8: **end for**
9: $\phi(G) \leftarrow \psi\left(\{f_v^T\}_{v \in V}; W^{T+1}\right)$

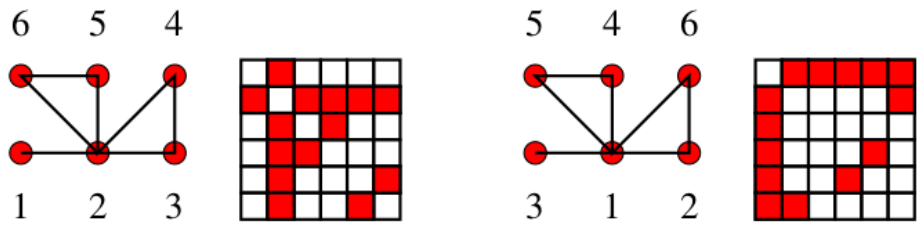
Given a graph properties $y(G) \in \mathbb{R}^d$ to regress, we have the optimization:

$$\min_{\{W^t\}_{t=1}^{T+1}} ||y(G) - \phi(G)||_2^2$$

The gradient with respect to $\{W^t\}_{t=1}^{T+1}$ can be computed via Back-propagation.

Invariance

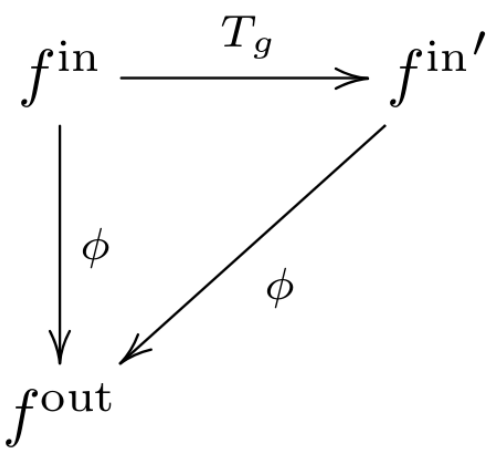
We renumber the vertices by a permutation $\sigma : \{1, 2, .., 6\} \mapsto \{1, 2, .., 6\}$. The adjacency matrices of G (left) and G' (right) are different, but topologically they represent the same graph:



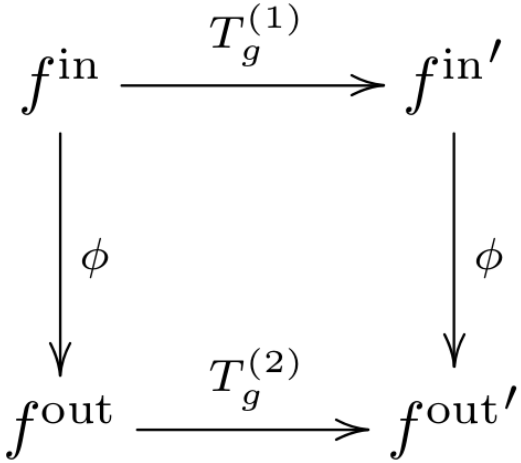
Therefore, ϕ must be **invariant** wrt permutation, i.e. $\phi(G) = \phi(G')$.

Invariance vs. Equivariance

T_g is an action of a group G on the space of inputs and outputs. In case of graphs, G is the symmetry group $\mathbb{S}_n$.



Invariance: $\phi(T_g(f)) = \phi(f)$



Equivariance: $\phi(T_g^{(1)}(f)) = T_g^{(2)}(\phi(f))$

Message Passing Neural Networks

To preserve the **permutation invariance**, the aggregation function ψ of MPNNs basically sums the messages from each node's neighborhood. The algorithm is simply expressed in matrix form as:

$$F^t = \sigma(AF^{t-1}W^t)$$

where:

- $A \in \{0,1\}^{n \times n}$ is the adjacency matrix.
  (or graph Laplacian $I_n - D^{-1/2}AD^{-1/2}$)
- $F^t \in \mathbb{R}^{n \times d}$ is the node feature matrix.
- $W^t \in \mathbb{R}^{d \times d'}$ is the weight (channels mixing) matrix, that is learnable.
- σ is the nonlinearity.

Limitations of GNNs

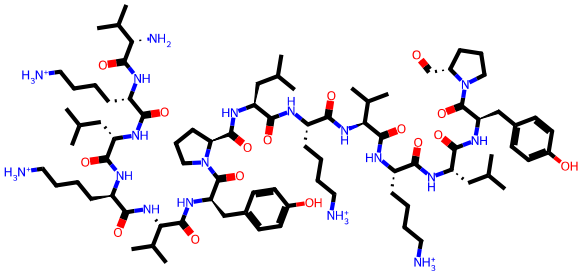
GNNs are powerful but have several limitations theoretically and practically:

- **Long-range** modeling (i.e. graphs with large diameters) [1, 2]
- Modeling highly **symmetric** structures [3, 4]
- Over-smoothing & Over-squashing (for another talk!)

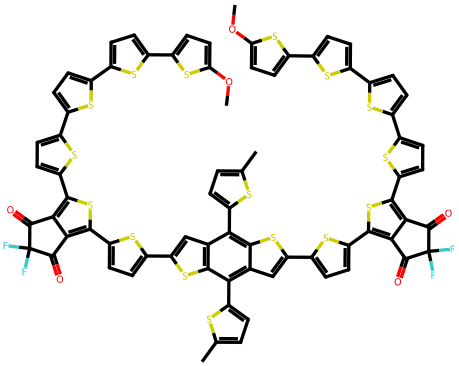
Our works:

1. On the Connection Between MPNN and Graph Transformer, ICML 2023
2. Multiresolution graph transformers and wavelet positional encoding for learning long-range and hierarchical structures, Journal of Chemical Physics
3. Predicting molecular properties with covariant compositional networks, Journal of Chemical Physics
4. Covariant compositional networks for learning graphs, ICLR 2018

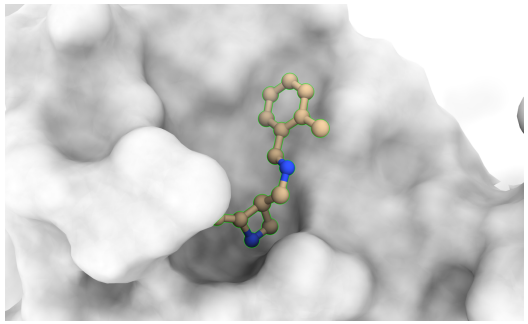
Long-range graphs



(a) Peptide



(b) Polymer



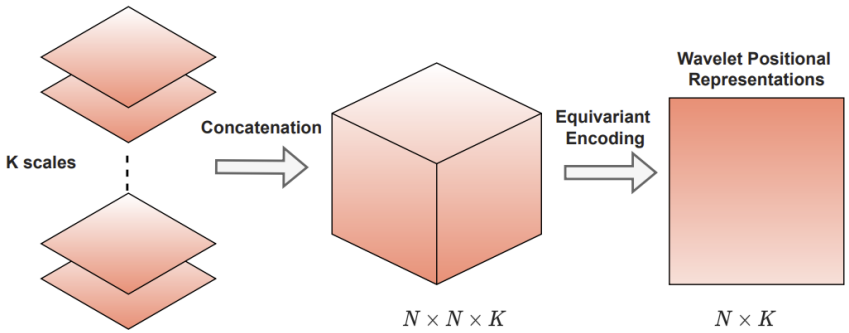
(c) Protein-Ligand

Figure: Macromolecules that are actually long-range graphs.

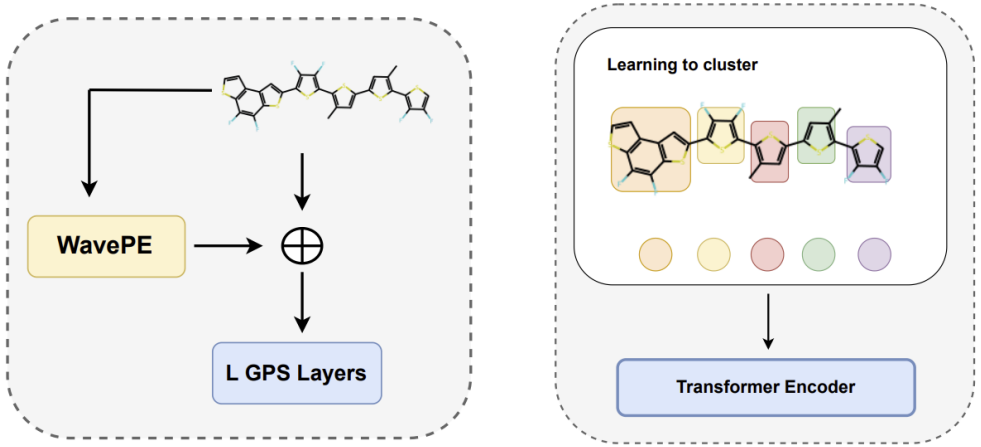
- Macromolecules have **hierarchical structures** and comprise multiple **long-range** dependencies among distant atoms.
- We want to predict functions of peptides, properties of polymers calculated from Density Functional Theory, and protein-ligand binding affinity.

New proposal for long-range graphs

Wavelet positional encoding



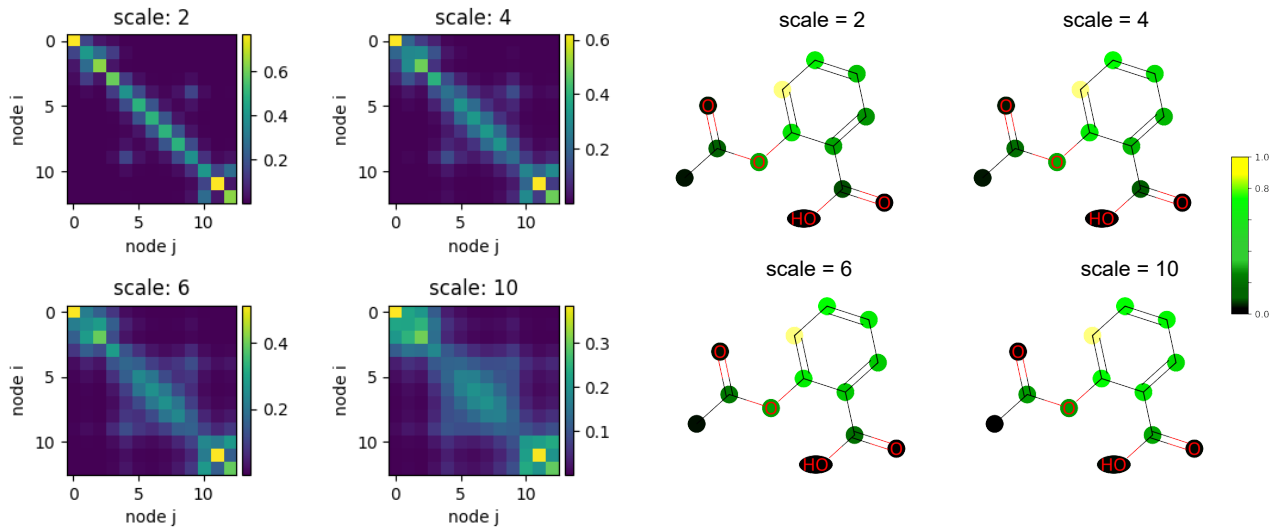
Multiresolution Graph Transformer



- Transformers are effective for computing the interactions between distant atoms via **self-attention** mechanisms.
- To adapt Transformer-like architectures to graphs, we need **positional encoding** schemes that embody the local structures.

