Wilkinson Fellowship's Seminar - January 5, 2022

Graph Representation Learning, Deep Generative Models On Graphs & Multiresolution Machine Learning

Truong Son Hy *

*Department of Computer Science
The University of Chicago

Ryerson Physical Laboratory

Content

1. Graph representation learning
   - Message passing neural networks
   - Permutation equivariance
   - Covariant compositional networks
2. Deep generative models on graphs
   - Variational Autoencoder (VAE)
   - Equivariant graph/molecule generation
   - Multiresolution graph VAE
3. Multiresolution matrix factorization
   - Reinforcement Learning & Stiefel manifold optimization
   - Graph wavelets & Wavelet neural networks
   - Tensor factorization

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.

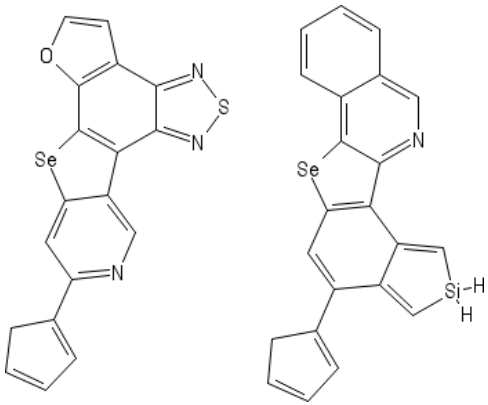
Keywords

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

Graph neural networks

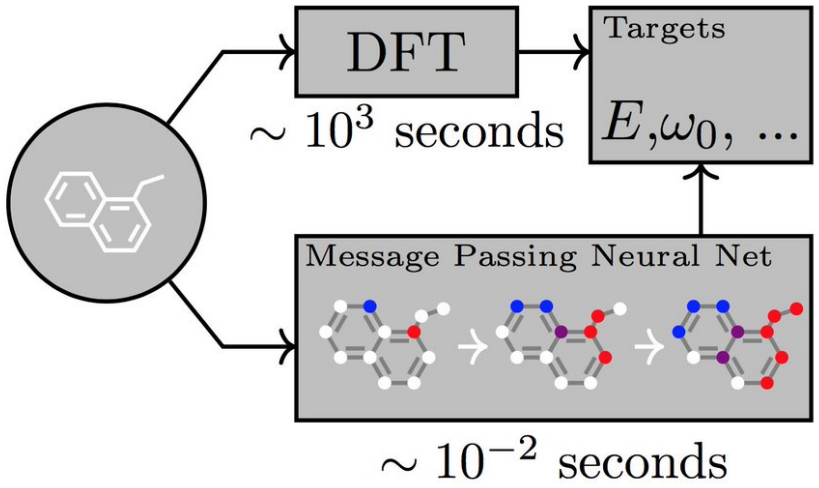


Figure 1. A Message Passing Neural Network predicts quantum properties of an organic molecule by modeling a computationally expensive DFT calculation.

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* $\psi(.)$.

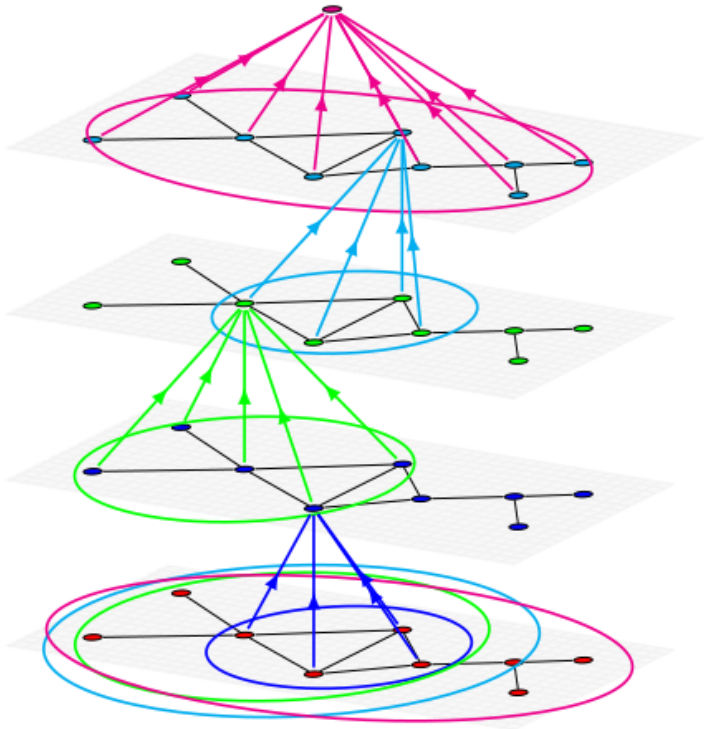
Message Passing Scheme (1)

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.

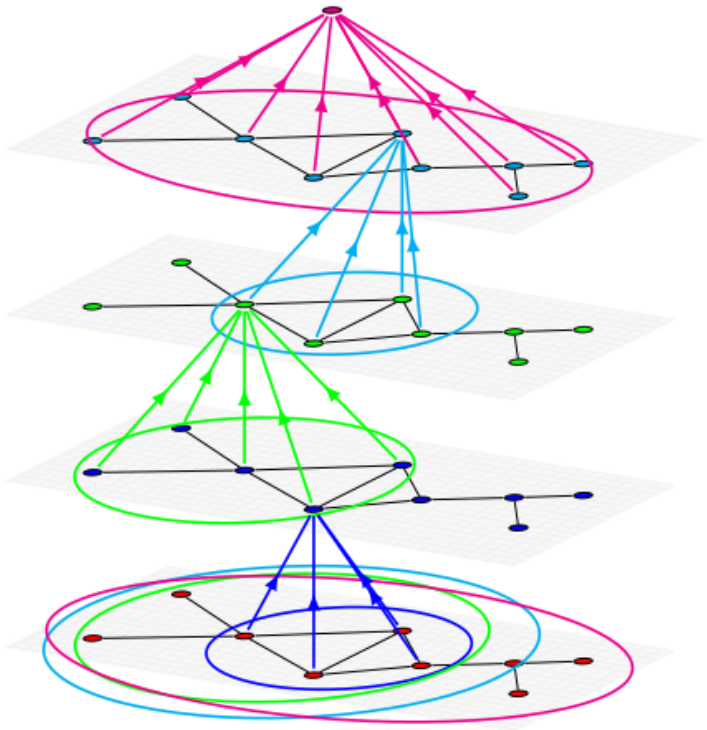
Message Passing Scheme (2)

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



Message Passing Scheme (2)

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



Note

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

Message Passing Scheme (3)

With learnable parameters:

for $v \in V$ **do**
  $f_v^0 \leftarrow \ell_v$
end for
for $t = 1 \to T$ **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: $\phi(G) \leftarrow \psi\left(\{f_v^T\}_{v \in V}; W^{T+1}\right)$

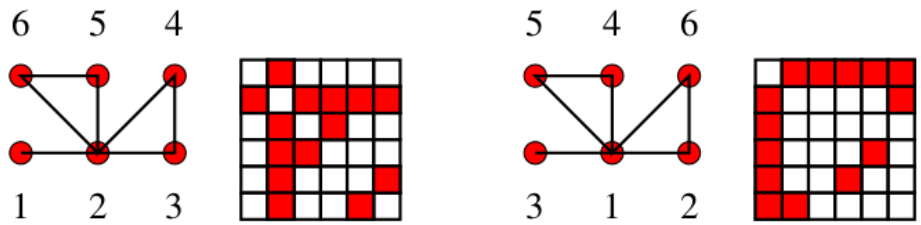
Given a graph properties $y(G) \in \mathbb{R}^d$ to regress, we have the optimization:

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

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

Invariance

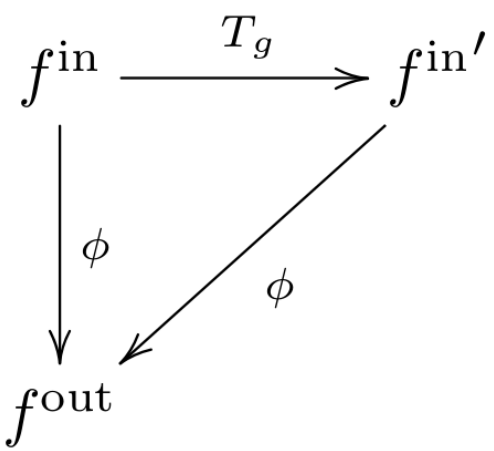
We renumber the vertices by a permutation $\sigma : \{1, 2, .., 6\} \mapsto \{1, 2, .., 6\}$. The adjacency matrices of G (left) and G' (right) are different, but topologically they represent the same graph:

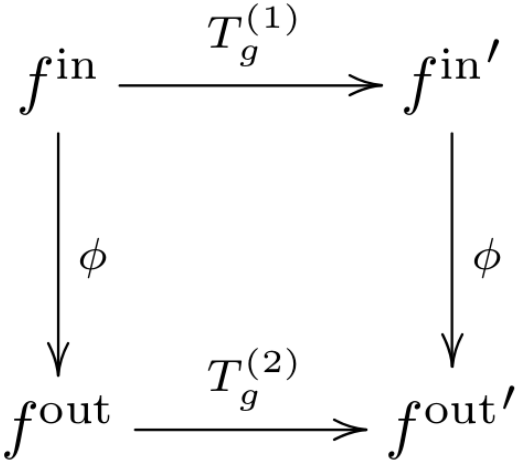


Therefore, ϕ must be **invariant** wrt permutation, i.e. $\phi(G) = \phi(G')$.

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 $\mathbb{S}_n$.

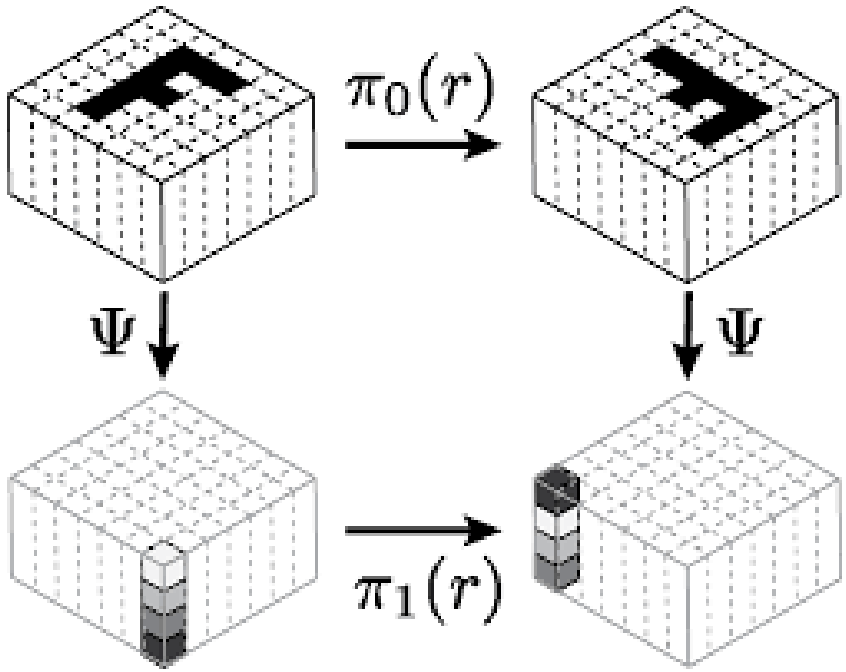




Invariance: $\phi(T_g(f)) = \phi(f)$

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

Equivariant Neural Networks



The first concept of Equivariant Neural Networks was introduced by Taco Cohen and Max Welling:

Group Equivariant Convolutional Networks, `https://arxiv.org/abs/1602.07576`

Steerable CNNs, `https://arxiv.org/abs/1612.08498`

Message Passing Neural Networks and its limitation (1)

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:

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

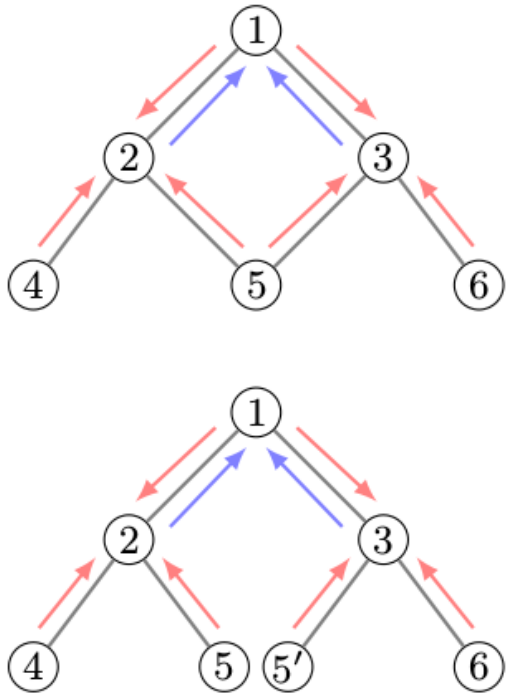
where:

- $A \in \{0,1\}^{n \times n}$ 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.

