

Higher order graph neural networks with P-tensors

Risi Kondor



THE UNIVERSITY OF
CHICAGO



Andrew
Hands



Tianyi
Sun



Richard
Xu



Qingqi
Zhang

1. Why I like GNNs.

2. Why I don't like GNNs.

Equivariance

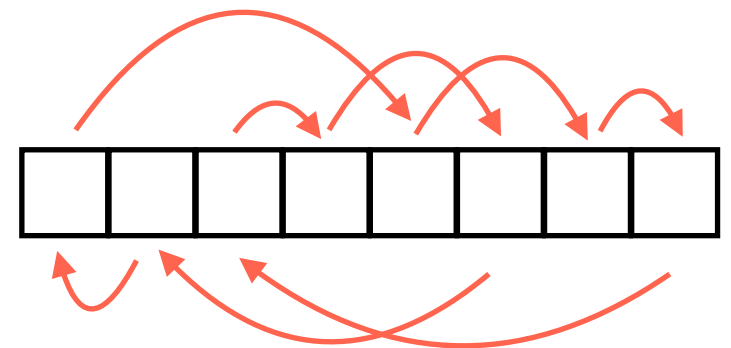
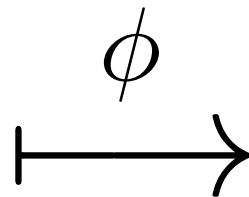
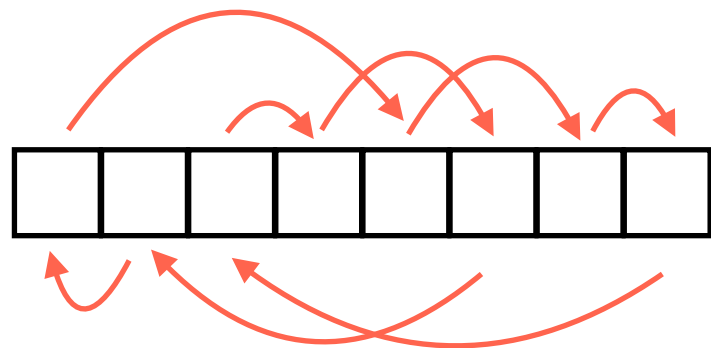
$$\begin{array}{ccc} f^{\text{in}} & \xrightarrow{T_g^{(1)}} & f^{\text{in}'} \\ \downarrow \phi & & \downarrow \phi \\ f^{\text{out}} & \xrightarrow{T_g^{(2)}} & f^{\text{out}'} \end{array}$$

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

First order permutation equivariance

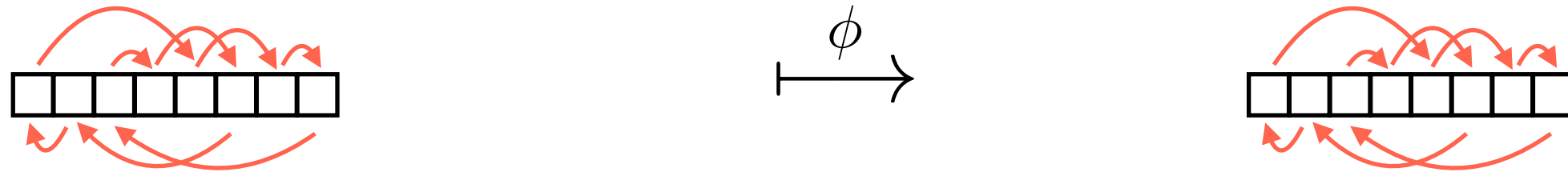
$$f \xrightarrow{\phi} f'$$

$$(f^\sigma)' = (f')^\sigma$$



[Deep sets: Zaheer et al., 2017]

First order permutation equivariance



$$f_i \xrightarrow{\alpha_1} f'_i$$

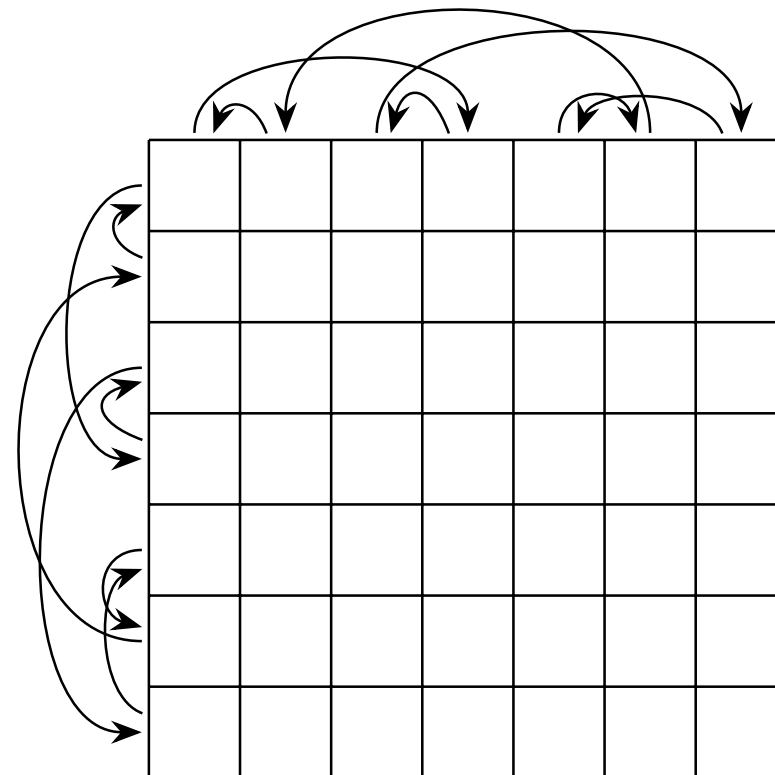
$$\frac{1}{n} \sum_i f_i \xrightarrow{\alpha_2} \frac{1}{n} \sum_i f'_i$$