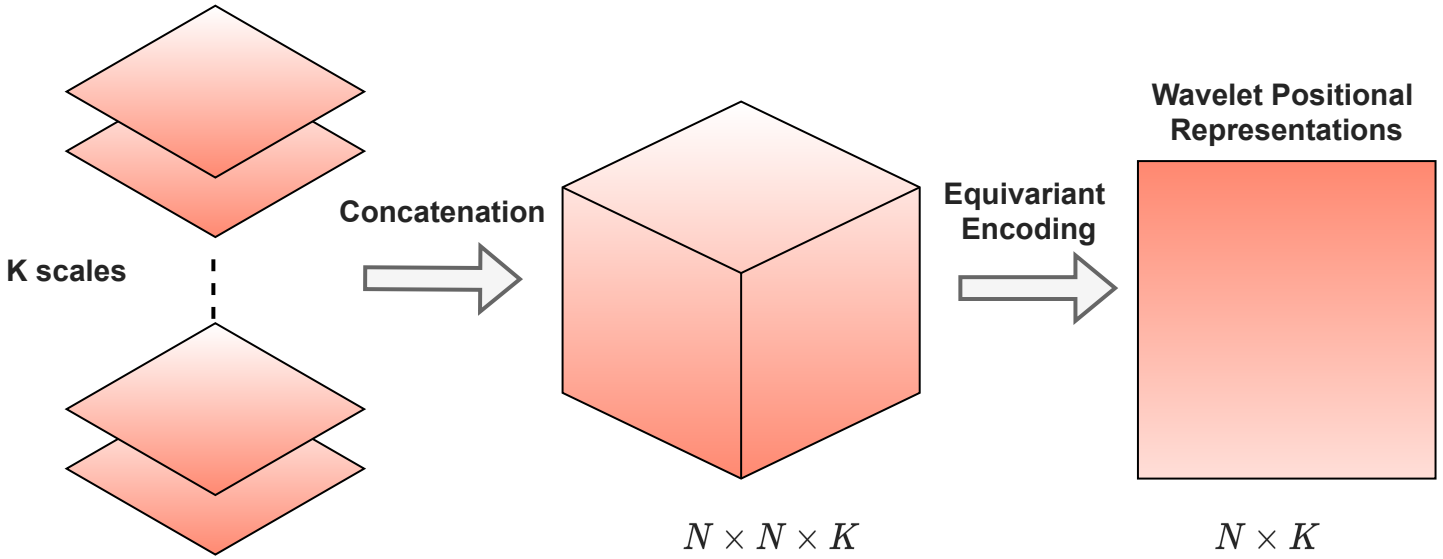
Wavelet Positional Encoding

- We compute the spectral wavelets at multiple scales.
- Low-scale wavelets are highly localized, whereas the high-scale wavelets can spread out more nodes on the molecular graphs.



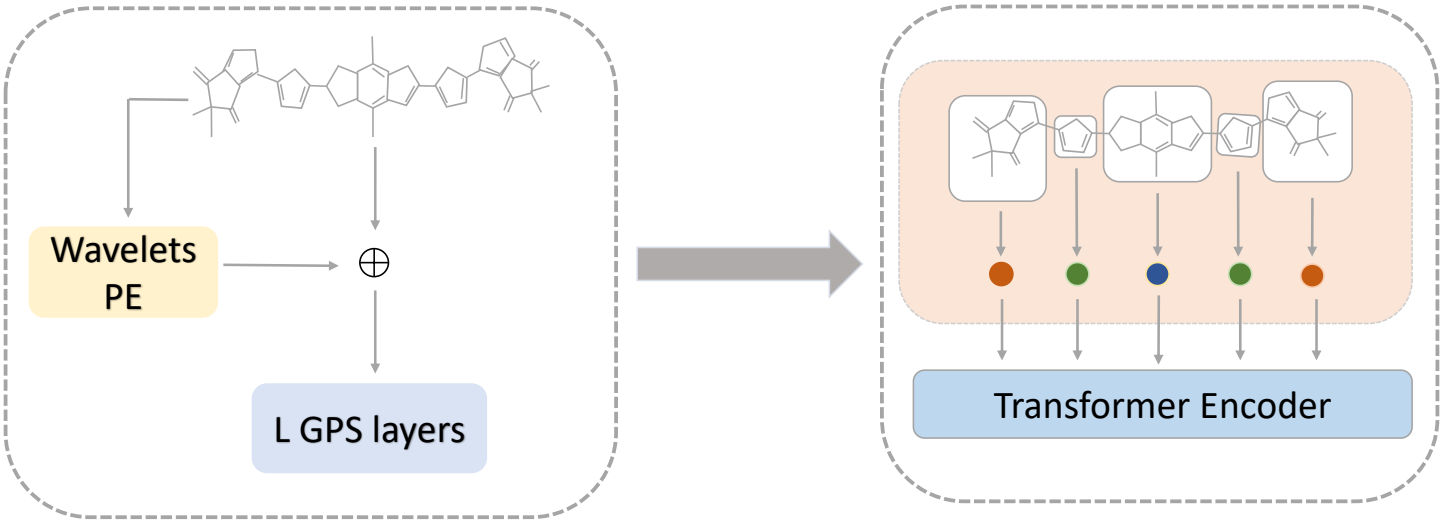
Visualization of some of the wavelets with scaling parameters on the Aspirin $C_9H_8O_4$ molecular graph with 13 nodes (i.e. heavy atoms).

Wavelet Positional Encoding (continued)



- We concatenate the diffusion matrices at different scales together to form a tensor.
- Then, we encode this tensor to node-level positional embeddings via permutation-equivariant neural networks.

Multiresolution Graph Transformer (MGT)



- We propose a **learning-to-cluster** algorithm that coarsens graphs iteratively to build a multiresolution (i.e. multiple of resolutions) representation of the input graph.
- We employ the **graph transformer** model learning on each resolution and we integrate our **wavelet positional encoding**.

Peptides Property Prediction

We outperform other state-of-the-art methods in predicting peptides' functionality.

Model	Params	Peptides-struct	Peptides-func
		MAE ↓	AP ↑
GCN	508k	0.3496 ± 0.0013	0.5930 ± 0.0023
GINE	476k	0.3547 ± 0.0045	0.5498 ± 0.0079
GatedGCN	509k	0.3420 ± 0.0013	0.5864 ± 0.0077
GatedGCN + RWPE	506k	0.3357 ± 0.0006	0.6069 ± 0.0035
Transformer + LapPE	488k	0.2529 ± 0.0016	0.6326 ± 0.0126
GPS	—	0.2500 ± 0.0005	0.6535 ± 0.0041
SAN + LapPE	493k	0.2683 ± 0.0043	0.6384 ± 0.0121
SAN + RWPE	500k	0.2545 ± 0.0012	0.6562 ± 0.0075
MGT + WavePE (ours)	499k	**0.2453 ± 0.0025**	**0.6817 ± 0.0064**

Polymer Property Prediction

We **achieve the chemical accuracy** in estimating the molecular properties of polymers that are calculated from Density Functional Theory. We also outperform all other competitive baselines.

Model	Params	Property		
		GAP	HOMO	LUMO
DFT error		1.2	2.0	2.6
Chemical accuracy		0.043	0.043	0.043
GCN	527k	0.1094 ± 0.0020	0.0648 ± 0.0005	0.0864 ± 0.0014
GCN + Virtual Node	557k	0.0589 ± 0.0004	0.0458 ± 0.0007	0.0482 ± 0.0010
GINE	527k	0.1018 ± 0.0026	0.0749 ± 0.0042	0.0764 ± 0.0028
GINE + Virtual Node	557k	0.0870 ± 0.0040	0.0565 ± 0.0050	0.0524 ± 0.0010
GPS	600k	0.0467 ± 0.0010	0.0322 ± 0.0020	0.0385 ± 0.0006
Transformer + LapPE	700k	0.2949 ± 0.0481	0.1200 ± 0.0206	0.1547 ± 0.0127
MGT + LapPE (ours)	499k	**0.0378 ± 0.0004**	**0.0270 ± 0.0010**	0.0300 ± 0.0006
MGT + RWPE (ours)	499k	0.0384 ± 0.0015	0.0274 ± 0.0005	**0.0290 ± 0.0007**
MGT + WavePE (ours)	499k	0.0387 ± 0.0011	0.0283 ± 0.0004	0.0290 ± 0.0010

Protein-Ligand Binding Affinity Prediction

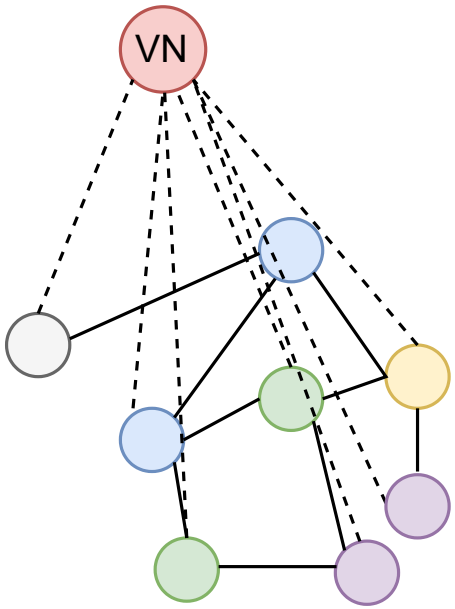
Understanding the multiscale structure of proteins is important in estimating their fitness and functionality. In this experiment, we show the effectiveness of our model in capturing the long-range and hierarchical structures of proteins. We show our competitive experimental result on ATOM3D benchmark.