Message Passing Neural Networks and its limitation (2)

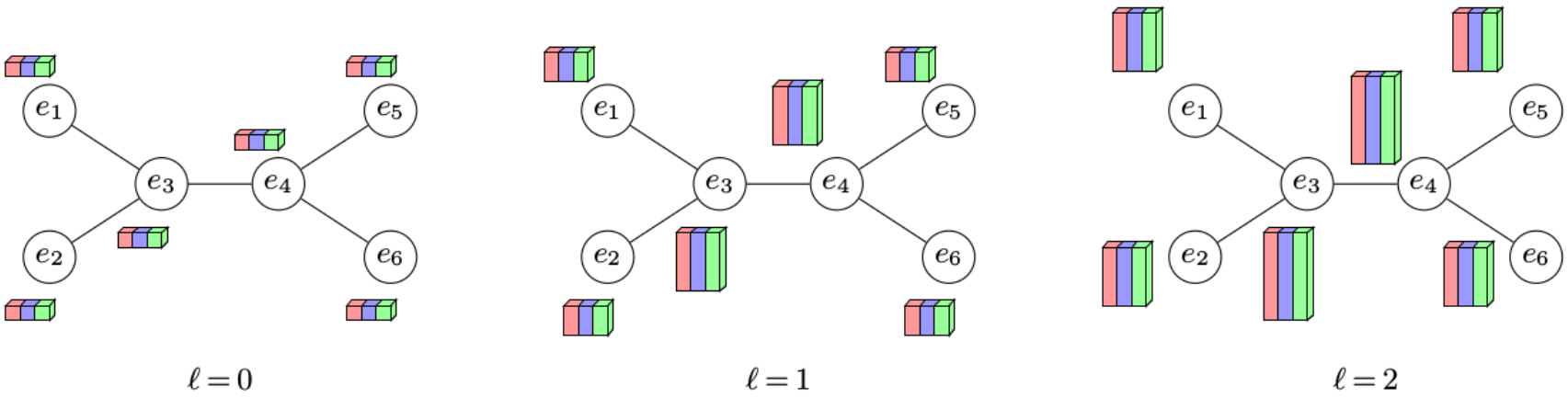
The summing operator **limits** the representative power of MPNNs such that each node loses their identity after being aggregated. For example:

These two graphs are **not** isomorphic, but after a single round of message passing (red arrows), the messages at vertices 2 and 3 will be identical in both graphs. In the second round, vertex 1 will get the same messages in both graphs (blue arrows), and will have **no** way to distinguish whether 5 and 5' are the same vertex or not.



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)

We propose **first order message passing** by representing each vertex v by a matrix: $f_v \in \mathbb{R}^{|\mathcal{N}_v| \times d}$, each row of this feature matrix corresponds to a vertex in the neighborhood of v.

Definition

We say that v is a **first order covariant node** if under the permutation of its receptive field $\mathcal{N}_v$ by any $\pi \in \mathbb{S}_{|\mathcal{N}_v|}$, its activation transforms as

$$f_v \mapsto P_\pi f_v,$$

where P_π is the permutation matrix:

$$[P_\pi]_{i,j} \triangleq \begin{cases} 1, & \pi(j) = i \\ 0, & otherwise \end{cases}$$

The transformed activation will be $[f'_v]_{a,s} = [f_v]_{\pi^{-1}(a),s}$ where s is the channel index.

Covariant Compositional Networks (3)

In **second order message passing**, we can represent node activations by a 3rd order tensor $F_v \in \mathbb{R}^{|\mathcal{N}_v| \times |\mathcal{N}_v| \times d}$ and require these feature tensors to transform covariantly in a similar manner.

Definition

We say that v is a **second order covariant node** if under the permutation of its receptive field $\mathcal{N}_v$ by an $\pi \in \mathbb{S}_{|\mathcal{P}_v|}$, its activation transforms as

$$F_v \mapsto P_\pi F_v P_\pi^T.$$

The transformed activation F'_v will be:

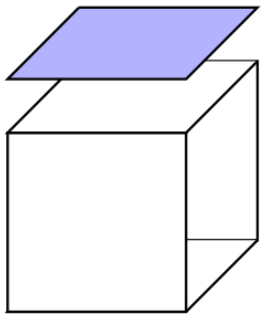
$$[F'_v]_{a,b,s} = [F_v]_{\pi^{-1}(a),\pi^{-1}(b),s}$$

where s is the channel index.

Covariant Compositional Networks (4)

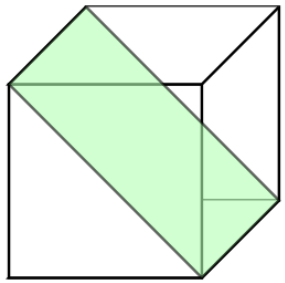
Permutation covariant operators:

1. Projections



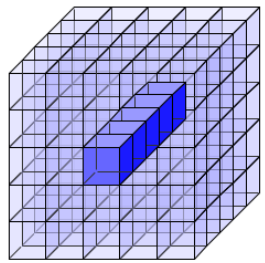
$$C_{i,j} = \sum_a A_{a,i,j}$$

2. Diagonals



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

3. Contractions

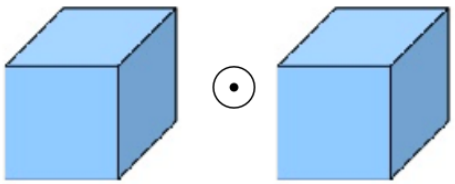


$$C_k = \sum_{i,j} A_{i,j,k}$$

Covariant Compositional Networks (5)

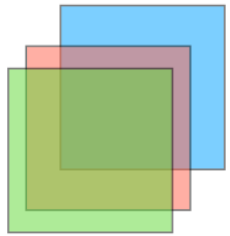
Permutation covariant operators (continued):

4. Hadamard products



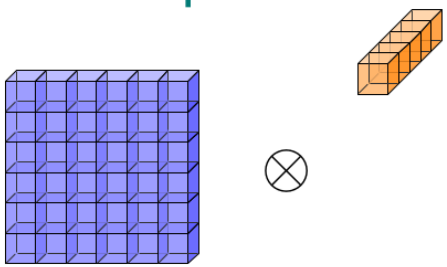
$$C_{i,j,k} = A_{i,j,k} B_{i,j,k}$$

5. Stacking



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

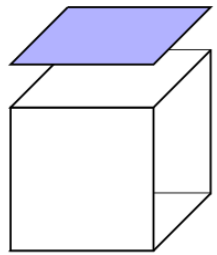
6. Tensor products



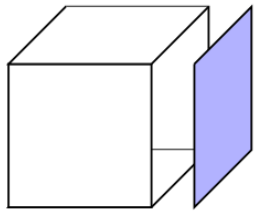
$$C_{i,j,k} = A_{i,j} B_k$$

Covariant Compositional Networks (6)

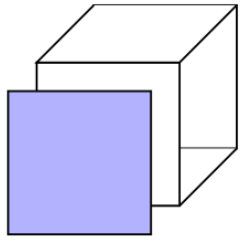
There are six different ways of covariantly reducing (contracting) a third order tensor to a second order tensor:



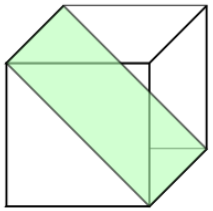
$$C_{i,j} = \sum_a A_{a,i,j}$$



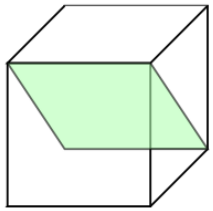
$$C_{i,j} = \sum_j A_{i,a,j}$$



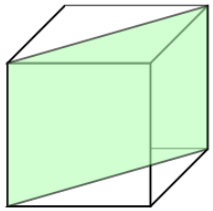
$$C_{i,j} = \sum_k A_{i,j,a}$$



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

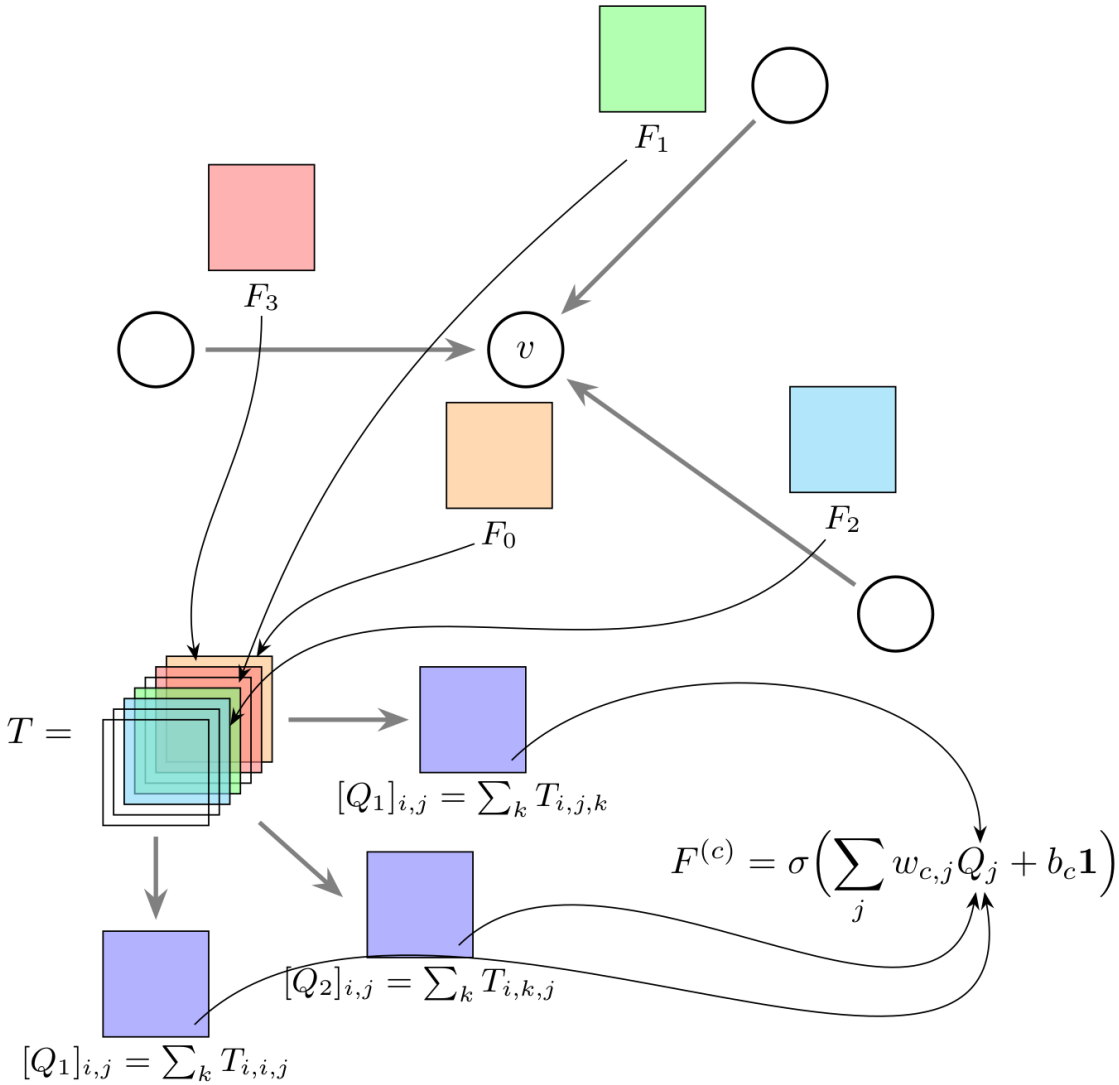


$$C_{i,j} = \sum_i A_{i,j,i}$$



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

Covariant Compositional Networks (7)



Second-order CCN

Covariant Compositional Networks (8)

	MUTAG	PTC	NCI1	NCI109
Wesifeiler–Lehman kernel[33]	84.50 ± 2.16	59.97 ± 1.60	84.76 ± 0.32	85.12 ± 0.29
Wesifeiler–Lehman edge kernel[33]	82.94 ± 2.33	60.18 ± 2.19	84.65 ± 0.25	85.32 ± 0.34
Shortest path kernel[29]	85.50 ± 2.50	59.53 ± 1.71	73.61 ± 0.36	73.23 ± 0.26
Graphlets kernel[31]	82.44 ± 1.29	55.88 ± 0.31	62.40 ± 0.27	62.35 ± 0.28
Random walk kernel[28]	80.33 ± 1.35	59.85 ± 0.95	TIMED OUT	TIMED OUT
Multiscale Laplacian graph kernel[62]	87.94 ± 1.61	63.26 ± 1.48	81.75 ± 0.24	81.31 ± 0.22
PSCN($k = 10$)[37]	88.95 ± 4.37	62.29 ± 5.68	76.34 ± 1.68	N/A
Neural graph fingerprints[21]	89.00 ± 7.00	57.85 ± 3.36	62.21 ± 4.72	56.11 ± 4.31
Second order CCN (our method)	91.64 ± 7.24	70.62 ± 7.04	76.27 ± 4.13	75.54 ± 3.36

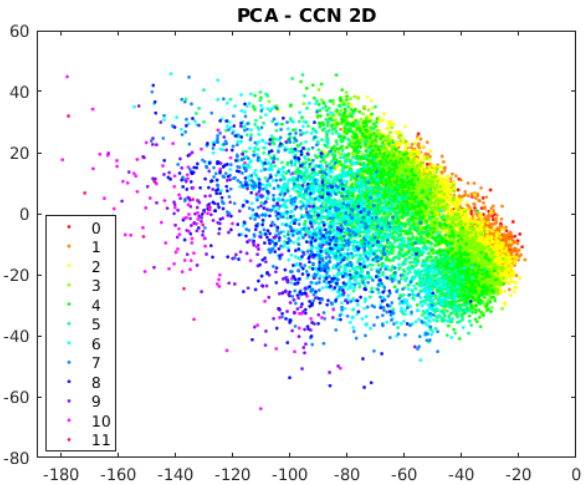
TABLE I. Classification results on the kernel datasets (accuracy ± standard deviation).