$$f'_i = \alpha_1 f_i + \alpha_2 \frac{1}{n} \sum_j f_j$$

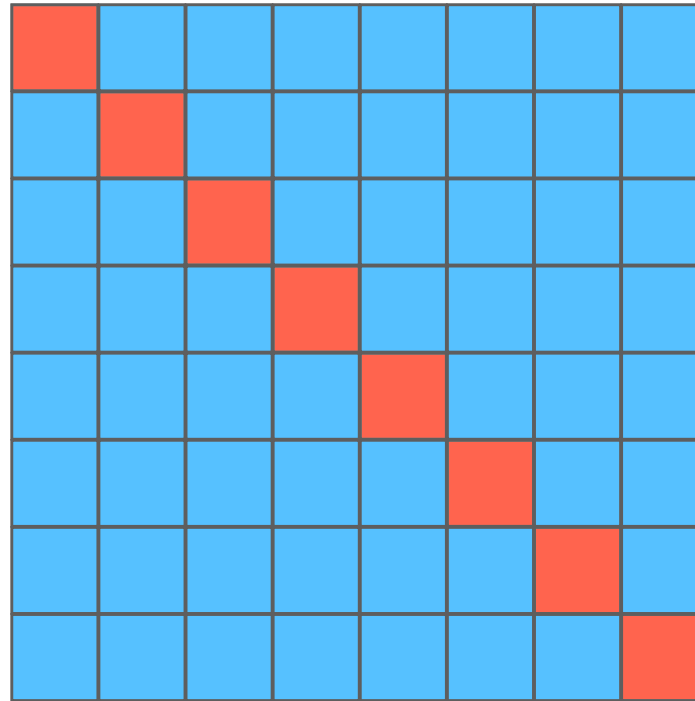
Second order permutation action

$$A \mapsto A^\sigma$$

$$A_{i,j}^\sigma = A_{\sigma^{-1}(i), \sigma^{-1}(j)}$$



Second order permutation action



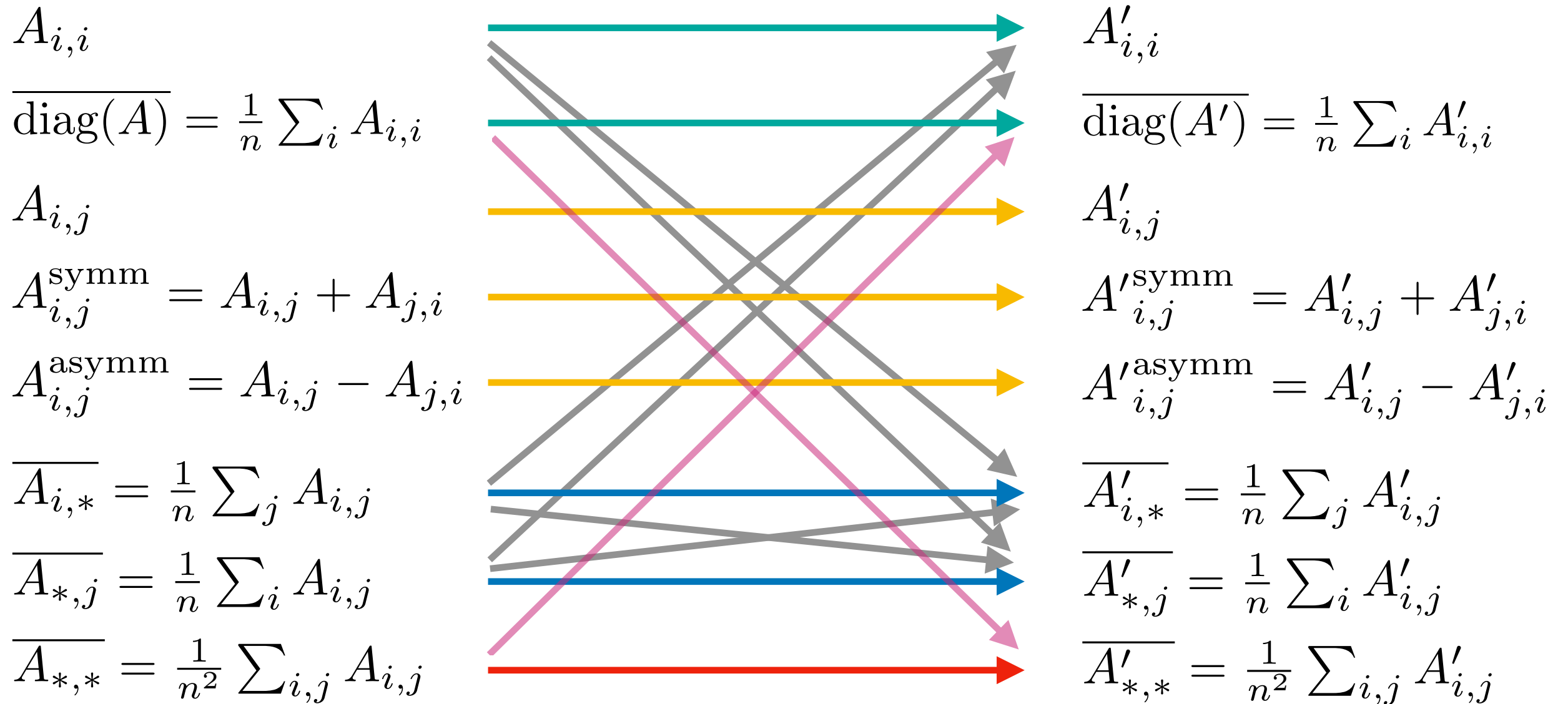
Orbit 1:

$$(i, i) \mapsto (\sigma(i), \sigma(i))$$

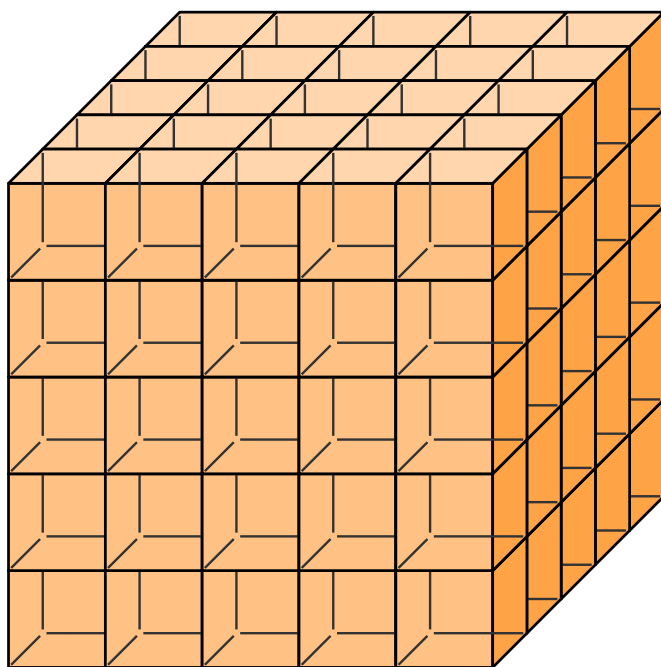
Orbit 2:

$$(i, j) \mapsto (\sigma(i), \sigma(j))$$

Second order permutation equivariance

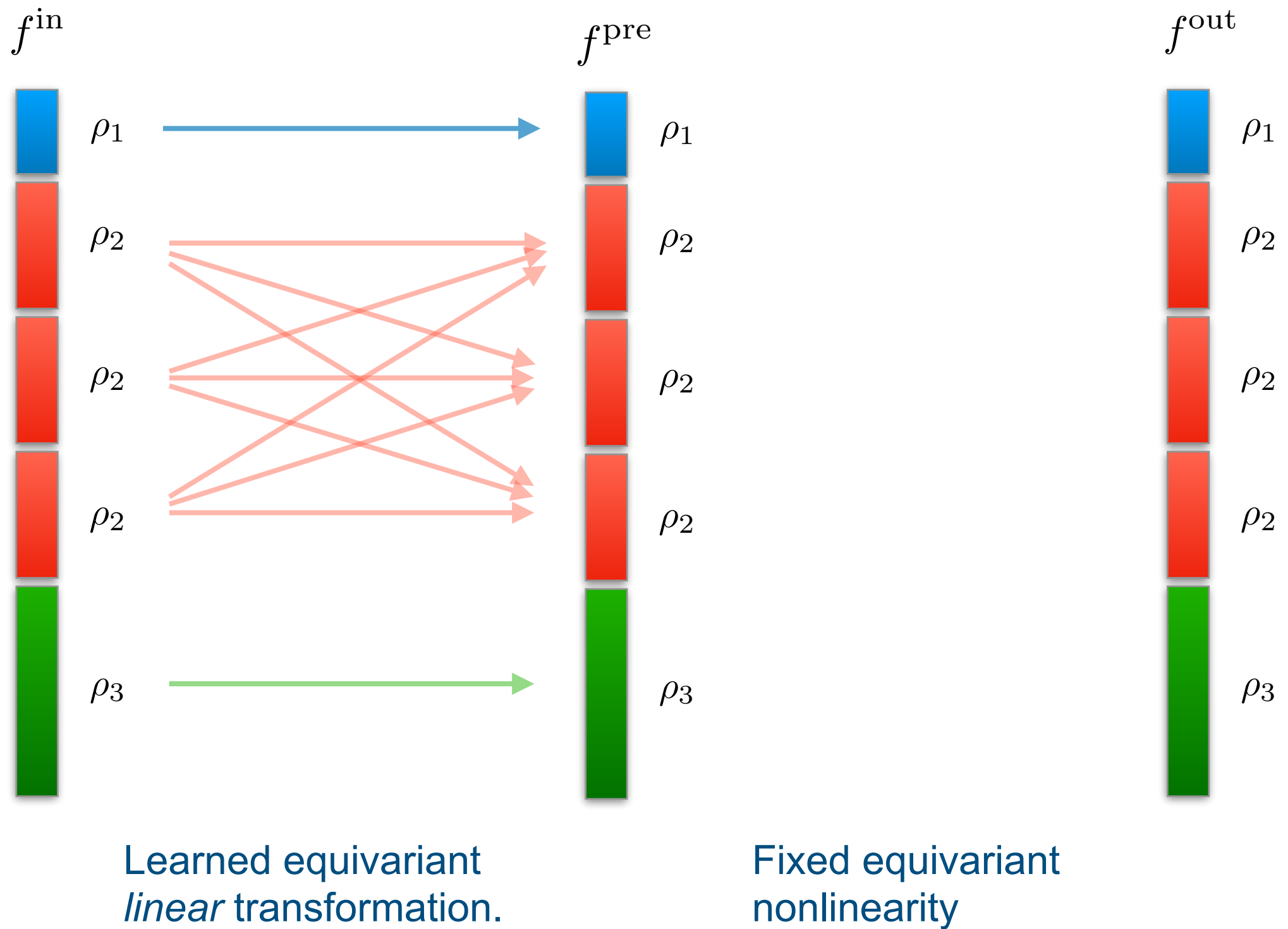


$$\begin{aligned}
 A'_{i,j} = & \delta_{i,j} \left(\alpha_1 A_{i,j} + \alpha_2 \frac{1}{n} \sum_i A_{i,i} \right) + \alpha_3 (A_{i,j} + A_{j,i})/2 + \alpha_4 (A_{i,j} - A_{j,i})/2 \\
 & + \alpha_5 \overline{A_{i,*}} + \alpha_6 \overline{A_{*,j}} + \alpha_7 \overline{A_{*,*}} + \dots
 \end{aligned}$$



$$T \xrightarrow{\sigma} T' \qquad [T']_{i_1, \dots, i_k} = [T]_{\sigma^{-1}(i_1), \dots, \sigma^{-1}(i_k)}.$$

Equivariant neuron



Representations of S_n

The **permutation matrices**

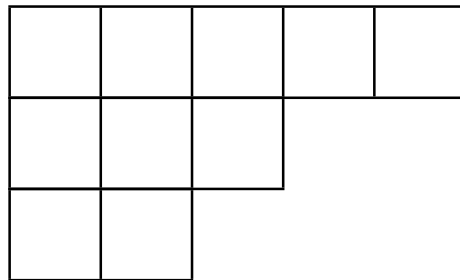
$$P_{i,j}^\sigma = \begin{cases} 1 & \text{if } \sigma(j) = i \\ 0 & \text{otherwise} \end{cases}$$

form a representation ρ_{def} of S_n .

Is it reducible? **No!**

Young diagrams

A **Young diagram** (Ferrers diagram) of size n is a diagram of n boxes arranged in k left-justified rows so that no row is longer than the one above it, e.g.,

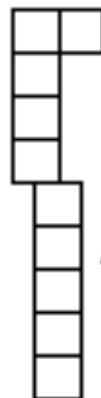
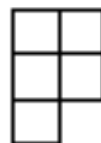
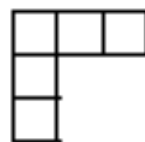
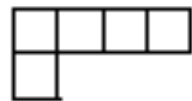
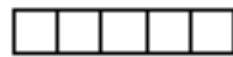


A Young diagram is really just the pictorial representation of an **integer partition**, i.e., a sequence $(\lambda_1, \lambda_2, \dots, \lambda_k)$ such that $\sum_i \lambda_i = n$.