Method	3D-CNN	GNN	ENN	GVP-GNN	MGT + WavePE
RMSE ↓	**1.416 ± 0.021**	1.570 ± 0.025	1.568 ± 0.012	1.594 ± 0.073	1.436 ± 0.066

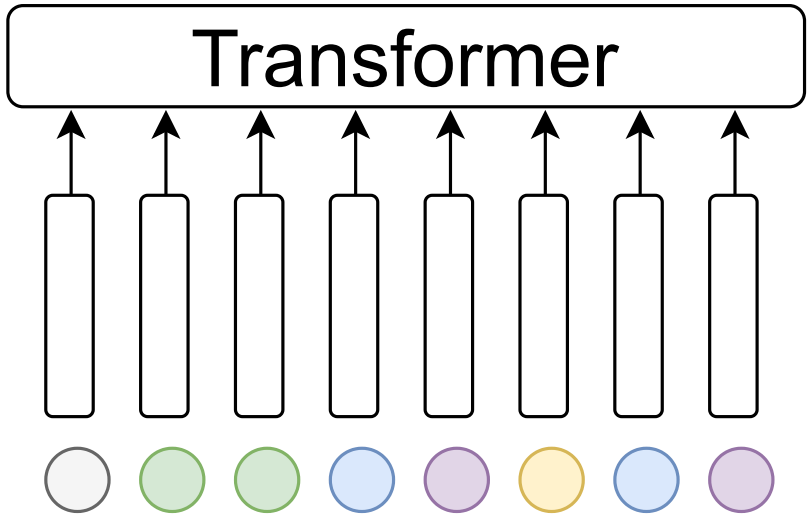
Software & Datasets

`https://github.com/HySonLab/Multires-Graph-Transformer`

Is there an alternative to Graph Transformer?



(a)



(b)

(a) MPNN + VN = we augment the graph with a virtual node (VN) connecting to all other nodes. VN acts as a "bridge" that reduces the maximum shortest path to 2.

(b) Graph Transformer = we treat each node embedding as a token and apply a Transformer on the sequence of node embeddings/tokens.

Our theoretical results

From our paper **On the Connection Between MPNN and Graph Transformer** at ICML 2023:

Theorem 1

MPNN + VN can simulate (not just approximate) equivariant DeepSets: $\mathbb{R}^{n\times d} \to \mathbb{R}^{n\times d}$. This implies that MPNN + VN of $\mathcal{O}(1)$ depth and $\mathcal{O}(n^d)$ width is permutation equivariant universal, and can approximate self-attention layer and transformers arbitrarily well.

Our theoretical results

From our paper **On the Connection Between MPNN and Graph Transformer** at ICML 2023:

Theorem 1

MPNN + VN can simulate (not just approximate) equivariant DeepSets: $\mathbb{R}^{n\times d} \to \mathbb{R}^{n\times d}$. This implies that MPNN + VN of $\mathcal{O}(1)$ depth and $\mathcal{O}(n^d)$ width is permutation equivariant universal, and can approximate self-attention layer and transformers arbitrarily well.

Theorem 2

Given any graph G of size n with node features $\boldsymbol{X} \in \mathcal{X}$, and a self-attention layer $\boldsymbol{L}$ on G (fix $\boldsymbol{W}_K, \boldsymbol{W}_Q, \boldsymbol{W}_V$), there exists a $\mathcal{O}(n)$ layer of heterogeneous MPNN + VN with the specific aggregate/update/message function that can approximate $\boldsymbol{L}$ on $\mathcal{X}$ arbitrarily well.

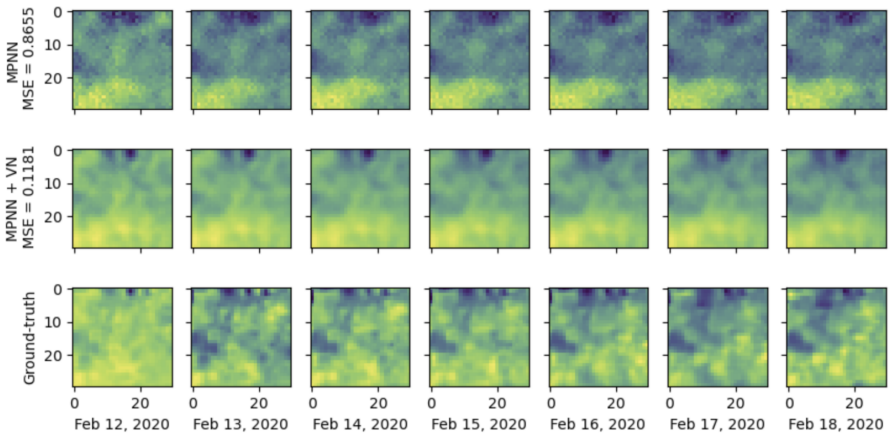
MPNN + VN on Long-Range Graph Benchmark (LRGB)

Model	# Params.	Peptides-functional		Peptides-structural	
		Test AP before VN	Test AP after VN ↑	Test MAE before VN	Test MAE after VN ↓
GCN	508k	0.5930±0.0023	0.6623±0.0038	0.3496±0.0013	**0.2488±0.0021**
GINE	476k	0.5498±0.0079	0.6346±0.0071	0.3547±0.0045	0.2584±0.0011
GatedGCN	509k	0.5864±0.0077	0.6635±0.0024	0.3420±0.0013	0.2523±0.0016
GatedGCN+RWSE	506k	0.6069±0.0035	**0.6685±0.0062**	0.3357±0.0006	0.2529±0.0009
Transformer+LapPE	488k	0.6326±0.0126	-	0.2529±0.0016	-
SAN+LapPE	493k	0.6384±0.0121	-	0.2683±0.0043	-
SAN+RWSE	500k	0.6439±0.0075	-	0.2545±0.0012	-

- Peptides-functional and Peptides-structural are two datasets of LRGB.
- Previously GT shows a large margin over MPNN.
- **Simply adding VN is enough to make MPNN outperform GT!**

MPNN + VN for climate modeling

We apply our MPNN + VN model to forecast daily **sea surface temperature** (SST) in the Pacific Ocean from 1982 to 2021, given 6 weeks of history to predict the next 1, 2 and 4 weeks of temperatures. The input is a grid graph of 30 longitudes and 30 latitudes at 0.5°-degree resolution. We report the error with Mean Square Error (MSE) metric.

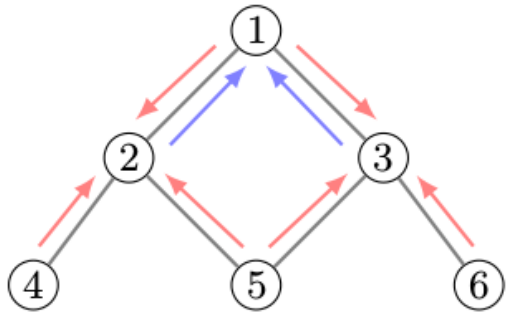


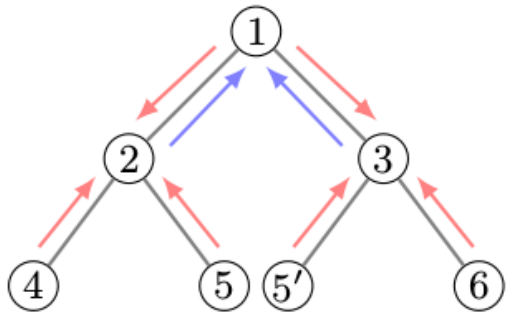
Model	4 weeks	2 weeks	1 week
MLP	0.3302	0.2710	0.2121
TF-Net	0.2833	**0.2036**	**0.1462**
Linear Transformer + LapPE	0.2818	0.2191	0.1610
MPNN	0.2917	0.2281	0.1613
MPNN + VN	**0.2806**	0.2130	0.1540

Limitation of GNNs on highly symmetric structures

The summing operator **limits** the representative power of MPNNs such that each node loses their identity after being aggregated. For example:

These two graphs are **not** isomorphic, but message passing scheme **fails** to distinguish whether 5 and 5' are the same vertex or not.

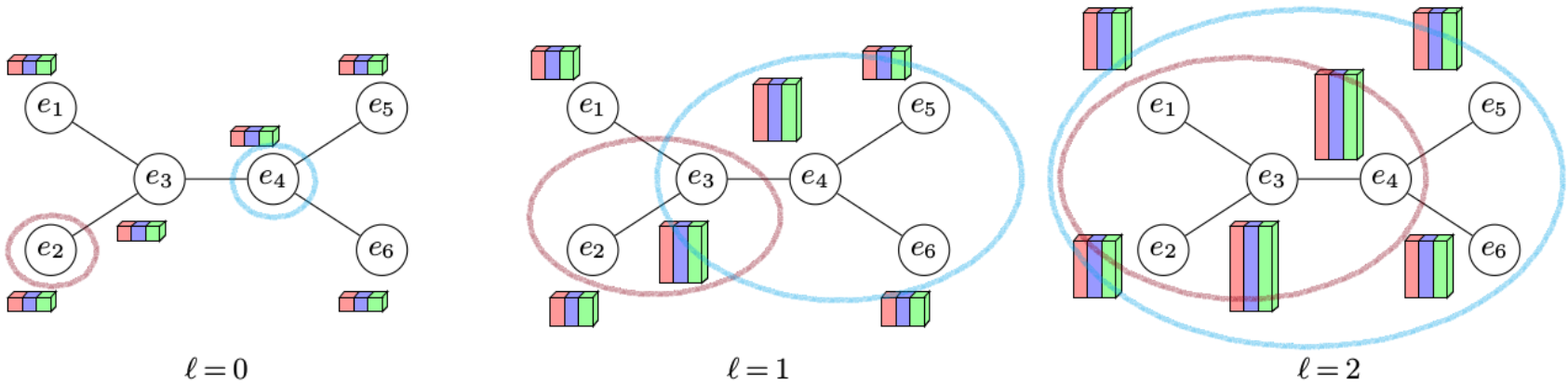




Weisfeiler-Lehman isomorphism test fails for highly symmetric structures such as regular graphs.