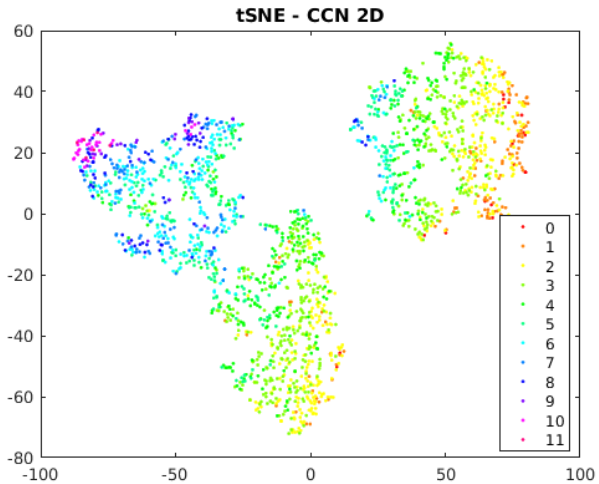
(Kondor et al., 2018), (Hy et al., 2018)

Covariant Compositional Networks (9)

	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**

TABLE II. HCEP regression results. Error of predicting power conversion efficiency in units of percent.





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

Covariant Compositional Networks (10)

	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

TABLE IV. The mean absolute error of CCN compared to DFT error when using the complete set of physical features used in Ref. 25 in addition to the graph of each molecule.

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

Software & Future works

Software:

1. `https://github.com/HyTruongSon/GraphFlow`
2. `https://github.com/HyTruongSon/LibCCNs`

Future works:

- Hypergraph neural networks
  - Simplicial complexes of graphs (algebraic geometry)
  - Hypergraph Fourier Transform
- k-fold atomic interaction in N-Body networks

Software & Future works

Software:

1. `https://github.com/HyTruongSon/GraphFlow`
2. `https://github.com/HyTruongSon/LibCCNs`

Future works:

- Hypergraph neural networks
  - Simplicial complexes of graphs (algebraic geometry)
  - Hypergraph Fourier Transform
- k-fold atomic interaction in N-Body networks

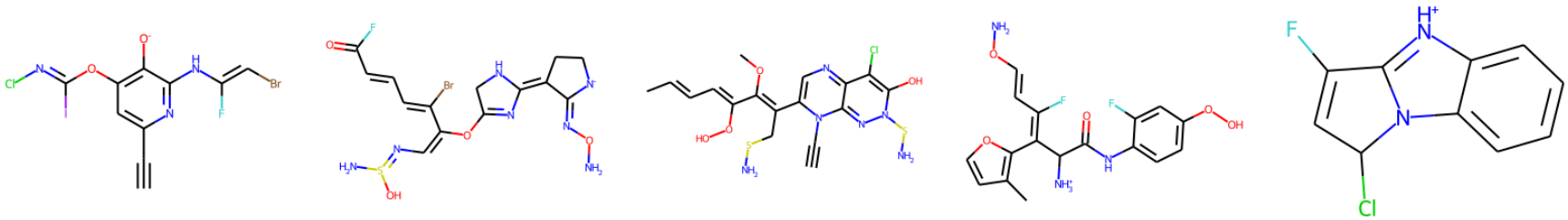
Question

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

Graph generation (1)

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 generation (2)

Methods:

1. Traditional graph generation approaches:
   - Erdös-Rényi (ER) Model
   - Stochastic Block Models (SBM)

   Limitation: Rely on a fixed, hand-crafted generation process. Lack the ability to learn a generative model from data.

Graph generation (2)

Methods:

1. Traditional graph generation approaches:
   - Erdös-Rényi (ER) Model
   - Stochastic Block Models (SBM)

   Limitation: Rely on a fixed, hand-crafted generation process. Lack the ability to learn a generative model from data.
2. Deep generative models:
   - **All-at-once**:
     - Generate the whole adjacency matrix with all node (atomic) features
     - VAEs, GANs

Graph generation (2)

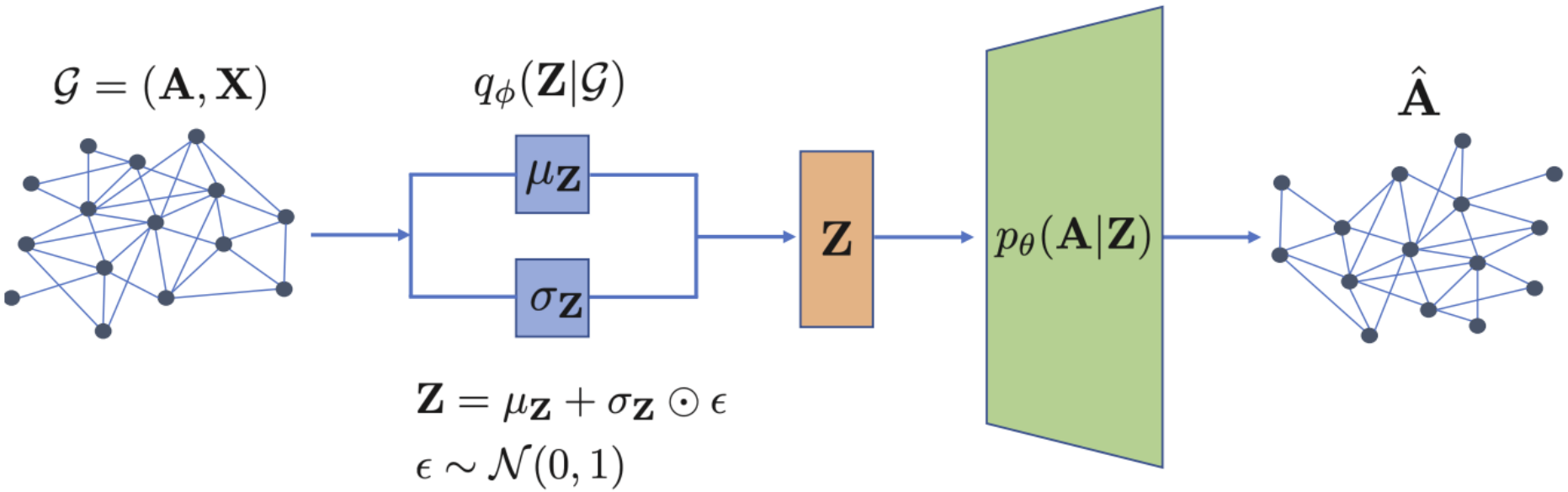
Methods:

1. Traditional graph generation approaches:
   - Erdös-Rényi (ER) Model
   - Stochastic Block Models (SBM)

   Limitation: Rely on a fixed, hand-crafted generation process. Lack the ability to learn a generative model from data.
2. Deep generative models:
   - **All-at-once**:
     - Generate the whole adjacency matrix with all node (atomic) features
     - VAEs, GANs
   - **Autoregressive**:
     - Generate a graph *incrementally* by adding one node (atom) or one edge (bond) at a time
     - Reinforcement Learning, LSTM-based language models, GRNN, GRANs, etc.

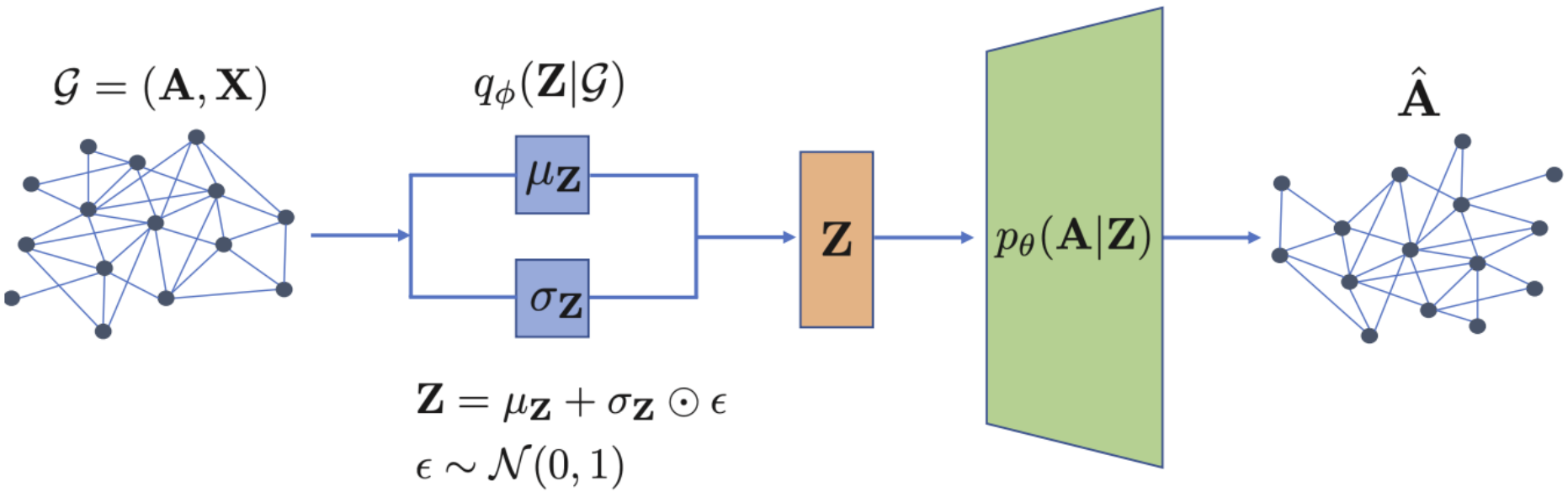
Graph Variational Autoencoder (1)

- **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 (that we sample latent embeddings $\boldsymbol{Z}$ from).



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 $\boldsymbol{Z} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{1})$.



Graph Variational Autoencoder (3)

Given a set of training graphs $\{\mathcal{G}_1, .., \mathcal{G}_n\}$, we can train a VAE model by minimizing 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:

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

Why 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 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 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 (1)

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.