Young diagrams

The Young diagrams (integer partitions) of n are in bijection with the irreducible representations of \mathbb{S}_n .

For example, the irreps of \mathbb{S}_5 are indexed by:



Young tableaux

A **standard Young tableau** is a Young diagram filled with the numbers $\{1, 2, \dots, n\}$ in such a way that in each row the numbers increase left to right and in each column they increase top to bottom.

For example,

1	2	4	6	7
3	5	8		
9				

is a standard Young tableau of shape $\lambda = (5, 3, 1)$.

The rows and columns of the irrep ρ_λ are in bijection with the standard Young tableaux of shape λ .

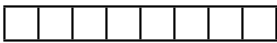
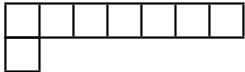
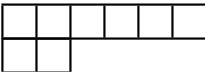
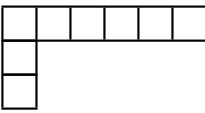
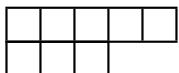
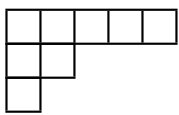
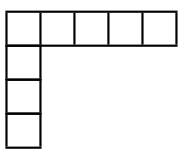
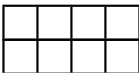
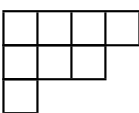
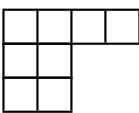
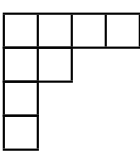
Young's orthogonal representation (YOR)

Defined in terms of adjacent transpositions $\tau_i = (i, i + 1)$, which act naturally on Young tableaux by swapping i and $i + 1$ (if legal). The matrix elements are given explicitly as:

$$[\rho_\lambda(\tau_i)]_{t',t} = \begin{cases} 1/d_t(i, i + 1) & \text{if } t = t' \\ \sqrt{1 - 1/d_t(i, i + 1)^2} & \text{if } t' = \tau_i(t) \\ 0 & \text{otherwise,} \end{cases}$$

where d_t is the North-Easterly distance from i to $i + 1$.

Sizes of the irreps

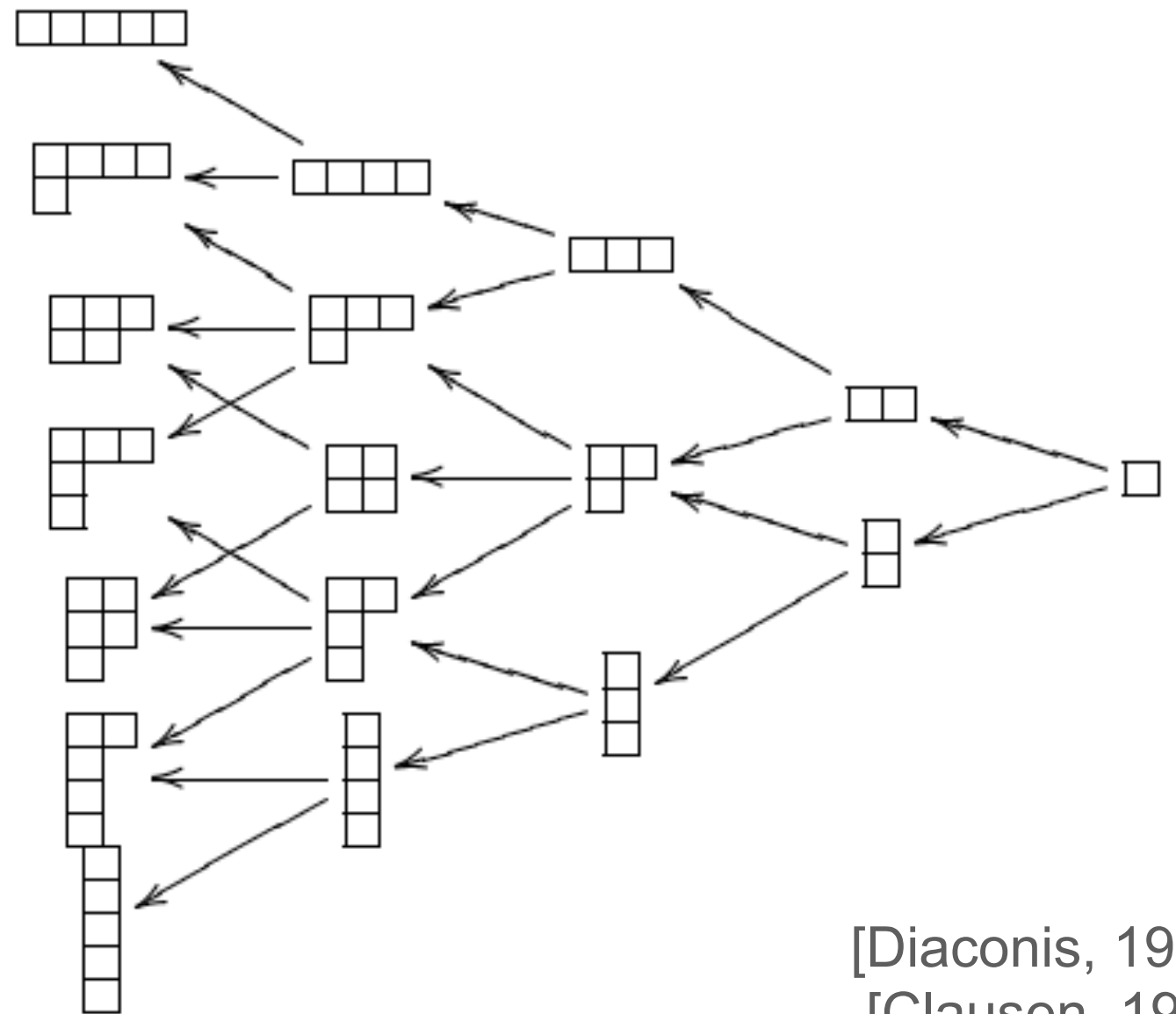
λ		d_λ
(n)		1
$(n-1, 1)$		$n-1$
$(n-2, 2)$		$\frac{n(n-3)}{2}$
$(n-2, 1, 1)$		$\frac{(n-1)(n-2)}{2}$
$(n-3, 3)$		$\frac{n(n-1)(n-5)}{6}$
$(n-3, 2, 1)$		$\frac{n(n-2)(n-4)}{3}$
$(n-3, 1, 1, 1)$		$\frac{(n-1)(n-2)(n-3)}{6}$
$(n-4, 4)$		$\frac{n(n-1)(n-2)(n-7)}{24}$
$(n-4, 3, 1)$		$\frac{n(n-1)(n-3)(n-6)}{8}$
$(n-4, 2, 2)$		$\frac{n(n-1)(n-4)(n-5)}{12}$
$(n-4, 2, 1, 1)$		$\frac{n(n-2)(n-3)(n-5)}{8}$