Covariant Compositional Networks (1)

We propose a new general architecture called **Covariant Compositional Networks** (CCNs) in which the messages are represented by higher order tensors and transform covariantly/equivariantly according to a specific representation of the symmetry group of its receptive field.

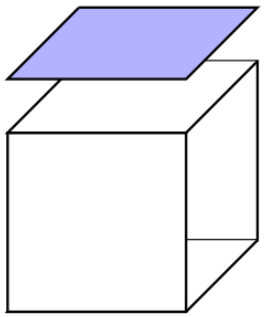


Feature tensors in a **first order** CCN for ethylene (C_2H_4) assuming three channels (red, green, blue).

Covariant Compositional Networks (2)

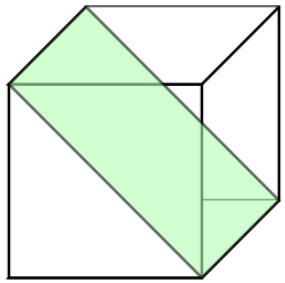
Permutation covariant operators:

1. Projections



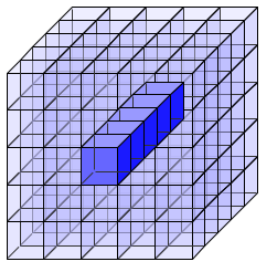
$$C_{i,j} = \sum_a A_{a,i,j}$$

2. Diagonals



$$C_{i,j} = \sum_i A_{i,i,j}$$

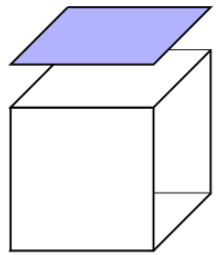
3. Contractions



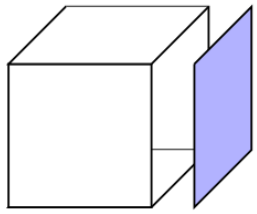
$$C_k = \sum_{i,j} A_{i,j,k}$$

Covariant Compositional Networks (3)

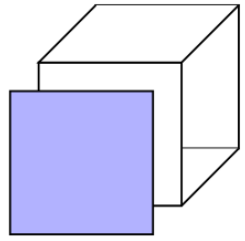
There are six different ways of covariantly reducing (contracting) a third order tensor to a second order tensor:



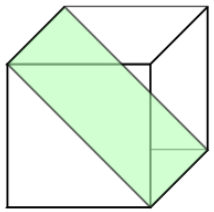
$$C_{i,j} = \sum_a A_{a,i,j}$$



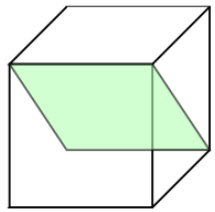
$$C_{i,j} = \sum_j A_{i,a,j}$$



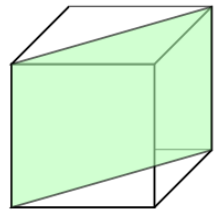
$$C_{i,j} = \sum_k A_{i,j,a}$$



$$C_{i,j} = \sum_i A_{i,i,j}$$



$$C_{i,j} = \sum_i A_{i,j,i}$$

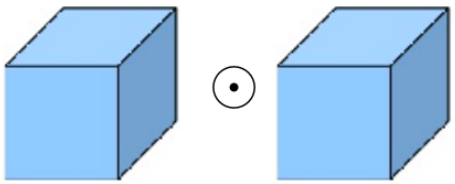


$$C_{i,j} = \sum_i A_{i,j,j}$$

Covariant Compositional Networks (4)

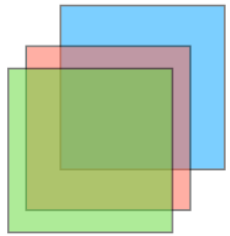
Permutation covariant operators (continued):

4. Hadamard products



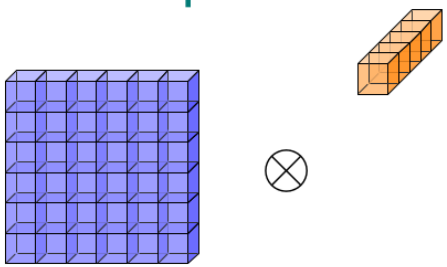
$$C_{i,j,k} = A_{i,j,k} B_{i,j,k}$$

5. Stacking



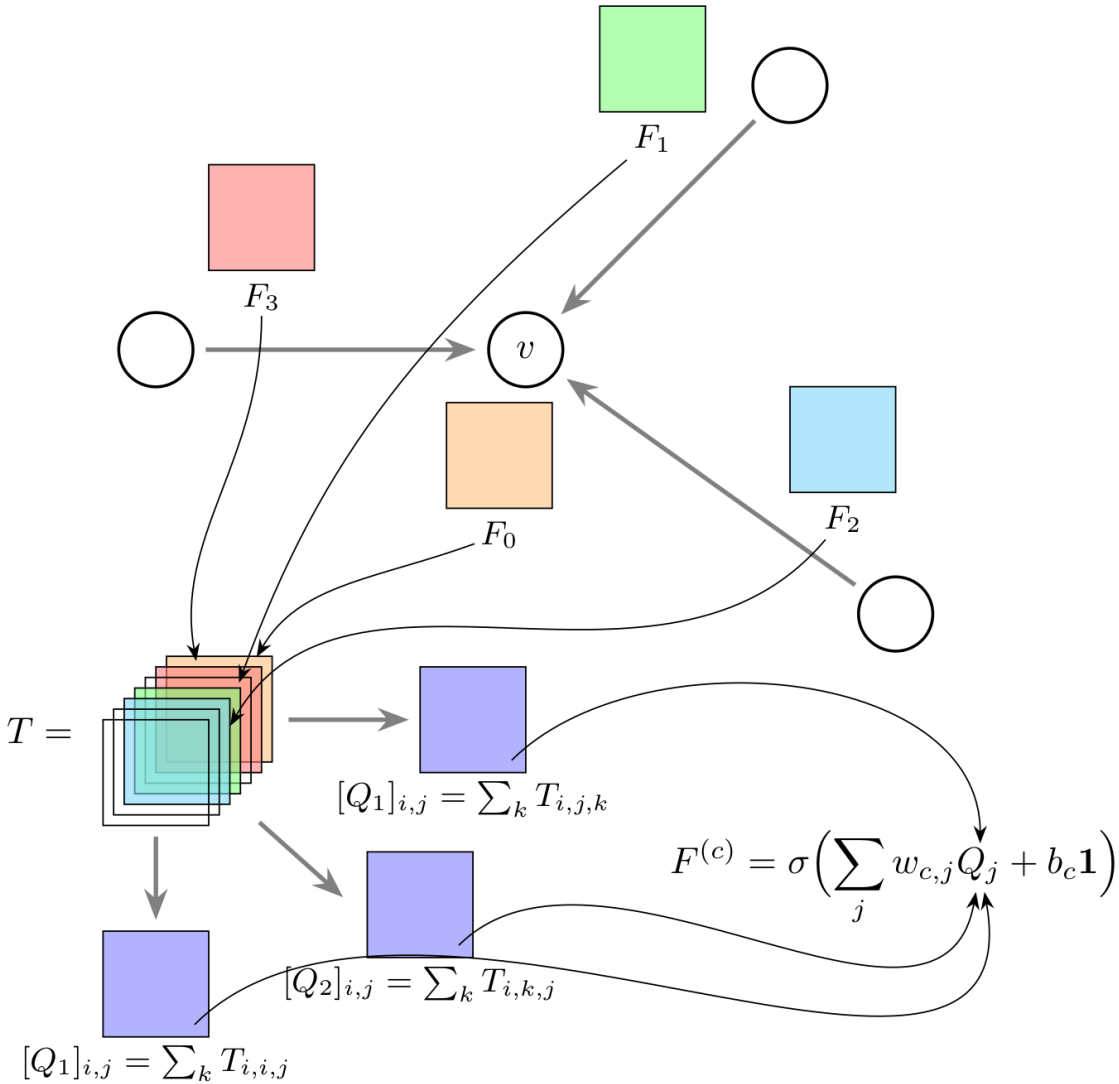
$$C_{i,j,k} = A^{(k)}_{i,j}$$

6. Tensor products



$$C_{i,j,k} = A_{i,j} B_k$$

Covariant Compositional Networks (5)

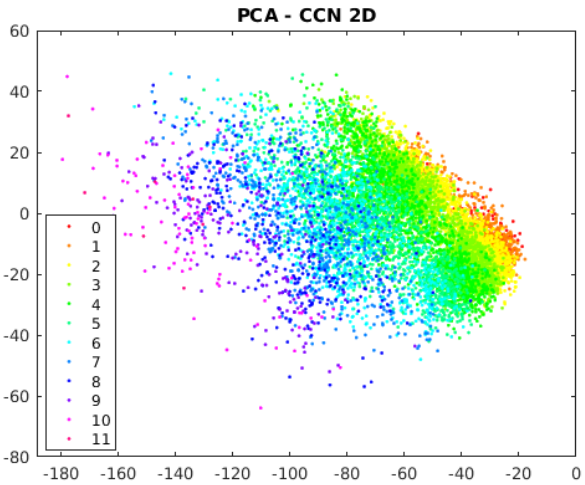


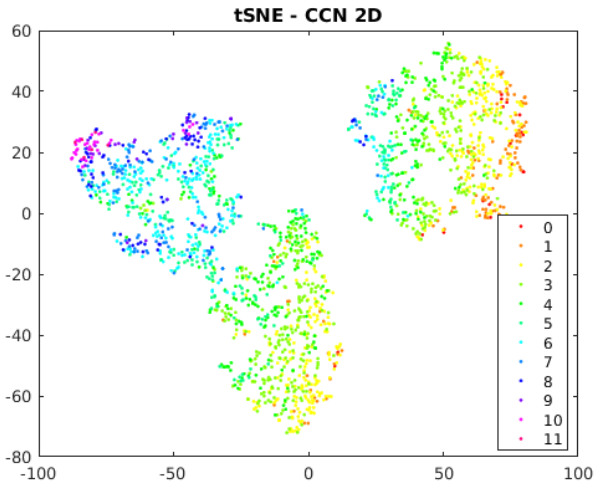
Second-order CCN

Covariant Compositional Networks (6)