Markov Random Fields (2)

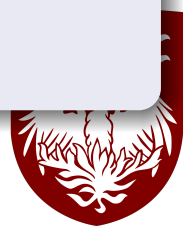
Hammersley–Clifford theorem

A positive distribution $p(\boldsymbol{z}) > 0$ satisfies the conditional independent properties of an undirected graph $\mathcal{G}$ iff p can be represented as a product of potential functions ψ, one per *maximal clique*, i.e.,

$$p(\boldsymbol{z}|\boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta})} \prod_{c \in \mathcal{C}} \psi_c(z_c|\theta_c)$$

where $\mathcal{C}$ is the set of all the (maximal) cliques of $\mathcal{G}$, and $Z(\boldsymbol{\theta})$ is the *partition function* to ensure the overall distribution sums to 1, and given by

$$Z(\boldsymbol{\theta}) = \sum_{\boldsymbol{z}} \prod_{c \in \mathcal{C}} \psi_c(z_c|\theta_c)$$



Markov Random Fields (3)

Our second order encoder inherits Gaussian MRF as *pairwise* MRF of the following form

$$p(\boldsymbol{z}|\boldsymbol{\theta}) \propto \prod_{s\sim t} \psi_{st}(z_s, z_t) \prod_t \psi_t(z_t)$$

where

$$\psi_{st}(z_s, z_t) = \exp\left(-\frac{1}{2} z_s \Lambda_{st} z_t \right),$$

$$\psi_t(z_t) = \exp\left(-\frac{1}{2} \Lambda_{tt} z_t^2 + \eta_t z_t \right)$$

are the edge and vertex potentals. The joint distribution can be written in the *information form* of a multivariate Gaussian in which

$$\boldsymbol{\Lambda} = \boldsymbol{\Sigma}^{-1}, \qquad \boldsymbol{\eta} = \boldsymbol{\Lambda}\boldsymbol{\mu}$$

$$p(\boldsymbol{z}|\boldsymbol{\theta}) \propto \exp\left(\boldsymbol{\eta}^T \boldsymbol{z} - \frac{1}{2} \boldsymbol{z}^T \boldsymbol{\Lambda} \boldsymbol{z} \right)$$

Markov Random Fields (4)

Our second order encoder produces tensor $\boldsymbol{L}$ as the second order activation, and set $\boldsymbol{\Sigma} = \boldsymbol{L}\boldsymbol{L}^T$ to ensure invertibility. The reparameterization trick is changed to

$$\boldsymbol{z} = \boldsymbol{\mu} + \boldsymbol{L}\boldsymbol{\epsilon}, \qquad \boldsymbol{\epsilon} \sim \mathcal{N}(0, 1)$$

Markov Random Fields (4)

Our second order encoder produces tensor $\boldsymbol{L}$ as the second order activation, and set $\boldsymbol{\Sigma} = \boldsymbol{L}\boldsymbol{L}^T$ to ensure invertibility. The reparameterization trick is changed to

$$\boldsymbol{z} = \boldsymbol{\mu} + \boldsymbol{L}\boldsymbol{\epsilon}, \qquad \boldsymbol{\epsilon} \sim \mathcal{N}(0, 1)$$

Hungarian matching

We allow the prior $\mathcal{N}(\hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\Sigma}})$ to be learnable in which $\hat{\boldsymbol{\mu}}$ and $\hat{\boldsymbol{\Sigma}}$ are parameters optimized by back propagation in a data driven manner. We want to solve the following convex optimization problem that is our new equivariant loss function

$$\min_{\sigma \in \mathbb{S}_n} \mathcal{D}_{\mathsf{KL}}(\mathcal{N}(\boldsymbol{P}_\sigma \boldsymbol{\mu}, \boldsymbol{P}_\sigma \boldsymbol{\Sigma} \boldsymbol{P}_\sigma^T) || \mathcal{N}(\hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\Sigma}}))$$

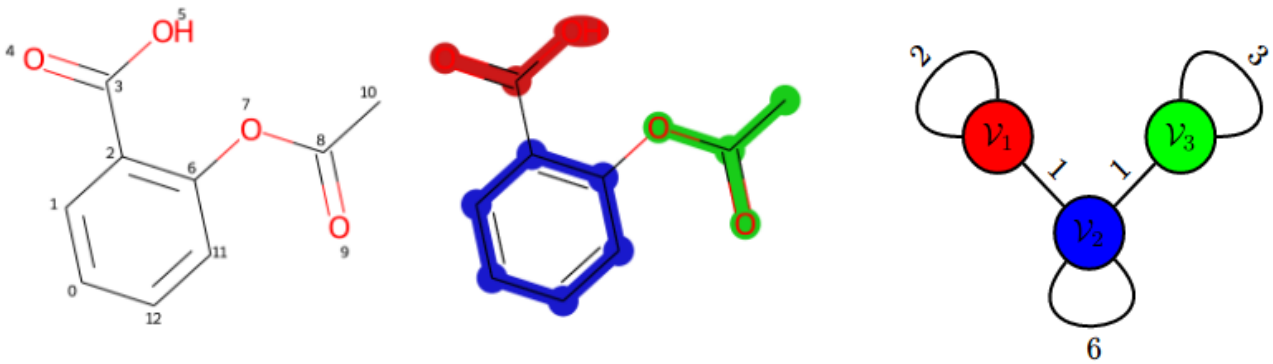
that can be approximated by matching on a bipartite graph.

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)

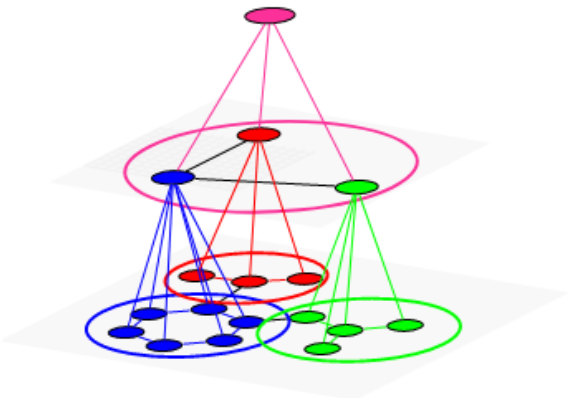
It is desirable to have a *balanced* K-cluster partition in which clusters $\mathcal{V}_1^{(\ell)}, .., \mathcal{V}_K^{(\ell)}$ (at the ℓ-th resolution level) have similar sizes that are close to $|\mathcal{V}^{(\ell)}|/K$.

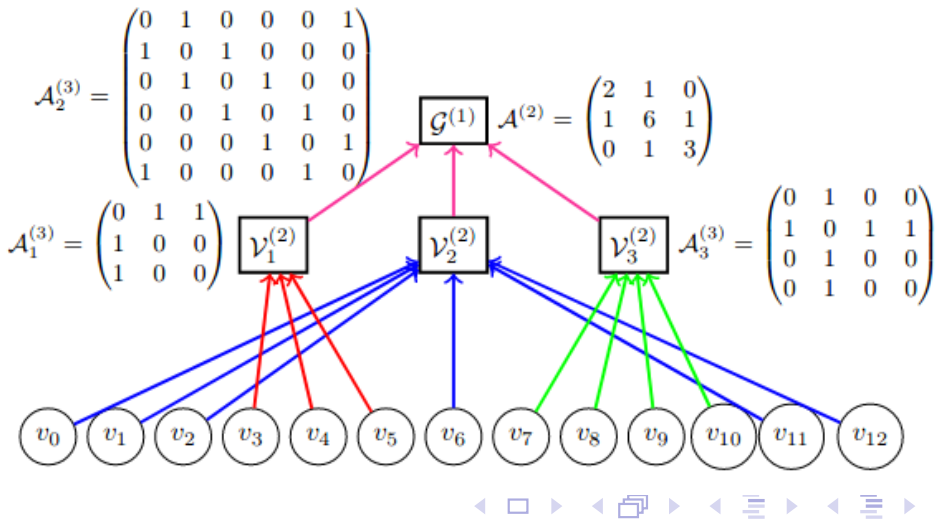
Multiresolution Graph Network (2)

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**.

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

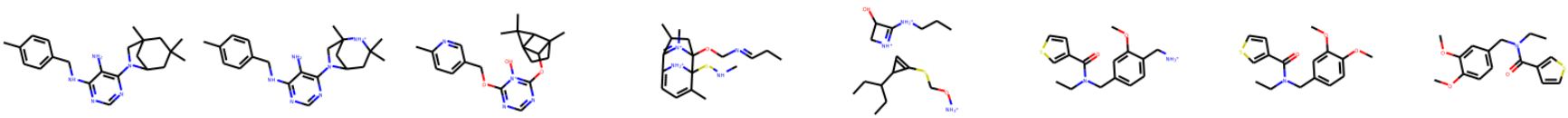
- ϕ denotes all learnable parameters of the encoders,
- θ denotes all learnable parameters of the decoders,
- $\mathcal{D}_{\text{KL}}(P_i^{(\ell)} || Q_i^{(\ell)})$ is the balanced-cut loss for graph $\mathcal{G}_i$ at level ℓ,
- $\hat{\boldsymbol{\mu}}^{(\ell)}$and $\hat{\boldsymbol{\Sigma}}^{(\ell)}$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

Multiresolution Equivariant Graph VAE (3)

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%

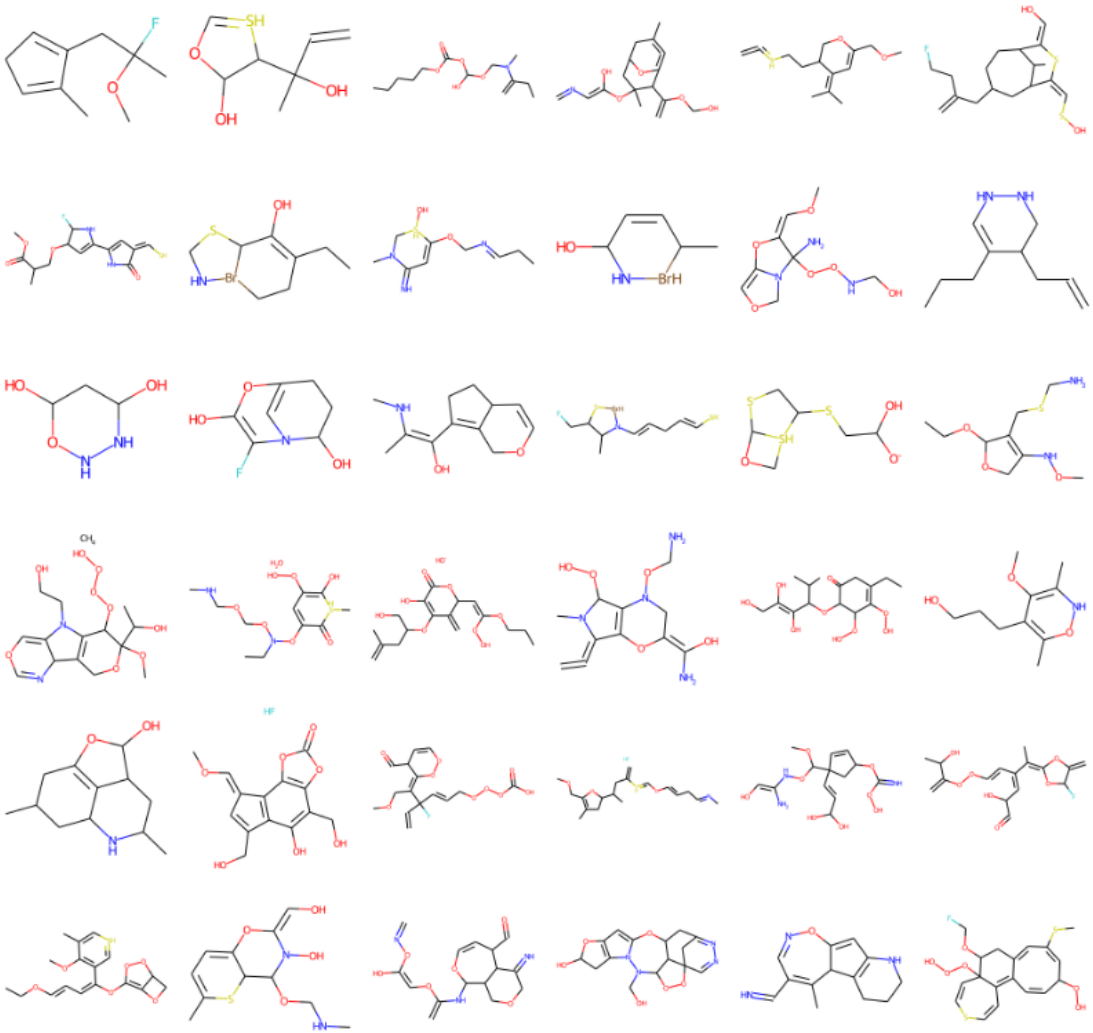
Table 1: Molecular graph generation results. GraphVAE results are taken from (Liu et al., 2018).