FFTs on the symmetric group

$$\hat{f}(\rho) = \sum_{\sigma \in \mathbb{S}_n} f(\sigma) \rho(\sigma)$$

$$f(\sigma) = \frac{1}{n!} \sum_{\lambda \vdash n} d_\lambda \operatorname{tr} [\hat{f}(\lambda) \rho_\lambda(\sigma^{-1})].$$

Clausen's FFT reduces the complexity of the FT and iFFT from $O(n!^2)$ to $O(n^3 n!)$.



[Diaconis, 1987]

[Clausen, 1989]

[Thiede, Hy & K, 2020]

C++/Python library: <https://github.com/risi-kondor/Snob2>

Is this way to implement permutation equivariance in GNNs? **No.**

General result

transfer indices

$$T_{a,\beta,\alpha,a,\beta}^{\text{out}} = \sum_x \sum_y T_{\alpha,x,\beta,x,\beta,y}^{\text{in}}$$

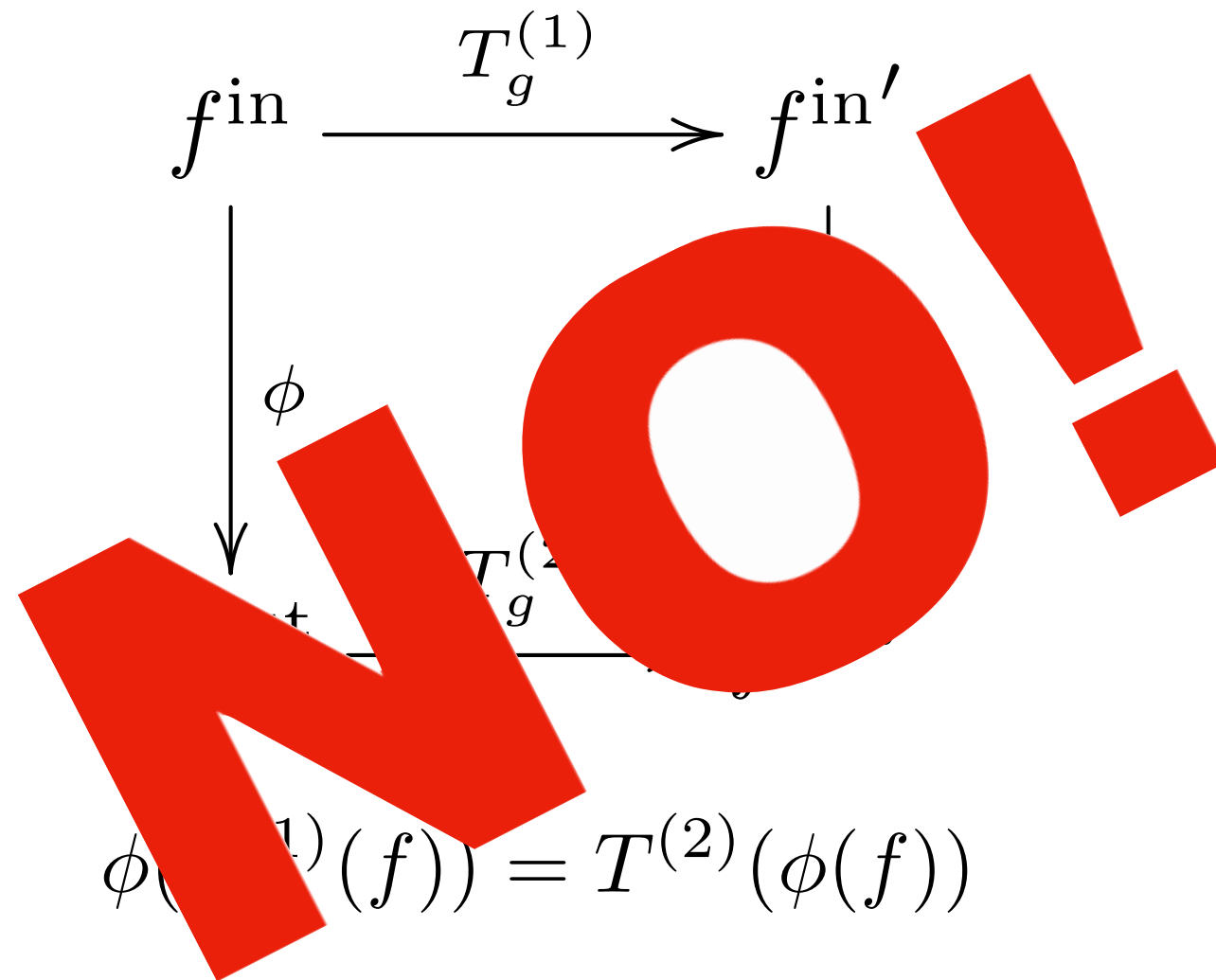
broadcast

reduction

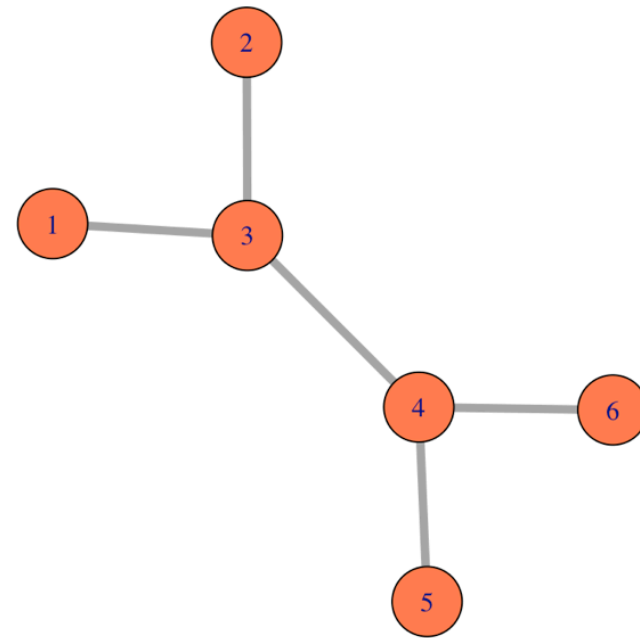
\mathcal{P}	ϕ
$\{\{1\}, \{2\}, \{3\}, \{4\}\}$	$T_{a,b}^{\text{out}} = \sum_{c,d} T_{c,d}^{\text{in}}$
$\{\{1\}, \{2\}, \{3, 4\}\}$	$T_{a,b}^{\text{out}} = \sum_c T_{c,c}^{\text{in}}$
$\{\{1\}, \{2, 4\}, \{3\}\}$	$T_{a,b}^{\text{out}} = \sum_c T_{c,b}^{\text{in}}$
$\{\{1\}, \{2, 3\}, \{4\}\}$	$T_{a,b}^{\text{out}} = \sum_c T_{b,c}^{\text{in}}$
$\{\{2\}, \{1, 4\}, \{3\}\}$	$T_{b,a}^{\text{out}} = \sum_c T_{c,b}^{\text{in}}$
$\{\{2\}, \{1, 3\}, \{4\}\}$	$T_{b,a}^{\text{out}} = \sum_c T_{b,c}^{\text{in}}$
$\{\{1, 2\}, \{3\}, \{4\}\}$	$T_{a,a}^{\text{out}} = \sum_{b,c} T_{b,c}^{\text{in}}$