Regression results on Havard Clean Energy Project (CEP) dataset

	Test MAE	Test RMSE
Lasso	0.867	1.437
Ridge regression	0.854	1.376
Random forest	1.004	1.799
Gradient boosted trees	0.704	1.005
Weisfeiler–Lehman kernel[33]	0.805	1.096
Neural graph fingerprints[21]	0.851	1.177
PSCN ($k = 10$)[37]	0.718	0.973
Second order CCN (our method)	**0.340**	**0.449**





Covariant Compositional Networks (7)

Regression results on QM9 dataset

	CCN	DFT error
α (Bohr^3)	**0.22**	0.4
C_v (cal/(mol K))	**0.07**	0.34
G (eV)	**0.06**	0.1
GAP (eV)	**0.12**	1.2
H (eV)	**0.06**	0.1
HOMO (eV)	**0.09**	2.0
LUMO (eV)	**0.09**	2.6
μ (Debye)	0.48	**0.1**
ω_1 (cm^{-1})	**2.81**	28
R_2 (Bohr^2)	4.00	-
U (eV)	**0.06**	0.1
U_0 (eV)	**0.05**	0.1
ZPVE (eV)	**0.0039**	0.0097

(Hy et al., 2018, Kondor et al., 2018)

Software

`https://github.com/HyTruongSon/LibCCNs`
`https://github.com/HyTruongSon/InvariantGraphNetworks-PyTorch`

Graph generation (1)

Question

How to generate new/unseen graphs in an **equivariant** and **multiresolution** manner? Why equivariance is important for graph generation?

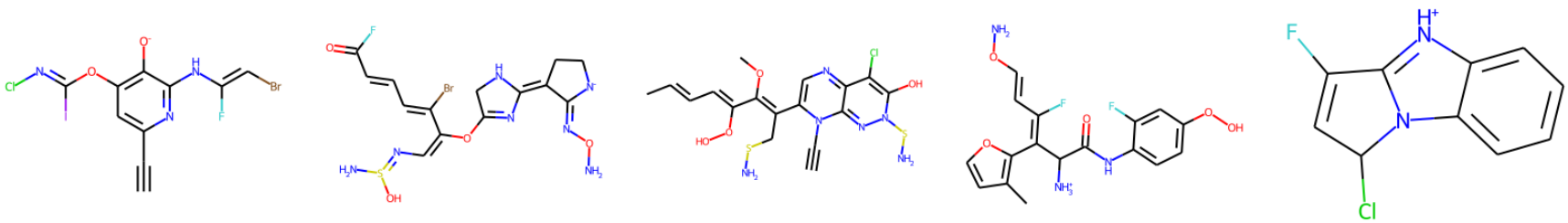
Our works:

1. Multiresolution Equivariant Graph Variational Autoencoder, Machine Learning: Science and Technology
2. Target-aware Variational Auto-encoders for Ligand Generation with Multimodal Protein Representation Learning, Machine Learning in Structural Biology @ NeurIPS 2023
3. The general theory of permutation equivarant neural networks and higher order graph variational encoders, Preprint

Graph generation (2)

Goal

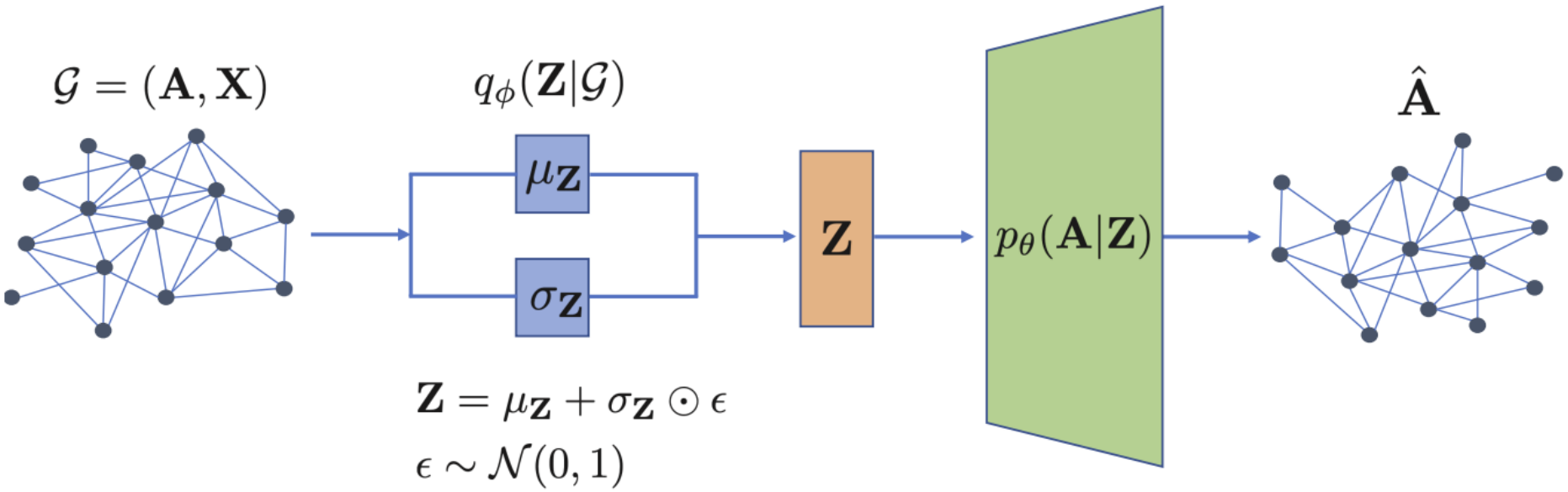
Design models that can observe a set of graphs $\{\mathcal{G}_1, .., \mathcal{G}_n\}$ and learn to generate graphs with similar characteristics as this training set.



Look-alike molecules generated from Multiresolution VAE trained on ZINC dataset

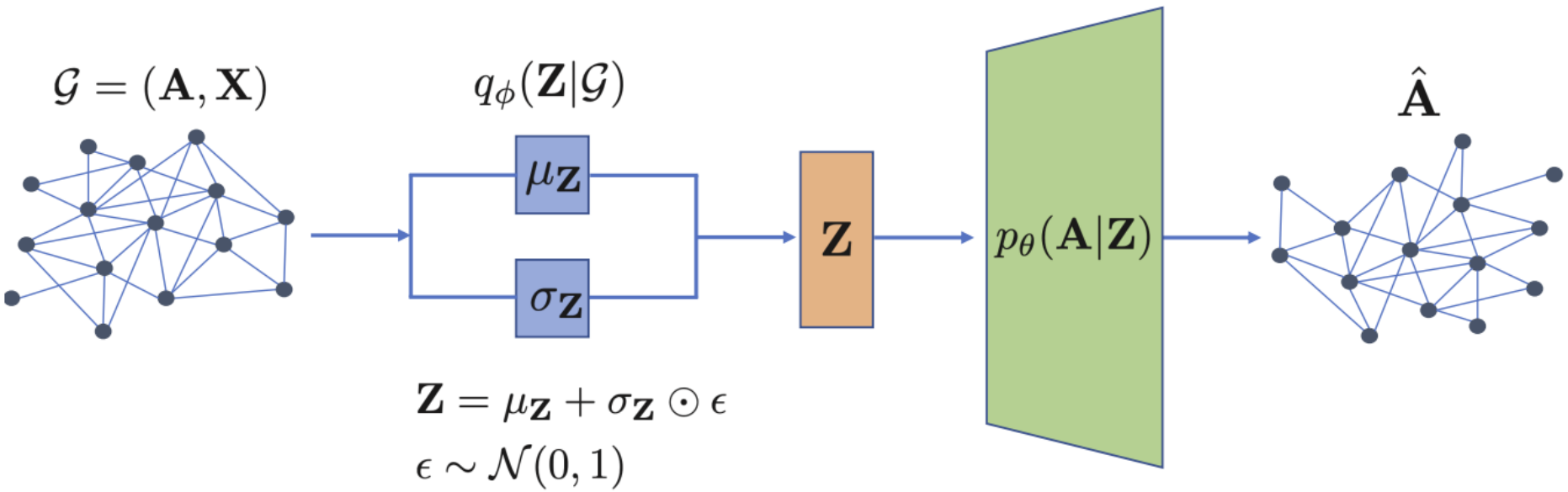
Graph Variational Autoencoder (1)

- **Probabilistic encoder** $q_\theta(\boldsymbol{Z}|\mathcal{G})$ defines a distribution over *latent representations*. We specify the latent conditional distribution as $\boldsymbol{Z} \sim \mathcal{N}(\mu_\theta(\mathcal{G}), \sigma_\theta(\mathcal{G}))$, where μ_θ and σ_θ are neural networks that output the mean and variance parameters for a normal distribution.



Graph Variational Autoencoder (2)

- **Probabilistic decoder** $p_\theta(\boldsymbol{A}|\boldsymbol{Z})$, from which we can sample realistic graphs (i.e. adjacency matrices) $\hat{\boldsymbol{A}} \sim p_\theta(\boldsymbol{A}|\boldsymbol{Z})$ by conditioning on a latent variable $\boldsymbol{Z}$.
- **Prior distribution** $p(\boldsymbol{Z})$ over the latent space. Assume $\boldsymbol{Z} \sim \mathcal{N}(\mathbf{0}, \mathbf{1})$.



Graph Variational Autoencoder (3)

Given a set of training graphs $\{\mathcal{G}_1, .., \mathcal{G}_n\}$, we can train a VAE model by maximizing the evidence likelihood lower bound (ELBO):

$$\mathcal{L} = \sum_{\mathcal{G}_i \in \{\mathcal{G}_1, .., \mathcal{G}_n\}} \mathbb{E}_{q_\theta(\boldsymbol{Z}|\mathcal{G}_i)}[p_\theta(\mathcal{G}_i|\boldsymbol{Z})] - \mathrm{KL}(q_\theta(\boldsymbol{Z}|\mathcal{G}_i)||p(\boldsymbol{Z}))$$

Basic idea:

- $\mathbb{E}_{q_\theta(\boldsymbol{Z}|\mathcal{G}_i)}[p_\theta(\mathcal{G}_i|\boldsymbol{Z})]$: Maximize the reconstruction ability of our decoder.
- $\mathrm{KL}(q_\theta(\boldsymbol{Z}|\mathcal{G}_i)||p(\boldsymbol{Z}))$: Minimize the KL-divergence between our posterior latent distribution $q_\theta(\boldsymbol{Z}|\mathcal{G}_i)$ and the prior $p(\boldsymbol{Z})$.