Interpolation on the latent:



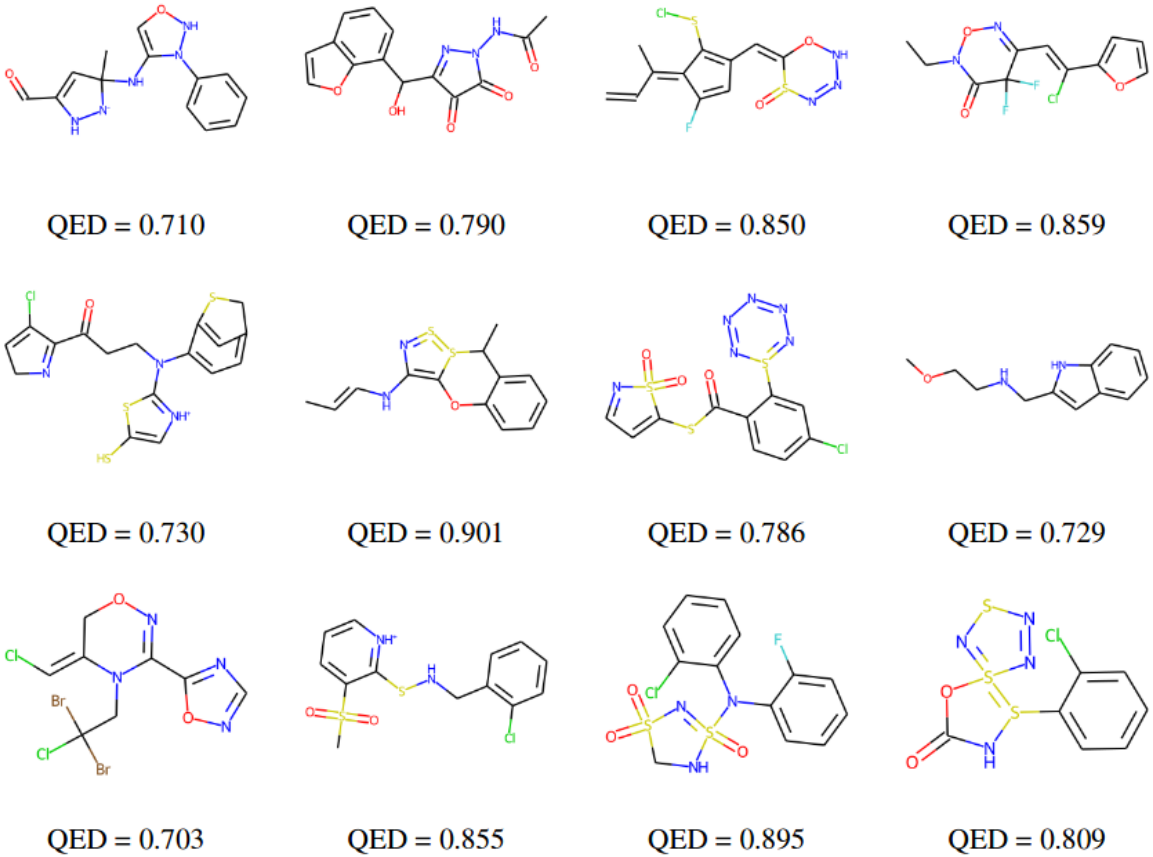
(Hy & Kondor, 2021)

Multiresolution Equivariant Graph VAE (4)



Some generated examples on ZINC by the all-at-once MGVAE with second order $\mathbb{S}_n$-equivariant decoders.

Multiresolution Equivariant Graph VAE (5)



Some generated molecules on ZINC by the autoregressive MGN with high QED (drug-likeness score).

Multiresolution Equivariant Graph VAE (6)

Table 3: Citation graph link prediction results (AUC & AP)

Dataset	**Cora**		**Citeseer**	
Method	AUC (ROC)	AP	AUC (ROC)	AP
SC	84.6 ± 0.01	88.5 ± 0.00	80.5 ± 0.01	85.0 ± 0.01
DW	83.1 ± 0.01	85.0 ± 0.00	80.5 ± 0.02	83.6 ± 0.01
VGAE	90.97 ± 0.77	91.88 ± 0.83	89.63 ± 1.04	91.10 ± 1.02
MGVAE (Spectral)	91.19 ± 0.76	92.27 ± 0.73	90.55 ± 1.17	91.89 ± 1.27
MGVAE (K-Means)	93.07 ± 5.61	92.49 ± 5.77	90.81 ± 1.19	91.98 ± 1.02
MGVAE	**95.67 ± 3.11**	**95.02 ± 3.36**	**93.93 ± 5.87**	**93.06 ± 6.33**

Method	MLP	GCN	GAT	MoNet	DiscenGCN	FactorGCN	GatedGCN$_E$	**MGN**
MAE	0.667	0.503	0.479	0.407	0.538	0.366	0.363	**0.290**

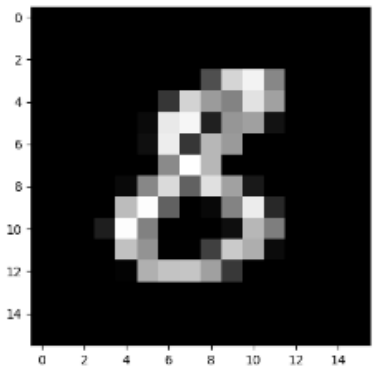
Table 11: Supervised MGN to predict solubility on ZINC dataset.

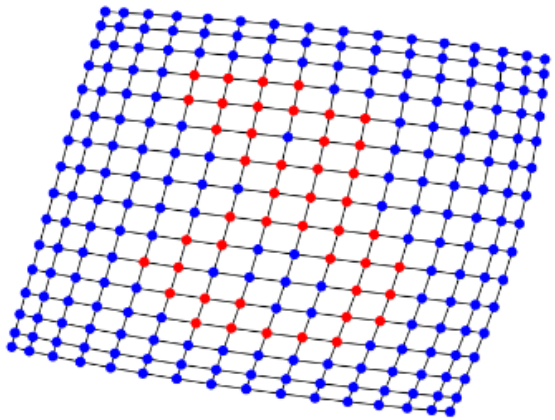
(Hy & Kondor, 2021)

Multiresolution Equivariant Graph VAE (7)

Method	FID$_\downarrow$ (32×32)	FID$_\downarrow$ (16×16)	FID$_\downarrow$ (8×8)
DCGAN	113.129	N/A	N/A
VEEGAN	68.749		
PACGAN	58.535		
PresGAN	42.019		
MGVAE	**39.474**	64.289	39.038

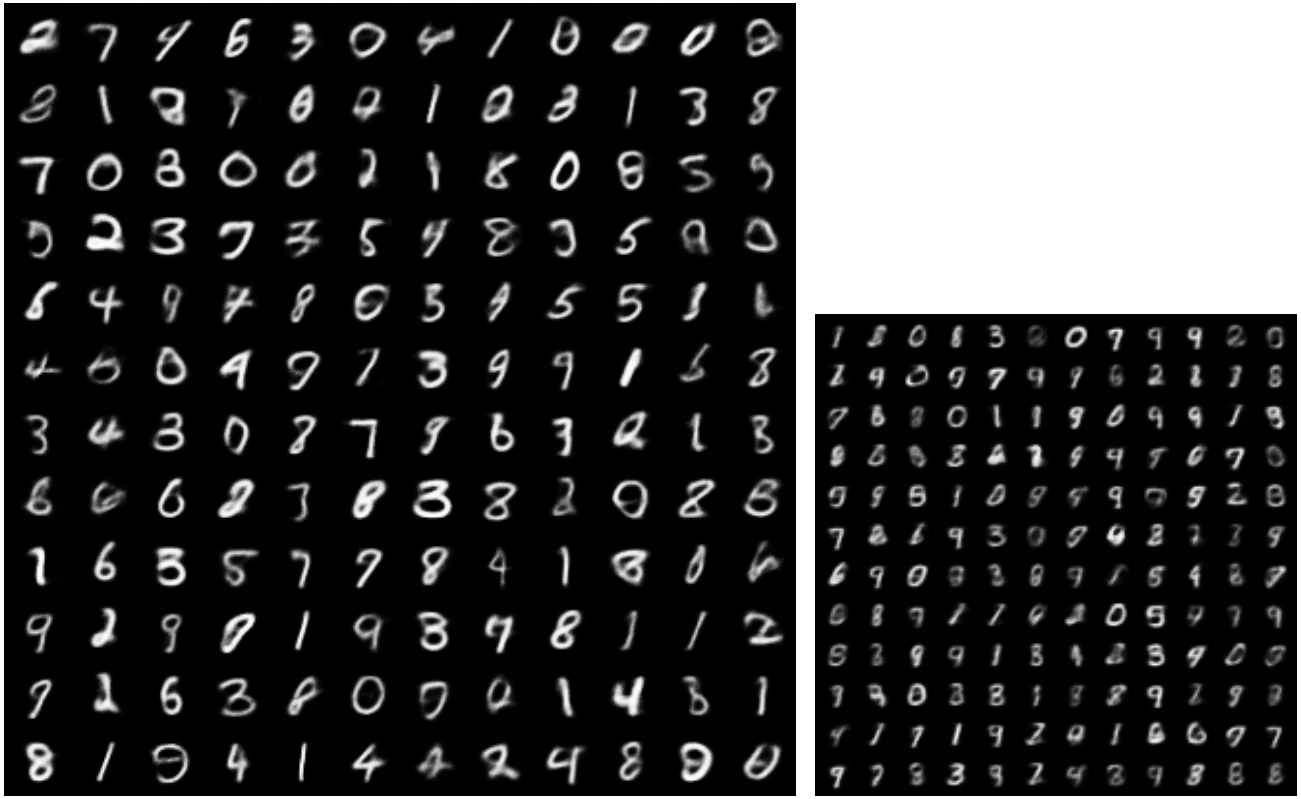
Graph-based image generation at multiple resolutions by MGVAE on MNIST digits





(Hy & Kondor, 2021)

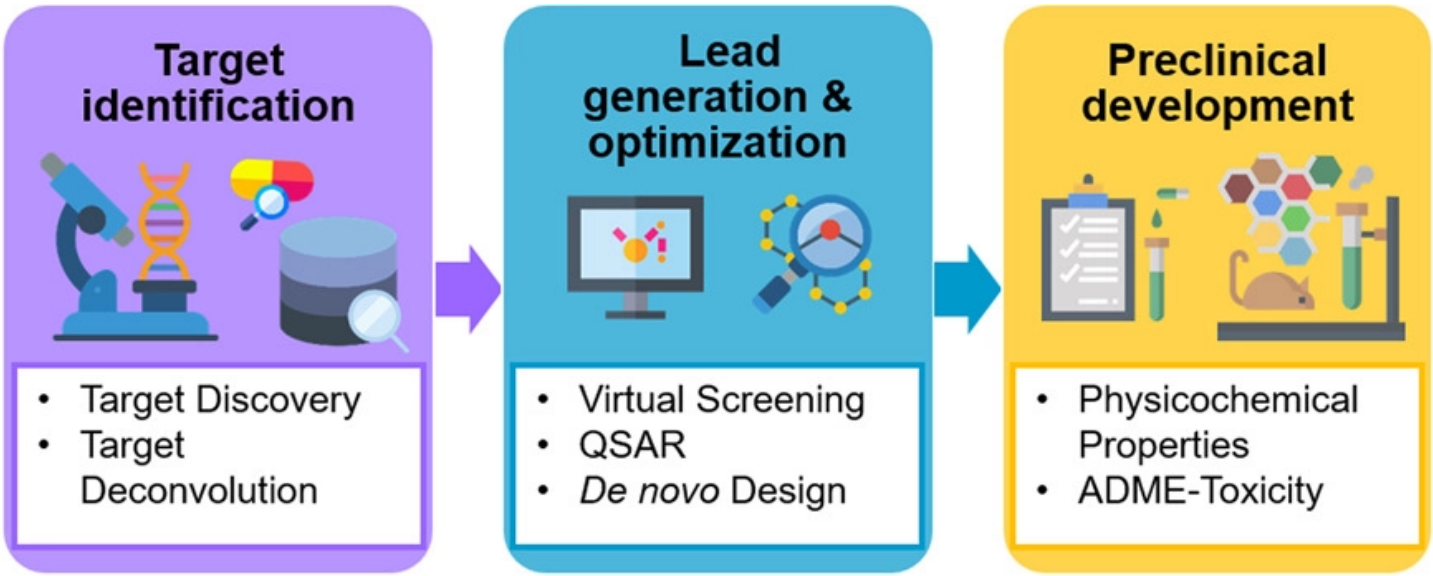
Multiresolution Equivariant Graph VAE (8)



Generated digit images at 32×32 (left) and 16×16 (right) resolutions

Future works (1)

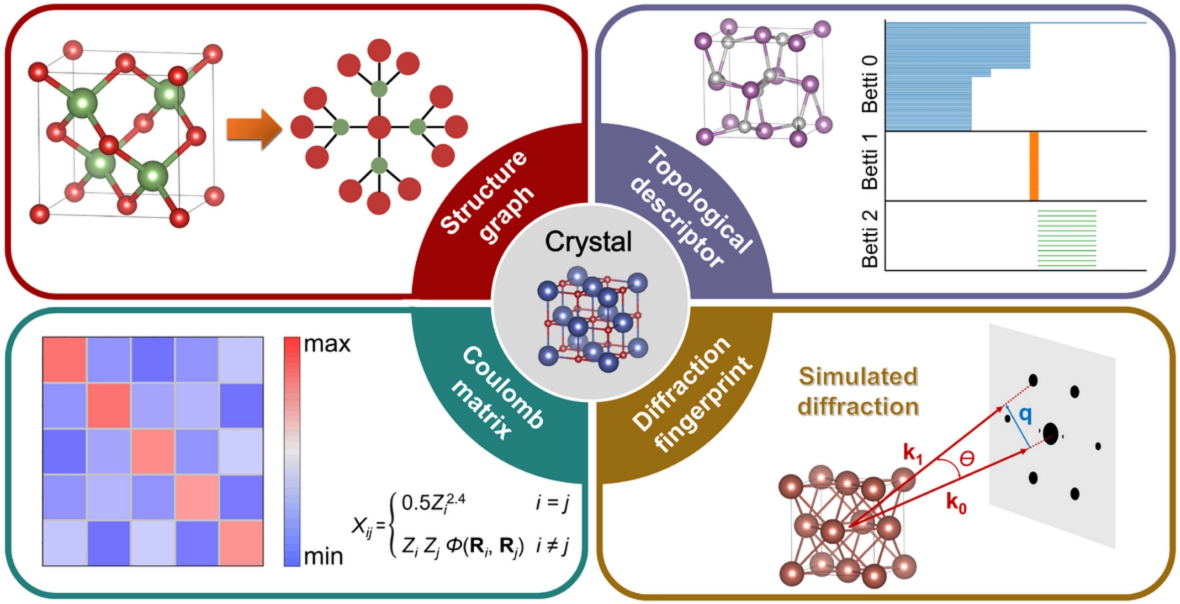
Drug discovery: Application of deep generative models on graphs into the lead optimization process that enhances the most promising compounds to improve effectiveness, safety and tolerability.