$\{\{1\}, \{2, 3, 4\}\}$	$T_{a,b}^{\text{out}} = T_{b,b}^{\text{in}}$
$\{\{2\}, \{1, 3, 4\}\}$	$T_{b,a}^{\text{out}} = T_{b,b}^{\text{in}}$
$\{\{1, 2, 3\}, \{4\}\}$	$T_{a,a}^{\text{out}} = \sum_b T_{a,b}^{\text{in}}$
$\{\{1, 2, 4\}, \{3\}\}$	$T_{a,a}^{\text{out}} = \sum_b T_{b,a}^{\text{in}}$
$\{\{1, 2\}, \{3, 4\}\}$	$T_{a,a}^{\text{out}} = \sum_c T_{c,c}^{\text{in}}$
$\{\{1, 3\}, \{2, 4\}\}$	$T_{a,b}^{\text{out}} = T_{a,b}^{\text{in}}$
$\{\{1, 4\}, \{2, 3\}\}$	$T_{a,b}^{\text{out}} = T_{b,a}^{\text{in}}$
$\{\{1, 2, 3, 4\}\}$	$T_{a,a}^{\text{out}} = T_{a,a}^{\text{in}}$

Proposition (Maron et al.). *The space of linear maps $\phi: \mathbb{R}^{d^{k_1}} \rightarrow \mathbb{R}^{d^{k_2}}$ that is equivariant to permutations $\tau \in \mathbb{S}_d$ in the sense of (6) is spanned by a basis indexed by the partitions of the set $\{1, 2, \dots, k_1 + k_2\}$.*

Do we really want equivariance to \mathbb{S}_n ?



Automorphism groups



The automorphism group $\text{Aut}(\mathcal{G})$ of a graph is the subgroup of permutations that leave the adjacency matrix fixed:

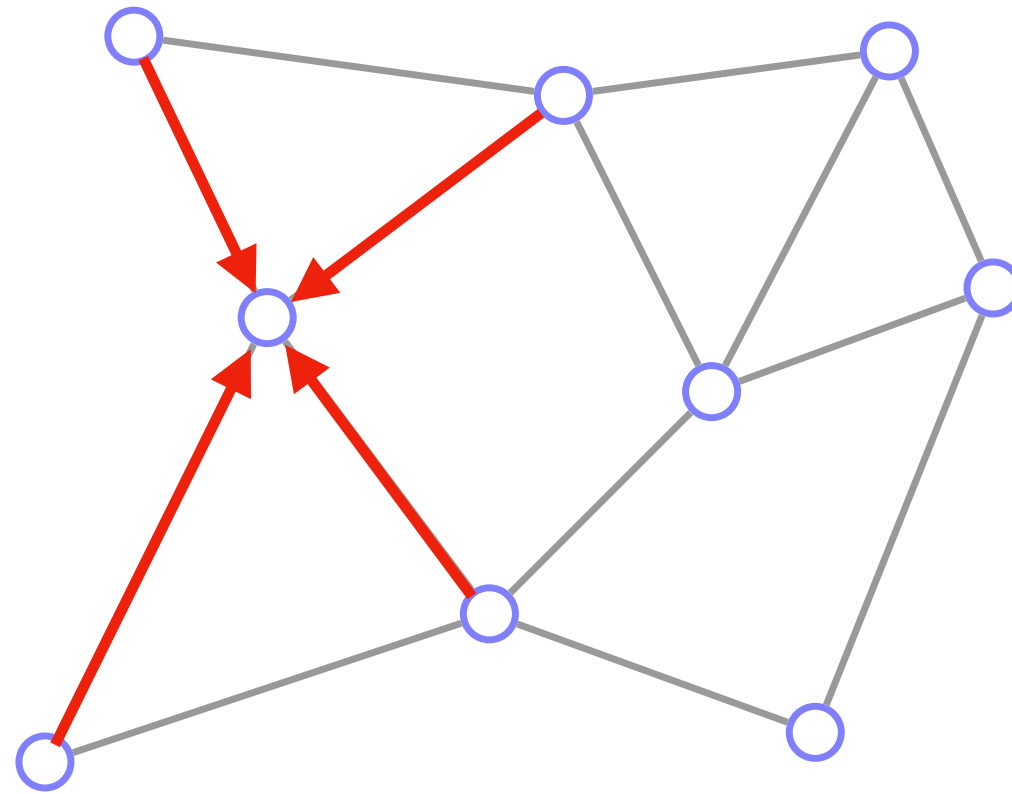
$$\sigma \circ A = A \quad \Longleftrightarrow \quad \sigma \in \text{Aut}(\mathcal{G})$$

What we really want is to be equivariant to just $\text{Aut}(\mathcal{G})$.