Why do we need equivariant generation? (1)

Suppose that we want to map the latent vector $\boldsymbol{z}_{\mathcal{G}}$ to a matrix $\hat{\boldsymbol{A}} \in [0,1]^{|\mathcal{V}|\times|\mathcal{V}|}$ of edge probabilities. The posterior distribution:

$$p_\theta(\mathcal{G}|\boldsymbol{z}_{\mathcal{G}}) = \prod_{(u,v)\in\mathcal{V}\times\mathcal{V}} \hat{\boldsymbol{A}}_{u,v}\boldsymbol{A}_{u,v} + (1-\hat{\boldsymbol{A}}_{u,v})(1-\boldsymbol{A}_{u,v})$$

where $\boldsymbol{A}$ denotes the true adjacency, and $\hat{\boldsymbol{A}}$ denotes the predicted edge probabilities.

Problem

But we do **not** know the correct ordering of nodes.

Why do we need equivariant generation? (2)

Graph matching problem (**NP-hard**, quadratic assignment problem):

$$p_\theta(\mathcal{G}|\boldsymbol{z}_\mathcal{G}) = \max_{\pi \in \Pi} \prod_{(u,v)\in\mathcal{V}\times\mathcal{V}} \hat{\boldsymbol{A}}^{\pi}_{u,v}\boldsymbol{A}_{u,v} + (1-\hat{\boldsymbol{A}}^{\pi}_{u,v})(1-\boldsymbol{A}_{u,v})$$

Approximate solution: Specify a set of particular orderings $\{\pi_1, .., \pi_n\}$ (this is also how autoregressive methods work in practice)

$$p_\theta(\mathcal{G}|\boldsymbol{z}_\mathcal{G}) \approx \sum_{\pi_i \in \{\pi_1,..,\pi_n\}} \prod_{(u,v)\in\mathcal{V}\times\mathcal{V}} \hat{\boldsymbol{A}}^{\pi_i}_{u,v}\boldsymbol{A}_{u,v} + (1-\hat{\boldsymbol{A}}^{\pi_i}_{u,v})(1-\boldsymbol{A}_{u,v})$$

Why do we need equivariant generation? (2)

Graph matching problem (**NP-hard**, quadratic assignment problem):

$$p_\theta(\mathcal{G}|\boldsymbol{z}_\mathcal{G}) = \max_{\pi \in \Pi} \prod_{(u,v)\in\mathcal{V}\times\mathcal{V}} \hat{\boldsymbol{A}}^{\pi}_{u,v}\boldsymbol{A}_{u,v} + (1-\hat{\boldsymbol{A}}^{\pi}_{u,v})(1-\boldsymbol{A}_{u,v})$$

Approximate solution: Specify a set of particular orderings $\{\pi_1, .., \pi_n\}$ (this is also how autoregressive methods work in practice)

$$p_\theta(\mathcal{G}|\boldsymbol{z}_\mathcal{G}) \approx \sum_{\pi_i \in \{\pi_1,..,\pi_n\}} \prod_{(u,v)\in\mathcal{V}\times\mathcal{V}} \hat{\boldsymbol{A}}^{\pi_i}_{u,v}\boldsymbol{A}_{u,v} + (1-\hat{\boldsymbol{A}}^{\pi_i}_{u,v})(1-\boldsymbol{A}_{u,v})$$

Almost perfect solution

- Equivariant (higher-order) latent, encoder, and decoder
- Thiede, Hy & Kondor, 2020

Markov Random Fields

Problem with the prior

- $\mathcal{N}(0, 1)$ is not a good prior for graph generation, because each node (atom)'s latent is sampled **independently**.
- We want a new prior $\mathcal{N}(\mu, \Sigma)$ that is both **learnable** and **equivariant**.
- That also requires a new **reparameterization trick**.
- Hy & Kondor, 2021

Markov network

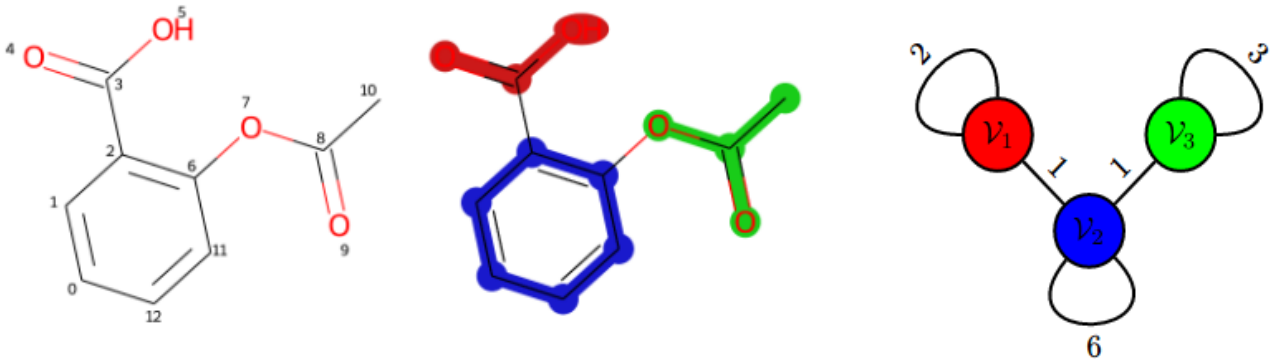
In general, k-th order graph encoders encode an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ into a k-th order latent $\boldsymbol{z} \in \mathbb{R}^{n^k \times d_z}$, with learnable parameters $\boldsymbol{\theta}$, can be represented as a parameterized Markov Random Field (MRF) or Markov network.

Multiresolution Graph Network (1)

What we want more?

Learning and then generating graphs in multiple levels of granulity.

The backbone of this coarse-graining architecture, Multiresolution Graph Network (MGN), is the **Learning to cluster** algorithm. The hard clustering can be differentible (for back-propagation) by the Gumbel-softmax trick.



Aspirin $C_9H_8O_4$, its 3-cluster partition and the corresponding coarsen graph.

Multiresolution Graph Network (2)

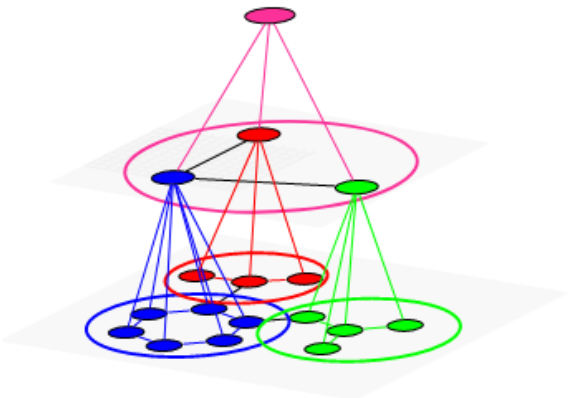
It is desirable to have a *balanced* K-cluster partition in which clusters $\mathcal{V}_1^{(\ell)}, .., \mathcal{V}_K^{(\ell)}$ (at the ℓ-th resolution level) have similar sizes that are close to $|\mathcal{V}^{(\ell)}|/K$.

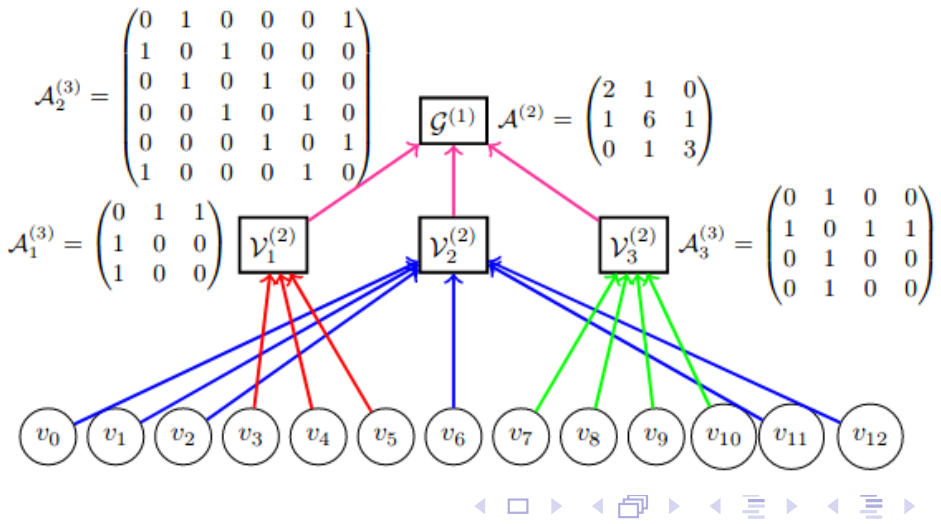
Multiresolution Graph Network (2)

It is desirable to have a *balanced* K-cluster partition in which clusters $\mathcal{V}_1^{(\ell)}, .., \mathcal{V}_K^{(\ell)}$ (at the ℓ-th resolution level) have similar sizes that are close to $|\mathcal{V}^{(\ell)}|/K$.

We enforce the clustering procedure to produce a balanced cut by minimizing the following Kullback–Leibler divergence:

$$\mathcal{D}_{KL}(P||Q) = \sum_{k=1}^{K} P(k) \log \frac{P(k)}{Q(k)}, \quad P = \left(\frac{|\mathcal{V}_1^{(\ell)}|}{|\mathcal{V}^{(\ell)}|}, .., \frac{|\mathcal{V}_K^{(\ell)}|}{|\mathcal{V}^{(\ell)}|} \right), \quad Q = \left(\frac{1}{K}, .., \frac{1}{K} \right)$$





Multiresolution Equivariant Graph VAE (1)

Based on the construction of multiresolution graph network, the latent hierarchy is partitioned into disjoint groups, $\mathcal{Z}_i = \{\mathcal{Z}_i^{(1)}, \mathcal{Z}_i^{(2)}, .., \mathcal{Z}_i^{(L)}\}$ where $\mathcal{Z}_i^{(\ell)}$ is the set of latents at the ℓ-th resolution level. We employ the use of **hierarchical VAEs**.