This figure is taken from (Sezen Vatansever et al., 2020).

Future works (2)

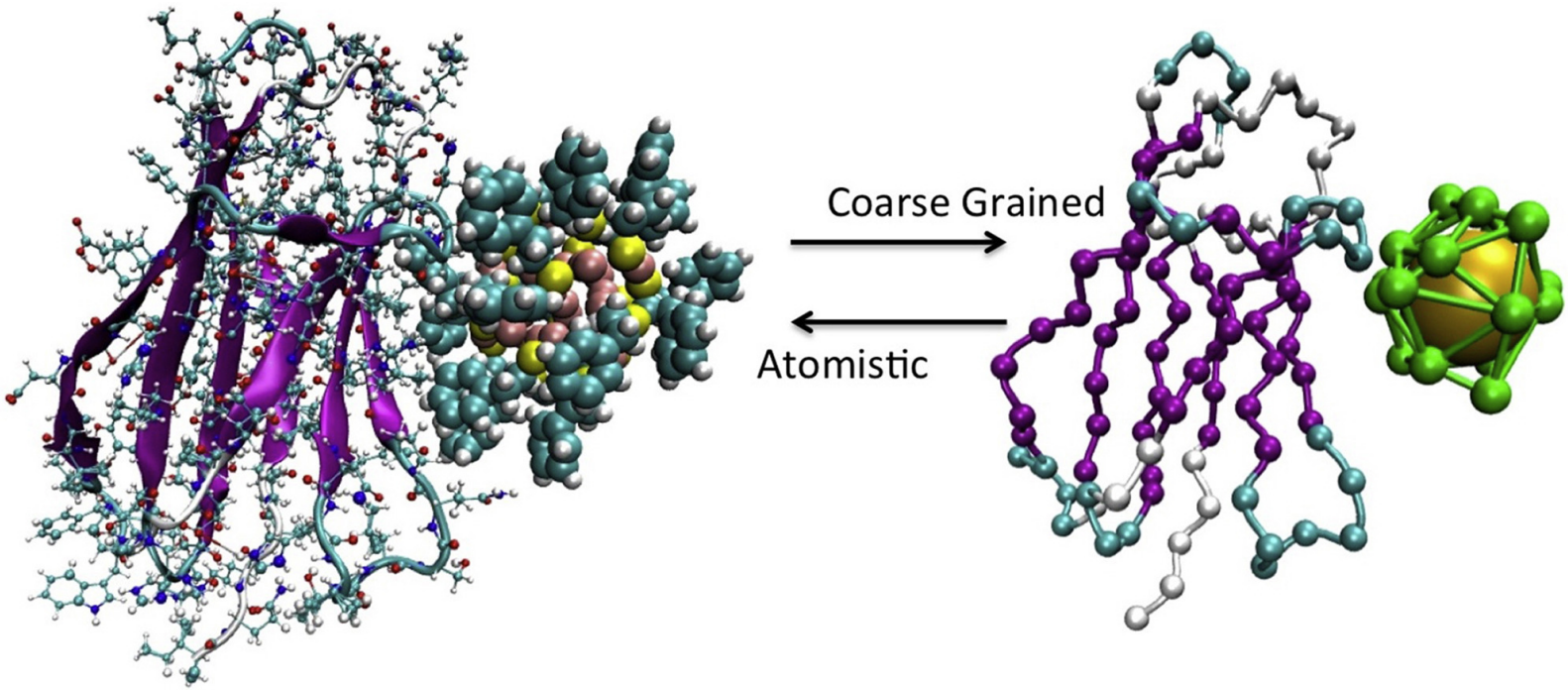
Material science: Constrained generative models to generate stable crystal structures by optimizing the formation energy in the latent space.



This figure is taken from (Shunning Li et al., 2021).

Future works (3)

Proteins: Multiscale modeling of proteins for the purpose of function prediction and protein design.



This figure is taken from (Giorgia Brancolini & Valentina Tozzini, 2019).

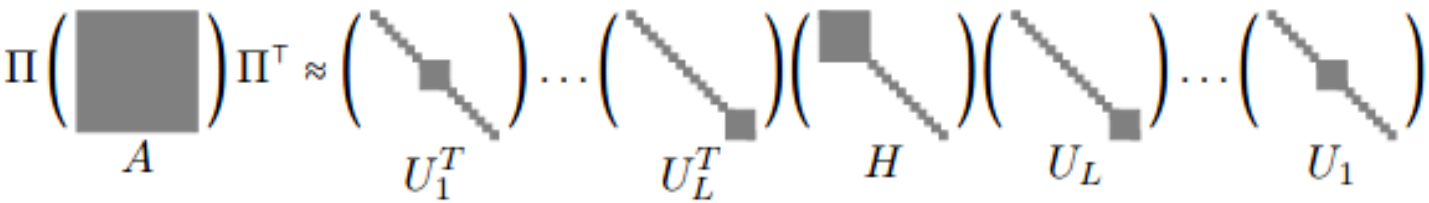
Multiresolution Matrix Factorization (1)

The *Multiresolution Matrix Factorization* (MMF) of a symmetric matrix $\boldsymbol{A} \in \mathbb{R}^{n \times n}$, introduced by (Kondor et al., 2014), is in the form

$$\boldsymbol{A} = \boldsymbol{U}_1^T \boldsymbol{U}_2^T \dots \boldsymbol{U}_L^T \boldsymbol{H} \boldsymbol{U}_L \dots \boldsymbol{U}_2 \boldsymbol{U}_1,$$

where:

- Each $\boldsymbol{U}_\ell$ is an orthogonal matrix that is a k-point rotation (small k),
- There is a nested sequence of sets $\mathbb{S}_L \subseteq \cdots \subseteq \mathbb{S}_1 \subseteq \mathbb{S}_0 = [n]$ such that the coordinates rotated by $\boldsymbol{U}_\ell$ are a subset of $\mathbb{S}_\ell$,
- $\boldsymbol{H}$ is an $\mathbb{S}_L$-core-diagonal matrix meaning that is diagonal with a an additional small $\mathbb{S}_L \times \mathbb{S}_L$ dimensional "core".



Not based on the low-rank assumption

Multiresolution Matrix Factorization (2)

Finding the best MMF to a symmetric matrix $\boldsymbol{A}$ involves solving

$$\min_{\substack{\mathbb{S}_L \subseteq \cdots \subseteq \mathbb{S}_1 \subseteq \mathbb{S}_0 = [n] \\ \boldsymbol{H} \in \mathbb{H}_n^{\mathbb{S}_L}; \boldsymbol{U}_1, \ldots, \boldsymbol{U}_L \in \mathbb{O}}} ||\boldsymbol{A} - \boldsymbol{U}_1^T \ldots \boldsymbol{U}_L^T \boldsymbol{H} \boldsymbol{U}_L \ldots \boldsymbol{U}_1||.$$

Assuming the Frobenius norm, it is equivalent to

$$\min_{\substack{\mathbb{S}_L \subseteq \cdots \subseteq \mathbb{S}_1 \subseteq \mathbb{S}_0 = [n] \\ \boldsymbol{U}_1, \ldots, \boldsymbol{U}_L \in \mathbb{O}}} ||\boldsymbol{U}_L \ldots \boldsymbol{U}_1 \boldsymbol{A} \boldsymbol{U}_1^T \ldots \boldsymbol{U}_L^T||_{\text{resi}}^2,$$

where $||\cdot||_{\text{resi}}^2$ is the squared residual norm

$$||\boldsymbol{H}||_{\text{resi}}^2 = \sum_{i \neq j; (i,j) \notin \mathbb{S}_L \times \mathbb{S}_L} |\boldsymbol{H}_{i,j}|^2.$$

Multiresolution Matrix Factorization (3)

There are two fundamental difficulties in MMF optimization:

1. Finding the optimal nested sequence of $\mathbb{S}_\ell$ is a **combinatorially hard** (e.g., there are $\binom{d_\ell}{k}$ ways to choose k indices out of $\mathbb{S}_\ell$);
2. The solution for $\boldsymbol{U}_\ell$ must satisfy the orthogonality constraint such that $\boldsymbol{U}_\ell^T \boldsymbol{U}_\ell = \boldsymbol{I}$.

Multiresolution Matrix Factorization (3)

There are two fundamental difficulties in MMF optimization:

1. Finding the optimal nested sequence of $\mathbb{S}_\ell$ is a **combinatorially hard** (e.g., there are $\binom{d_\ell}{k}$ ways to choose k indices out of $\mathbb{S}_\ell$);
2. The solution for $\boldsymbol{U}_\ell$ must satisfy the orthogonality constraint such that $\boldsymbol{U}_\ell^T \boldsymbol{U}_\ell = \boldsymbol{I}$.

Existing greedy algorithms to approximate MMF (Kondor et al., 2014)

Limitations:

- Greedy heuristics (e.g., clustering) used in selecting k rows/columns for each rotation. Not globally optimal sequence of $\mathbb{S}_\ell$.

Multiresolution Matrix Factorization (3)

There are two fundamental difficulties in MMF optimization:

1. Finding the optimal nested sequence of $\mathbb{S}_\ell$ is a **combinatorially hard** (e.g., there are $\binom{d_\ell}{k}$ ways to choose k indices out of $\mathbb{S}_\ell$);
2. The solution for $\boldsymbol{U}_\ell$ must satisfy the orthogonality constraint such that $\boldsymbol{U}_\ell^T \boldsymbol{U}_\ell = \boldsymbol{I}$.

Existing greedy algorithms to approximate MMF (Kondor et al., 2014)

Limitations:

- Greedy heuristics (e.g., clustering) used in selecting k rows/columns for each rotation. Not globally optimal sequence of $\mathbb{S}_\ell$.
- Iterative algorithm to optimize only 1 rotation at a time. Not globally optimal sequence of $\boldsymbol{U}_\ell$.

Multiresolution Matrix Factorization (3)

There are two fundamental difficulties in MMF optimization:

1. Finding the optimal nested sequence of $\mathbb{S}_\ell$ is a **combinatorially hard** (e.g., there are $\binom{d_\ell}{k}$ ways to choose k indices out of $\mathbb{S}_\ell$);
2. The solution for $\boldsymbol{U}_\ell$ must satisfy the orthogonality constraint such that $\boldsymbol{U}_\ell^T \boldsymbol{U}_\ell = \boldsymbol{I}$.

Existing greedy algorithms to approximate MMF (Kondor et al., 2014)

Limitations:

- Greedy heuristics (e.g., clustering) used in selecting k rows/columns for each rotation. Not globally optimal sequence of $\mathbb{S}_\ell$.
- Iterative algorithm to optimize only 1 rotation at a time. Not globally optimal sequence of $\boldsymbol{U}_\ell$.
- Limited to only $k = 2$. Bigger rotation matrices are better!!