GNNs achieve this by sneakily using the adjacency matrix as **side information**.

2. Why I don't like GNNs

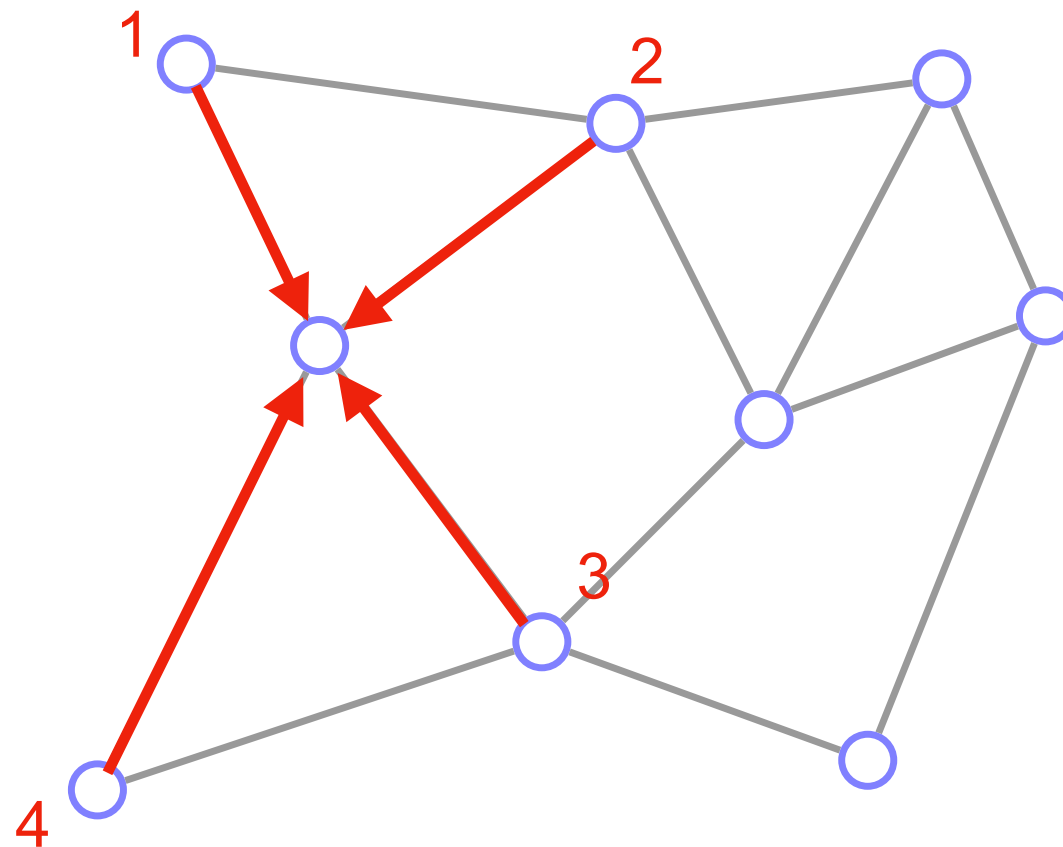
MPNN



$$f_i^{\ell+1} = \xi \left(W \sum_{j \in \mathcal{N}(i)} f_j^{\ell} + b \right)$$

Associative & commutative

1. MPNNs have amnesia



As soon as we sum the inputs, we lose the ability to distinguish what came from which neighbor.

2. MPNNs have no sense of global structure

arXiv:1312.6203v3 [cs.LG] 21 May 2014

Spectral Networks and Deep Locally Connected Networks on Graphs

Juan Bruna
New York University
bruna@cims.nyu.edu

Wojciech Zaremba
New York University
woj.zaremba@gmail.com

Arthur Szlam
The City College of New York
aszlam@ccny.cuny.edu

Yann LeCun
New York University
yann@cs.nyu.edu

Abstract

Convolutional Neural Networks are extremely efficient architectures in image and audio recognition tasks, thanks to their ability to exploit the local translational invariance of signal classes over their domains. In this paper we consider possible generalizations of CNNs to signals defined on more general domains without the action of a translation group. In particular, we propose two constructions, one based upon a hierarchical clustering of the domain, and another based on the spectrum of the graph Laplacian. We show through experiments that for low-dimensional graphs it is possible to learn convolutional layers with a number of parameters independent of the input size, resulting in efficient deep architectures.

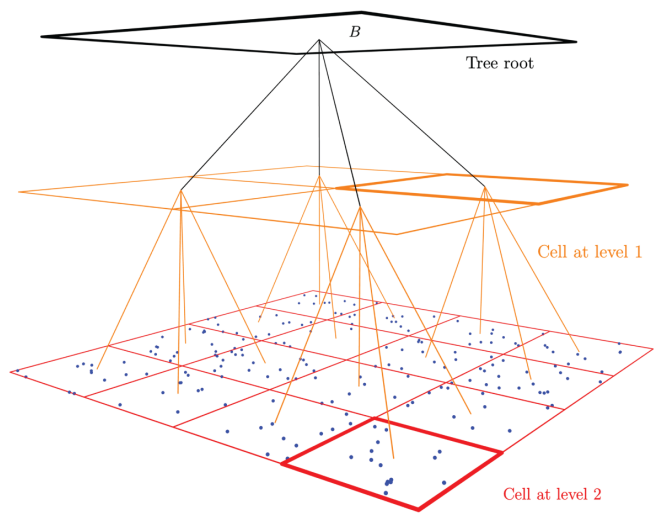
1 Introduction

Convolutional Neural Networks (CNNs) have been extremely successful in machine learning problems where the coordinates of the underlying data representation have a grid structure (in 1, 2 and 3 dimensions), and the data to be studied in those coordinates has translational equivariance/invariance with respect to this grid. Speech [11], images [14, 20, 22] or video [23, 18] are prominent examples that fall into this category.

On a regular grid, a CNN is able to exploit several structures that play nicely together to greatly reduce the number of parameters in the system:

1. The translation structure, allowing the use of filters instead of generic linear maps and hence weight sharing.
2. The metric on the grid, allowing compactly supported filters, whose support is typically much smaller than the size of the input signals.
3. The multiscale dyadic clustering of the grid, allowing subsampling, implemented through stride convolutions and pooling.