We write our multiresolution variational lower bound $\mathcal{L}_{\text{MGVAE}}(\phi, \theta)$ on $\log p(\mathcal{G})$ compactly as

$$\mathcal{L}_{\text{MGVAE}}(\phi, \theta) = \sum_i \sum_\ell \Big[\mathbb{E}_{q_\phi(\mathcal{Z}_i^{(\ell)} | \mathcal{G}_i^{(\ell)})} [\log p_\theta(\mathcal{G}_i^{(\ell)} | \mathcal{Z}_i^{(\ell)})] - \mathcal{D}_{\text{KL}}(q_\phi(\mathcal{Z}_i^{(\ell)} | \mathcal{G}_i^{(\ell)}) || p_0(\mathcal{Z}_i^{(\ell)})) \Big]$$

Multiresolution Equivariant Graph VAE (2)

In general, the overall optimization is given as follows:

$$\min_{\phi,\theta,\{\hat{\boldsymbol{\mu}}^{(\ell)},\hat{\boldsymbol{\Sigma}}^{(\ell)}\}_\ell} \mathcal{L}_{\text{MGVAE}}(\phi,\theta;\{\hat{\boldsymbol{\mu}}^{(\ell)},\hat{\boldsymbol{\Sigma}}^{(\ell)}\}_\ell) + \sum_{i,\ell} \lambda^{(\ell)} \mathcal{D}_{\text{KL}}(P_i^{(\ell)} || Q_i^{(\ell)}),$$

where

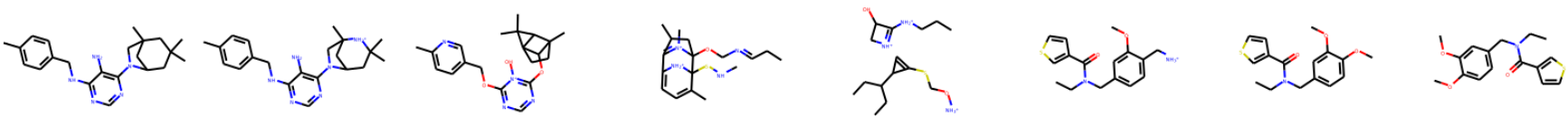
- ϕ denotes all learnable parameters of the encoders,
- θ denotes all learnable parameters of the decoders,
- $\mathcal{D}_{\text{KL}}(P_i^{(\ell)} || Q_i^{(\ell)})$ is the balanced-cut loss for graph $\mathcal{G}_i$ at level ℓ,
- $\hat{\boldsymbol{\mu}}^{(\ell)}$and $\hat{\boldsymbol{\Sigma}}^{(\ell)}$ are learnable parameters of the prior in an equivariant manner.

Multiresolution Equivariant Graph VAE (3)

Molecular graph generation results on QM9 & ZINC datasets:

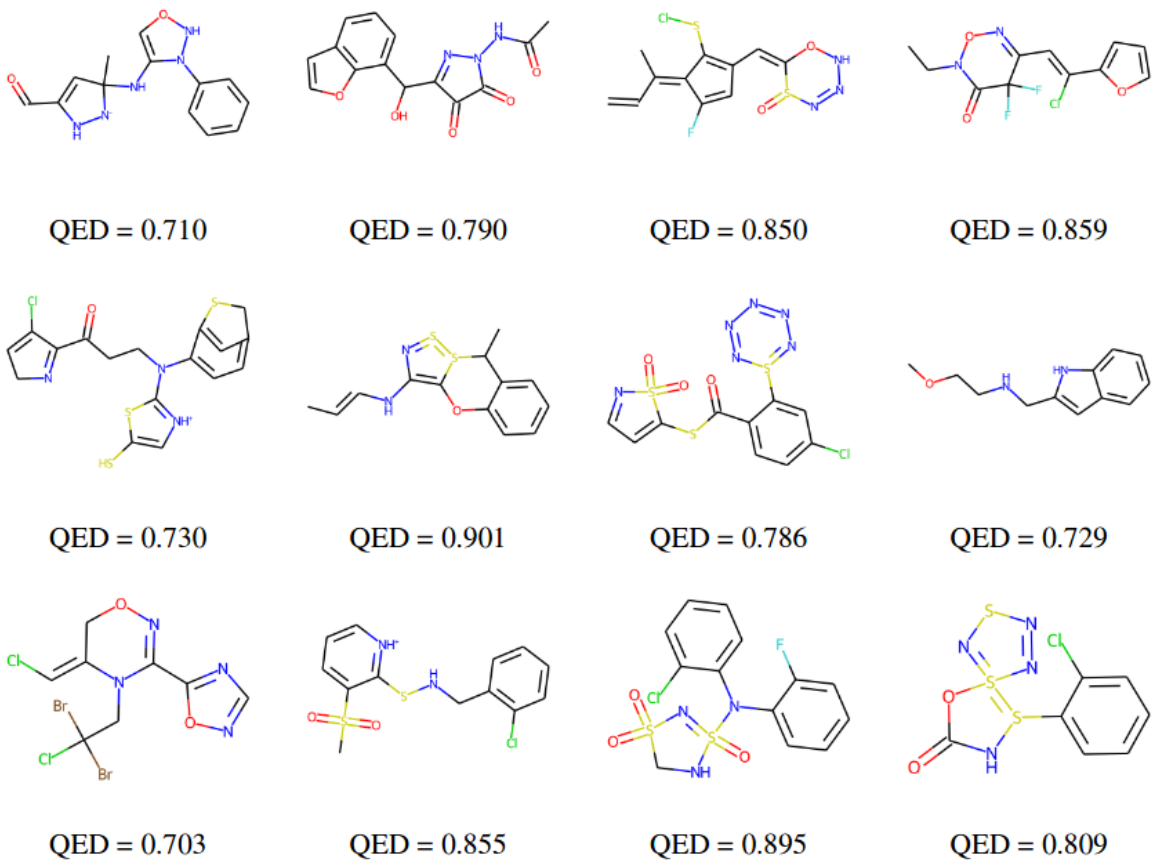
Dataset	Method	Training size	Input features	Validity	Novelty	Uniqueness
QM9	GraphVAE	~ 100K	Graph	61.00%	85.00%	40.90%
	CGVAE			100%	94.35%	98.57%
	MolGAN			98.1%	94.2%	10.4%
	Autoregressive MGN	10K		100%	95.01%	97.44%
	All-at-once MGVAE			100%	100%	95.16%
ZINC	GraphVAE	~ 200K	Graph	14.00%	100%	31.60%
	CGVAE			100%	100%	99.82%
	JT-VAE			100%	-	-
	Autoregressive MGN	1K		100%	99.89%	99.69%
	All-at-once MGVAE	10K	Chemical	99.92%	100%	99.34%

Interpolation on the latent:



(Hy & Kondor, 2021)

Multiresolution Equivariant Graph VAE (4)

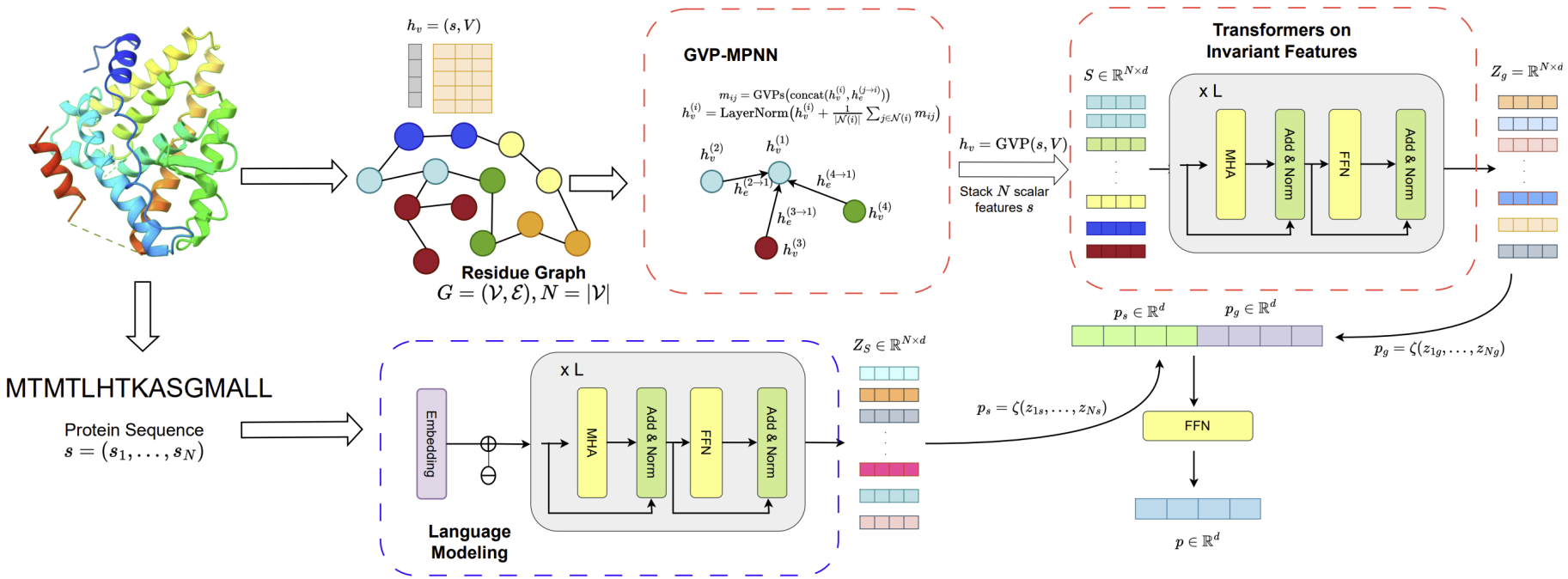


Some generated molecules with high QED (drug-likeness score).