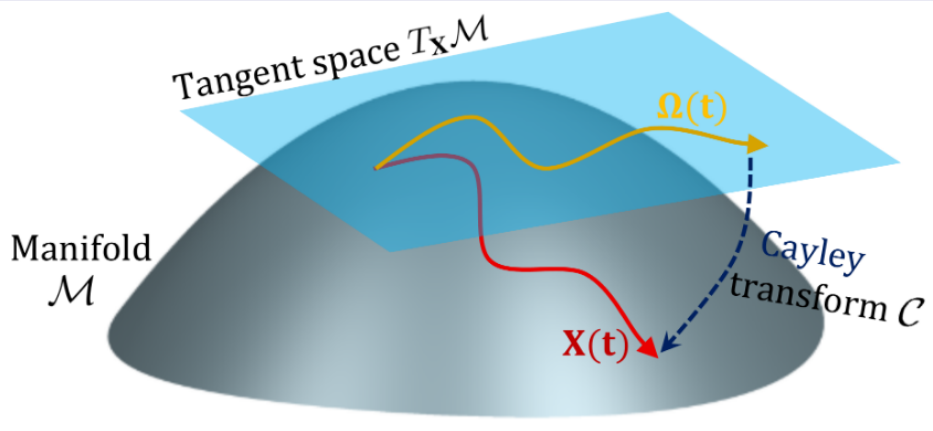
Learnable MMF (1)

The MMF optimization problem is equivalent to

$$\min_{\mathbb{S}_L \subseteq \cdots \subseteq \mathbb{S}_1 \subseteq \mathbb{S}_0 = [n]} \; \min_{\boldsymbol{U}_1, \ldots, \boldsymbol{U}_L \in \mathbb{O}} \; || \boldsymbol{U}_L \ldots \boldsymbol{U}_1 \boldsymbol{A} \boldsymbol{U}_1^T \ldots \boldsymbol{U}_L^T ||_{\text{resi}}^2,$$

Proposal 1 – Stiefel manifold optimization

Given a fixed $\mathbb{S}_L \subseteq \cdots \subseteq \mathbb{S}_1 \subseteq \mathbb{S}_0 = [n]$, we use gradient descent algorithm on the Stiefel manifold to optimize all rotations $\{\boldsymbol{U}_\ell\}_{\ell=1}^L$ *simultaneously*, whilst satisfying the orthogonality constraints.



Learnable MMF (2)

The MMF optimization problem is equivalent to

$$\min_{\mathbb{S}_L \subseteq \cdots \subseteq \mathbb{S}_1 \subseteq \mathbb{S}_0 = [n]} \min_{\boldsymbol{U}_1, \ldots, \boldsymbol{U}_L \in \mathbb{O}} ||\boldsymbol{U}_L \ldots \boldsymbol{U}_1 \boldsymbol{A} \boldsymbol{U}_1^T \ldots \boldsymbol{U}_L^T||_{\text{resi}}^2,$$

Proposal 2 – Reinforcement Learning

- Formulate the problem of finding the nested sequence of indices as learning a **Markov Decision Process** (MDP).

Learnable MMF (2)

The MMF optimization problem is equivalent to

$$\min_{\mathbb{S}_L \subseteq \cdots \subseteq \mathbb{S}_1 \subseteq \mathbb{S}_0 = [n]} \; \min_{\boldsymbol{U}_1, \ldots, \boldsymbol{U}_L \in \mathbb{O}} ||\boldsymbol{U}_L \ldots \boldsymbol{U}_1 \boldsymbol{A} \boldsymbol{U}_1^T \ldots \boldsymbol{U}_L^T||_{\text{resi}}^2,$$

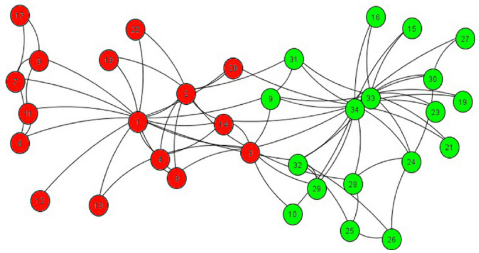
Proposal 2 – Reinforcement Learning

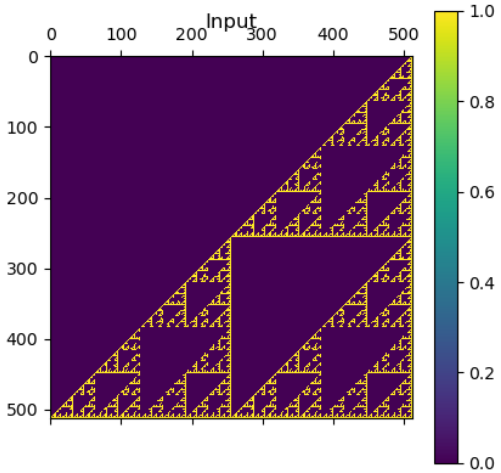
- Formulate the problem of finding the nested sequence of indices as learning a **Markov Decision Process** (MDP).
- **REINFORCE** – Gradient (stochastic) policy method of Reinforcement Learning (RL).
- The RL agent is modeled by **graph neural networks** (GNN).

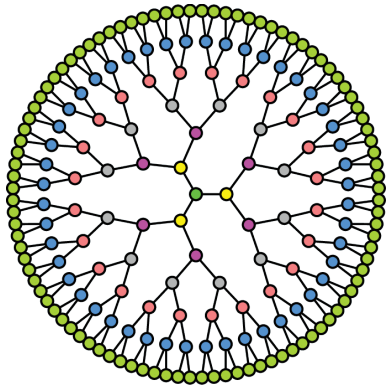
Note: RL has been applied to solve combinatorial, NP-hard, graph problems such as Traveling Saleman (TSP), etc.

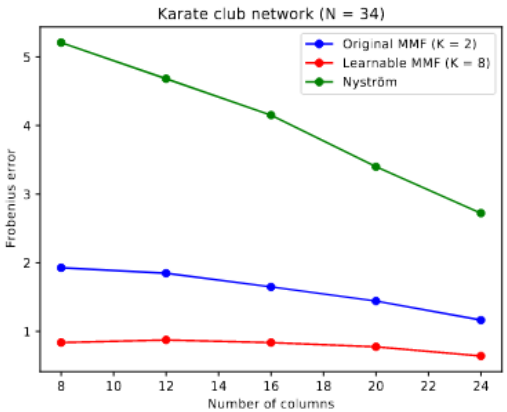
Learnable MMF (3)

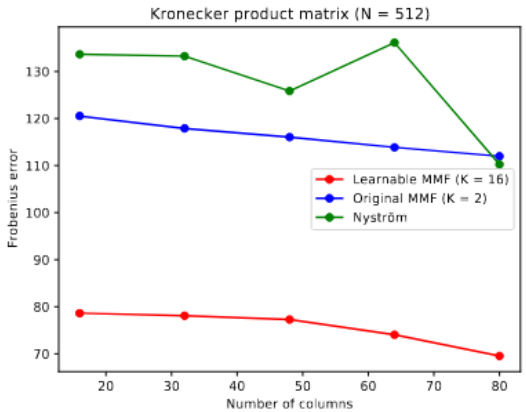
Learnable MMF (red) outperforms the **original MMF** (blue) and the **Nyström method** (green) in matrix approximation:

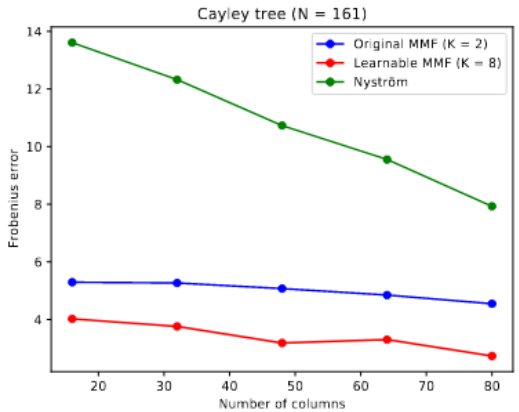












Karate club network | Kronecker matrix | Cayley tree

Graph Fourier Transform (1)

The eigendecomposition of the normalized graph Laplacian operator $\tilde{\boldsymbol{L}} = \boldsymbol{U}^T \boldsymbol{H} \boldsymbol{U}$ can be used as the basis of a **Graph Fourier Transform** (GFT) of a graph signal/function $\boldsymbol{f} : V \to \mathbb{R}$ (Shuman et al., 2013):

Graph Fourier Transform (1)

The eigendecomposition of the normalized graph Laplacian operator $\tilde{\boldsymbol{L}} = \boldsymbol{U}^T \boldsymbol{H} \boldsymbol{U}$ can be used as the basis of a **Graph Fourier Transform** (GFT) of a graph signal/function $\boldsymbol{f} : V \to \mathbb{R}$ (Shuman et al., 2013):

- Forward: $\hat{\boldsymbol{f}} = \boldsymbol{U}^T \boldsymbol{f}$
- Inverse: $\boldsymbol{f} = \boldsymbol{U} \hat{\boldsymbol{f}}$

Graph Fourier Transform (1)

The eigendecomposition of the normalized graph Laplacian operator $\tilde{\boldsymbol{L}} = \boldsymbol{U}^T \boldsymbol{H} \boldsymbol{U}$ can be used as the basis of a **Graph Fourier Transform** (GFT) of a graph signal/function $\boldsymbol{f} : V \to \mathbb{R}$ (Shuman et al., 2013):

- Forward: $\hat{\boldsymbol{f}} = \boldsymbol{U}^T \boldsymbol{f}$
- Inverse: $\boldsymbol{f} = \boldsymbol{U} \hat{\boldsymbol{f}}$
- Graph convolution: $\boldsymbol{f} *_{\mathcal{G}} \boldsymbol{g} = \boldsymbol{U}\big((\boldsymbol{U}^T \boldsymbol{g}) \odot (\boldsymbol{U}^T \boldsymbol{f})\big)$, where $\boldsymbol{g}$ denotes the convolution kernel, and $\odot$ is the element-wise Hadamard product.

Graph Fourier Transform (1)

The eigendecomposition of the normalized graph Laplacian operator $\tilde{\boldsymbol{L}} = \boldsymbol{U}^T \boldsymbol{H} \boldsymbol{U}$ can be used as the basis of a **Graph Fourier Transform** (GFT) of a graph signal/function $\boldsymbol{f} : V \to \mathbb{R}$ (Shuman et al., 2013):

- Forward: $\hat{\boldsymbol{f}} = \boldsymbol{U}^T \boldsymbol{f}$
- Inverse: $\boldsymbol{f} = \boldsymbol{U} \hat{\boldsymbol{f}}$
- Graph convolution: $\boldsymbol{f} *_{\mathcal{G}} \boldsymbol{g} = \boldsymbol{U}\big((\boldsymbol{U}^T \boldsymbol{g}) \odot (\boldsymbol{U}^T \boldsymbol{f})\big)$, where $\boldsymbol{g}$ denotes the convolution kernel, and $\odot$ is the element-wise Hadamard product.
- Based on GFT, (Bruna et al., 2014) and (Defferrard et al., 2016) construct convolutional neural networks (CNNs) **learning on spectral domain** of graphs.

Graph Fourier Transform (2)

Limitations of GFT

- High computational cost:
  - Eigendecomposition of the graph Laplacian has complexity $O(n^3)$
  - "Fourier transform" itself involves multiplying the signal with a **dense** matrix of eigenvectors.
- The graph convolution is **not localized** in the vertex domain, even if the graph itself has well defined local communities.

Graph Fourier Transform (2)

Limitations of GFT

- High computational cost:
  - Eigendecomposition of the graph Laplacian has complexity $O(n^3)$
  - "Fourier transform" itself involves multiplying the signal with a **dense** matrix of eigenvectors.
- The graph convolution is **not localized** in the vertex domain, even if the graph itself has well defined local communities.

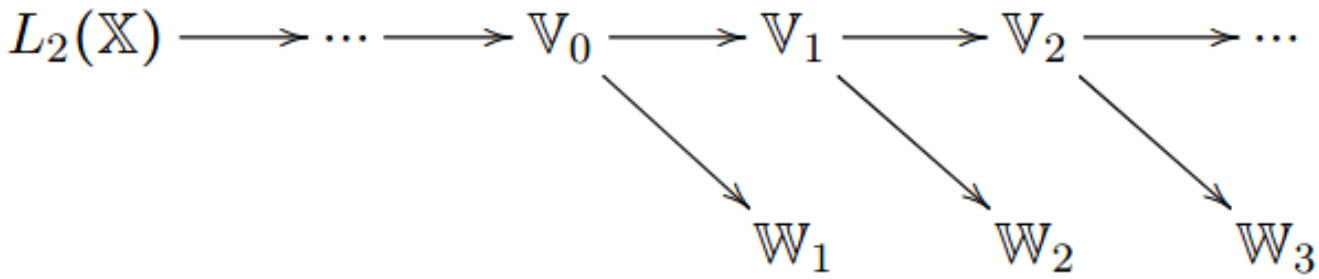
Multiresolution Analysis (MRA)

The functional analytic view of wavelets is provided by Multiresolution Analysis (Mallat, 1989), which, similarly to Fourier analysis, is a way of filtering some function space into a sequence of subspaces

$$\cdots \subset \mathbb{V}_{-1} \subset \mathbb{V}_0 \subset \mathbb{V}_1 \subset \mathbb{V}_2 \subset \dots$$

Multiresolution analysis (1)

Iteratively, each $\mathbb{V}_\ell$ is splitted into the orthogonal sum $\mathbb{V}_\ell = \mathbb{V}_{\ell+1} \oplus \mathbb{W}_{\ell+1}$ of a smoother part $\mathbb{V}_{\ell+1}$, called the *approximation space*; and a rougher part $\mathbb{W}_{\ell+1}$, called the *detail space*:



Each $\mathbb{V}_\ell$ has an orthonormal basis $\Phi_\ell \triangleq \{\phi_m^\ell\}_m$ in which each ϕ is called a *father* wavelet. Each complementary space $\mathbb{W}_\ell$ is also spanned by an orthonormal basis $\Psi_\ell \triangleq \{\psi_m^\ell\}_m$ in which each ψ is called a *mother* wavelet.

Multiresolution analysis (2)

Instead of diagonalizing $\boldsymbol{A}$ in a single step as in PCA, multiresolution analysis will involve a sequence of basis transforms $\boldsymbol{U}_1, \boldsymbol{U}_2, \ldots, \boldsymbol{U}_L$, transforming $\boldsymbol{A}$ step by step as:

$$\boldsymbol{A} \to \boldsymbol{U}_1 \boldsymbol{A} \boldsymbol{U}_1^T \to \boldsymbol{U}_2 \boldsymbol{U}_1 \boldsymbol{A} \boldsymbol{U}_1^T \boldsymbol{U}_2^T \to \cdots \to \boldsymbol{U}_L \ldots \boldsymbol{U}_2 \boldsymbol{U}_1 \boldsymbol{A} \boldsymbol{U}_1^T \boldsymbol{U}_2^T \ldots \boldsymbol{U}_L^T,$$

Multiresolution analysis (2)

Instead of diagonalizing $\boldsymbol{A}$ in a single step as in PCA, multiresolution analysis will involve a sequence of basis transforms $\boldsymbol{U}_1, \boldsymbol{U}_2, \ldots, \boldsymbol{U}_L$, transforming $\boldsymbol{A}$ step by step as:

$$\boldsymbol{A} \to \boldsymbol{U}_1 \boldsymbol{A} \boldsymbol{U}_1^T \to \boldsymbol{U}_2 \boldsymbol{U}_1 \boldsymbol{A} \boldsymbol{U}_1^T \boldsymbol{U}_2^T \to \cdots \to \boldsymbol{U}_L \ldots \boldsymbol{U}_2 \boldsymbol{U}_1 \boldsymbol{A} \boldsymbol{U}_1^T \boldsymbol{U}_2^T \ldots \boldsymbol{U}_L^T,$$

Each individual rotation $\boldsymbol{U}_\ell : \mathbb{V}_{\ell-1} \to \mathbb{V}_\ell \oplus \mathbb{W}_\ell$ is a sparse basis transform that expresses $\Phi_\ell \cup \Psi_\ell$ in the previous basis $\Phi_{\ell-1}$ such that:

$$\phi_m^\ell = \sum_{i=1}^{\dim(\mathbb{V}_{\ell-1})} [\boldsymbol{U}_\ell]_{m,i} \phi_i^{\ell-1}, \qquad \psi_m^\ell = \sum_{i=1}^{\dim(\mathbb{V}_{\ell-1})} [\boldsymbol{U}_\ell]_{m+\dim(\mathbb{V}_{\ell-1}),i} \phi_i^{\ell-1}.$$

Multiresolution analysis (3)

In the case $\boldsymbol{A}$ is the normalized graph Laplacian of a graph $\mathcal{G} = (V, E)$, the wavelet transform (up to level L) expresses a graph signal (function over the vertex domain) $f : V \to \mathbb{R}$, without loss of generality $f \in \mathbb{V}_0$, as:

$$f(v) = \sum_{\ell=1}^{L} \sum_{m} \alpha_m^\ell \psi_m^\ell(v) + \sum_{m} \beta_m \phi_m^L(v), \qquad \text{for each} \quad v \in V,$$

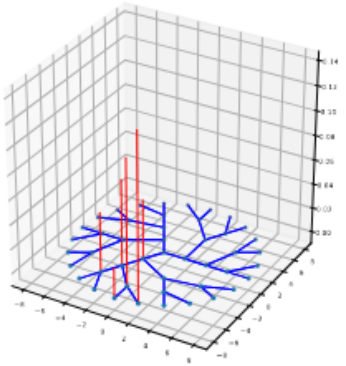
where $\alpha_m^\ell = \langle f, \psi_m^\ell \rangle$ and $\beta_m = \langle f, \phi_m^L \rangle$ are the wavelet coefficients.

Multiresolution analysis (4)

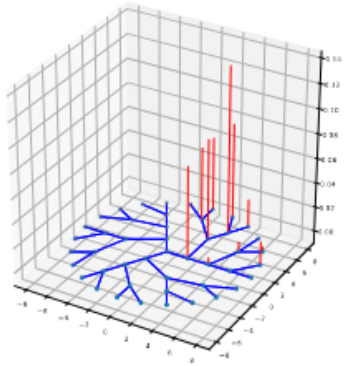
MMF gives us a total of N wavelets:

- L mother wavelets $\overline{\psi} = \{\psi^1, .., \psi^L\}$,
- $N - L$ father wavelets $\overline{\phi} = \{\phi^L_m = \boldsymbol{H}_{m,:}\}_{m \in \mathbb{S}_L}$.

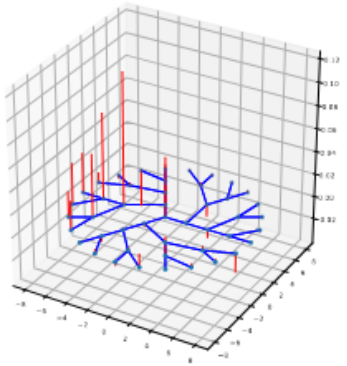
Visualization of some of the wavelets on the Cayley tree of 46 vertices. Thelow index wavelets (low ℓ) are **highly localized**, whereas the high index ones are smoother and spread out over large parts of the graph.



$\ell = 1$



$\ell = 20$



$\ell = 39$

Wavelet Neural Networks (1)

Analogous to the convolution based on GFT (Bruna et al., 2014), each convolution layer $k = 1, .., K$ of our wavelet network transforms an input vector $\boldsymbol{f}^{(k-1)}$ of size $|V| \times F_{k-1}$ into an output $\boldsymbol{f}^{(k)}$ of size $|V| \times F_k$ as

$$\boldsymbol{f}_{:,j}^{(k)} = \sigma\left(\boldsymbol{W} \sum_{i=1}^{F_{k-1}} \boldsymbol{g}_{i,j}^{(k)} \boldsymbol{W}^T \boldsymbol{f}_{:,i}^{(k-1)}\right) \quad \text{for} \quad j = 1, \ldots, F_k,$$

where $\boldsymbol{W} = [\overline{\phi}, \overline{\psi}]$ is our wavelet basis matrix, $\boldsymbol{g}_{i,j}^{(k)}$ is a parameter/filter in the form of a diagonal matrix, and σ is an element-wise linearity.

Fast Wavelet Transform

Since the wavelet basis is **sparse**, the wavelet transform can be implemented efficiently by sparse matrix multiplication.

Wavelet Neural Networks (2)

Only 4.69% and 15.25% of elements of the wavelet basis are non-zero in Citeseer and Cora, while the GFT basis is completely dense.

Table 1: Node classification on citation graphs. Baseline results are taken from (Xu et al., 2019).

Method	Cora	Citeseer
MLP	55.1%	46.5%
ManiReg (Belkin et al., 2006)	59.5%	60.1%
SemiEmb (Weston et al., 2008)	59.0%	59.6%
LP (Zhu et al., 2003)	68.0%	45.3%
DeepWalk (Perozzi et al., 2014)	67.2%	43.2%
ICA (Getoor, 2005)	75.1%	69.1%
Planetoid (Yang et al., 2016)	75.7%	64.7%
Spectral CNN (Bruna et al., 2014)	73.3%	58.9%
ChebyNet (Defferrard et al., 2016)	81.2%	69.8%
GCN (Kipf and Welling, 2017)	81.5%	70.3%
MoNet (Monti et al., 2017)	81.7%	N/A
GWNN (Xu et al., 2019)	82.8%	71.7%
$\mathbf{MMF}_1$	**84.35%**	68.07%
$\mathbf{MMF}_2$	**84.55%**	**72.76%**
$\mathbf{MMF}_3$	**87.59%**	**72.90%**

(1) 20%/20%/60%, (2) 40%/20%/40%, (3) 60%/20%/20%

(Hy & Kondor, 2021)

Wavelet Neural Networks (3)

Average percentages of non-zero elements of the wavelet basis:
19.23% (MUTAG), 18.18% (PTC), 2.26% (PROTEINS), 11.43% (NCI1)

Table 2: Graph classification. Baseline results are taken from (Maron et al., 2019).

Method	MUTAG	PTC	PROTEINS	NCI1
DGCNN (Zhang et al., 2018)	85.83 ± 1.7	58.59 ± 2.5	75.54 ± 0.9	74.44 ± 0.5
PSCN (Niepert et al., 2016)	88.95 ± 4.4	62.29 ± 5.7	75 ± 2.5	76.34 ± 1.7
DCNN (Atwood and Towsley, 2016)	N/A	N/A	61.29 ± 1.6	56.61 ± 1.0
CCN (Kondor et al., 2018)	**91.64 ± 7.2**	**70.62 ± 7.0**	N/A	76.27 ± 4.1
GK (Shervashidze et al., 2009)	81.39 ± 1.7	55.65 ± 0.5	71.39 ± 0.3	62.49 ± 0.3
RW (Vishwanathan et al., 2010)	79.17 ± 2.1	55.91 ± 0.3	59.57 ± 0.1	N/A
PK (Neumann et al., 2015)	76 ± 2.7	59.5 ± 2.4	73.68 ± 0.7	82.54 ± 0.5
WL (Shervashidze et al., 2011)	84.11 ± 1.9	57.97 ± 2.5	74.68 ± 0.5	**84.46 ± 0.5**
IEGN (Maron et al., 2019)	84.61 ± 10	59.47 ± 7.3	75.19 ± 4.3	73.71 ± 2.6
MMF	86.31 ± 9.47	67.99 ± 8.55	**78.72 ± 2.53**	71.04 ± 1.53

Our WNNs outperform 7/8, 7/8, 8/8, and 2/8 baseline methods on these datasets, respectively.

(Hy & Kondor, 2021)

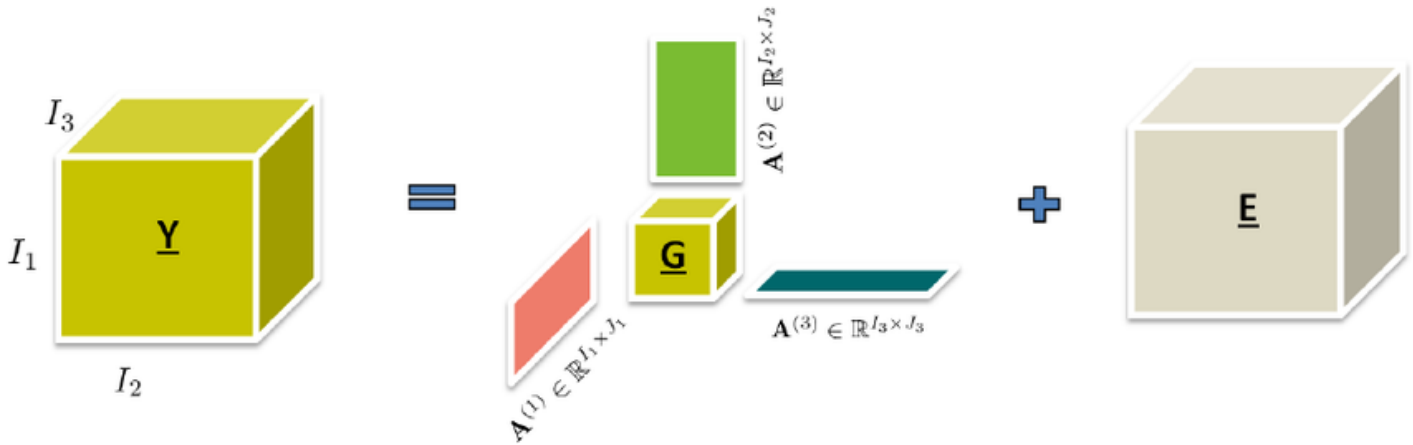
Software & Future works

Software:

`https://github.com/risilab/Learnable_MMF`

Future works:

1. *Multiresolution Tensor Factorization:*



Tensor factorization. Figure taken from (Zhou & Cichocki, 2012)

2. *Hierarchical data visualization*: "wavelet" t-SNE.

Q&A

Thank you for your attention!