Software

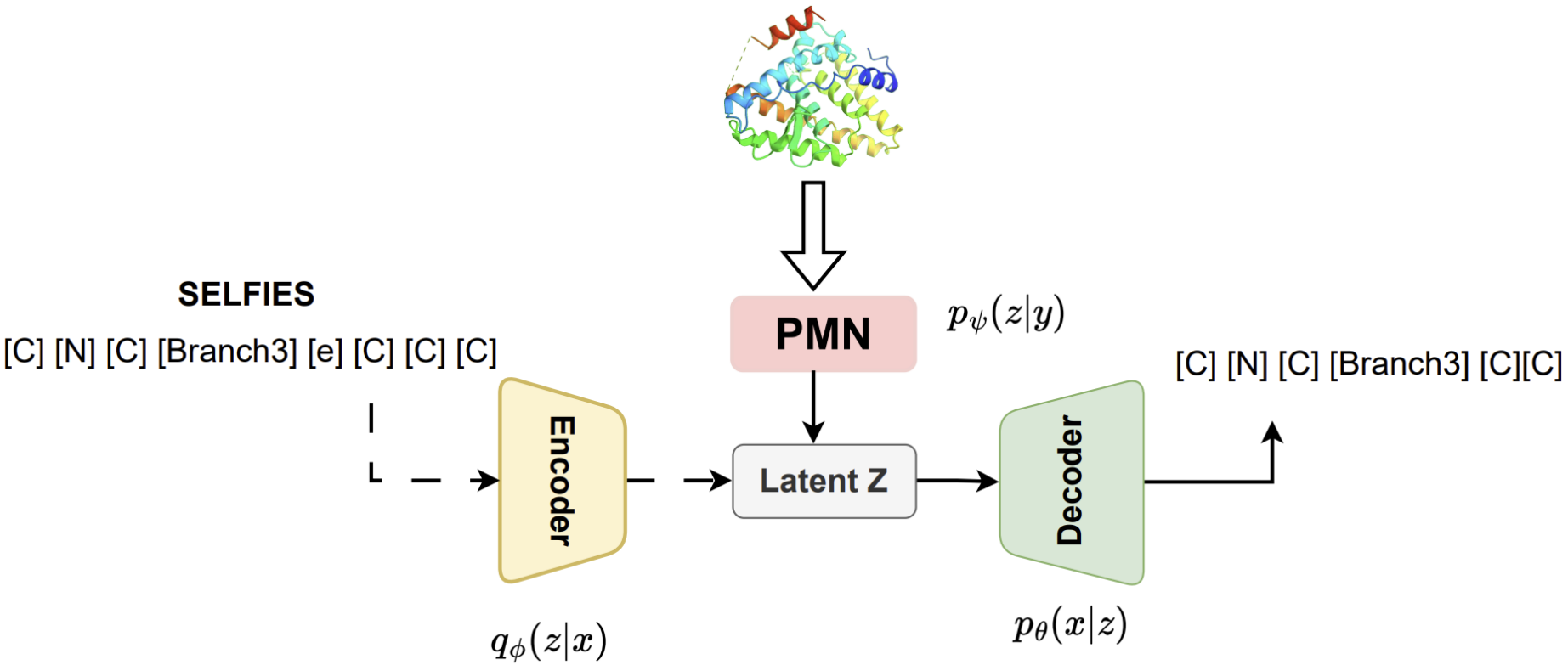
`https://github.com/HyTruongSon/MGVAE`

How can we generate protein-binding ligands?



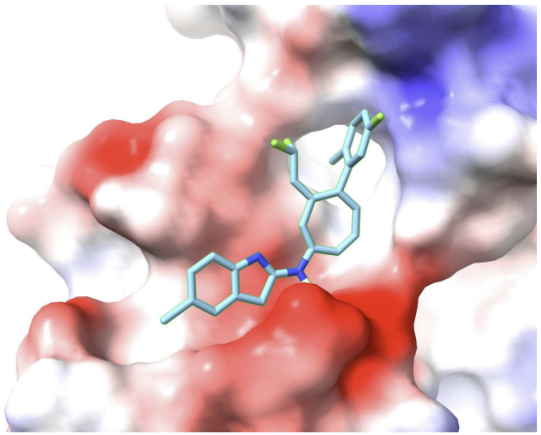
Overview of our **Protein Multimodal Network** (PMN) that learns to produce a unified representation of the protein structures including primary structure (i.e. amino-acid sequence) and tertiary structure (i.e. 3D structure) by Large Language Models and Graph Neural Networks.

Protein-binding ligand generation (1)

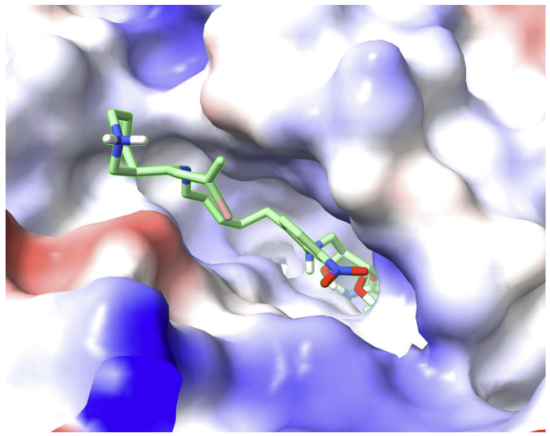


TargetVAE with an encoder, decoder, and a prior network. The PMN prior network computes the conditions from protein structures for constructing the latent space of the VAE framework, which learns to generate SELFIES representations of molecules.

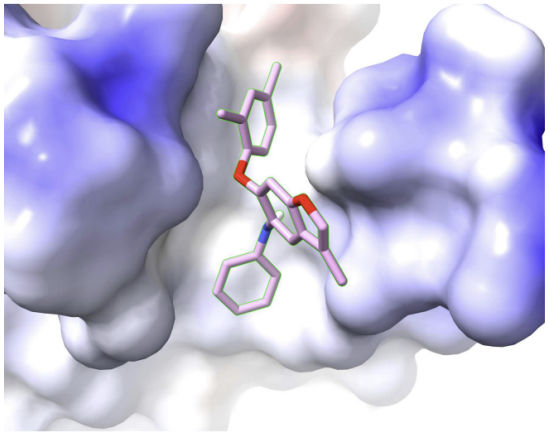
Protein-binding ligand generation (2)



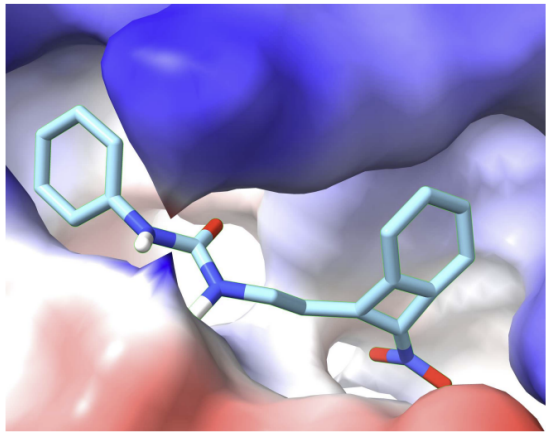
(a) 1iep, 3.59, -9.48 kcal/mol



(b) 2rgp, 4.28, -8.17 kcal/mol



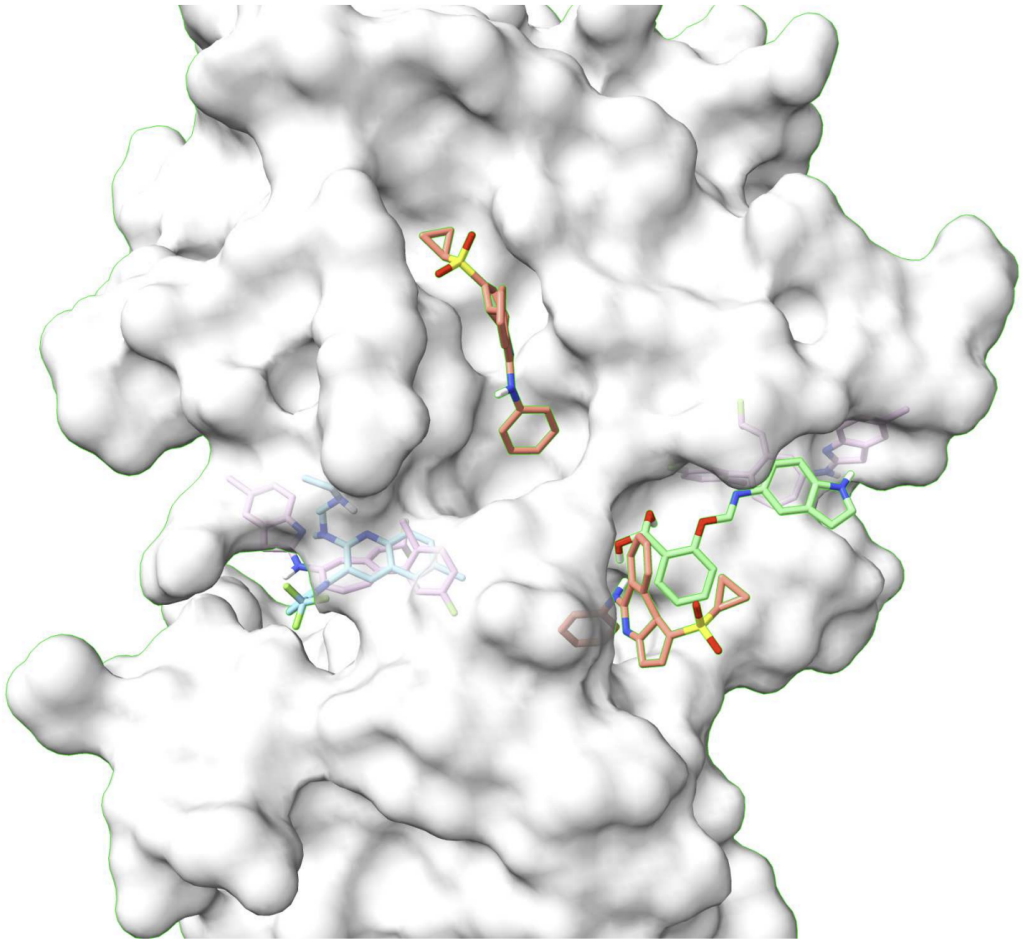
(c) 3eml,2.20,-8.29 kcal/mol



(d) 3ny8, 2.69,-8.78 kcal/mol

Each figure is associated with the name of each target protein, synthetic accessibility, and binding affinity in kcal/mol of the generated ligand.

Protein-binding ligand generation (3)



Multiple generated ligands with different poses bind to a given target protein. **Our TargetVAE can generate ligands for a protein without the prior knowledge of the binding pocket.**

Thank you for your attention!

For prospective students/interns/residents:

- Email me at `sonpascal93@gmail.com`
- Include your CV, academic transcript, github, etc.
- Your ideas, proposals, suggestions, etc.

For potential collaborators: I am always happy to hear from you and your ideas by any mean of contact.