If there are n input coordinates on a grid in d dimensions, a fully connected layer with m outputs requires $n \cdot m$ parameters, which in typical operating regimes amounts to a complexity of $O(n^2)$ parameters. Using arbitrary filters instead of generic fully connected layers reduces the complexity to $O(n)$ parameters per feature map, as does using the metric structure by building a "locally connected" net [8, 17]. Using the two together gives $O(k \cdot S)$ parameters, where k is the number of feature maps and S is the support of the filters, and as a result the learning complexity is independent of n . Finally, using the multiscale dyadic clustering allows each successive layer to use a factor of 2^d less (spatial) coordinates per filter.



Multiscale

Neural Message Passing for Quantum Chemistry

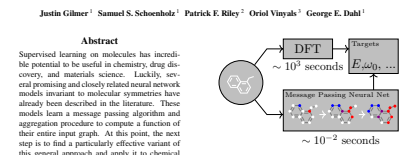


Figure 1. A Message Passing Neural Network predicts quantum properties of an organic molecule by modeling a computationally expensive DFT calculation.

Abstract

Supervised learning on molecules has incredible potential to be useful in chemistry, drug discovery, and materials science. Luckily, several promising and closely related neural network models invariant to molecular symmetries have already been described in the literature. These models learn a message passing algorithm and aggregation procedure to compute a function of their entire input graph. At this point, the next step is to find a particularly effective variant of this general approach and apply it to chemical prediction benchmarks until we either solve them or reach the limits of the approach. In this paper, we reformulate existing models into a single common framework we call Message Passing Neural Networks (MPNNs) and explore additional novel variations within this framework. Using MPNNs we demonstrate state of the art results on an important molecular property prediction benchmark; these results are strong enough that we believe future work should focus on datasets with larger molecules or more accurate ground truth labels.

1. Introduction

The past decade has seen remarkable success in the use of deep neural networks to understand and translate natural language [Wu et al., 2016], generate and decode complex audio signals [Hinton et al., 2012], and infer features from real-world images and videos [Krizhevsky et al., 2012]. Although chemists have applied machine learning to many problems over the years, predicting the properties of molecules and materials using machine learning (and especially deep learning) is still in its infancy. To date, most research applying machine learning to chemistry tasks [Hansen et al., 2015; Huang & von Lilienfeld, 2016; Rupp et al., 2012; Rogers & Hahn, 2010; Montavon et al., 2012; Behler & Parrinello, 2007; Schoenholz et al., 2016] has revolved around feature engineering. While neural networks have been applied in a variety of situations [Merkwirth & Langmuir, 2005; Micheli, 2009; Lasci et al., 2013; Duvenaud et al., 2015], they have yet to become widely adopted. This situation is reminiscent of the state of image models before the broad adoption of convolutional neural networks and is due, in part, to a dearth of empirical evidence that neural architectures with the appropriate inductive bias can be successful in this domain.

Recently, large scale quantum chemistry calculation and molecular dynamics simulations coupled with advances in high throughput experiments have begun to generate data at an unprecedented rate. Most classical techniques do not make effective use of the larger amounts of data that are now available. The time is ripe to apply more powerful and flexible machine learning methods to these problems, assuming we can find models with suitable inductive biases. The symmetries of atomic systems suggest neural networks that operate on graph structured data and are invariant to graph isomorphism might also be appropriate for molecules. Sufficiently successful models could someday help automate challenging chemical search problems in drug discovery or materials science.

In this paper, our goal is to demonstrate effective machine learning models for chemical prediction problems

$$x *_G g := U^T(\text{diag}(w_g)Ux) .$$

$$x_{k+1,j} = h \left(V \sum_{i=1}^{f_k-1} F_{k,i,j} V^T x_{k,i} \right) \quad (j = 1 \dots f_k) ,$$

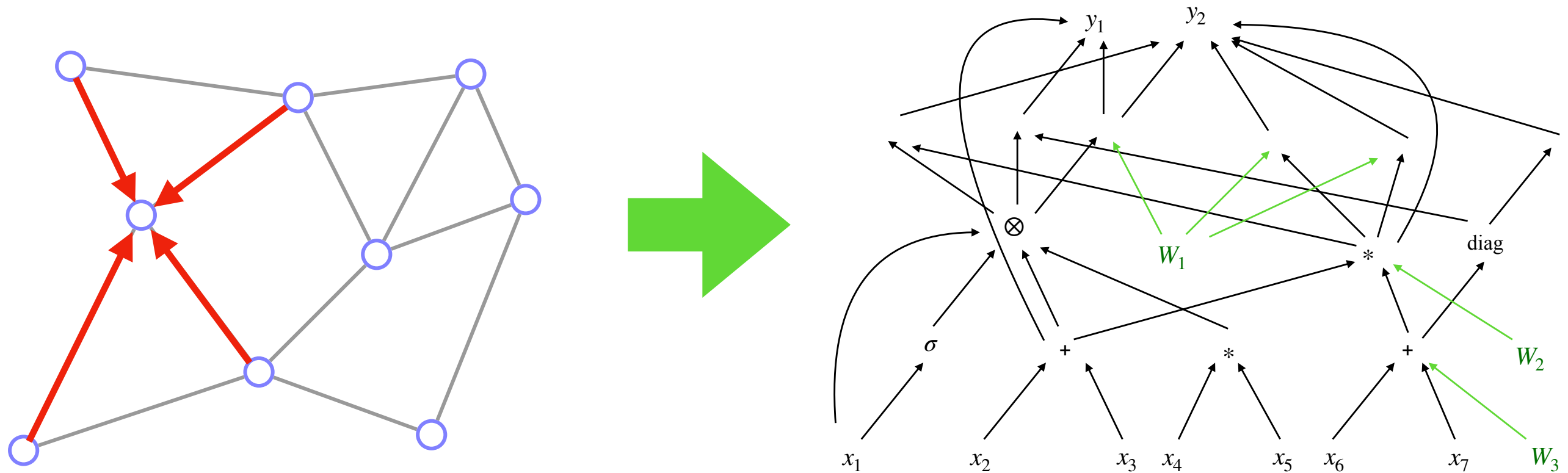
$$f_i^{\ell+1} = \xi \left(W \sum_{j \in \mathcal{N}(i)} f_j^{\ell} + b \right)$$



Global

Local

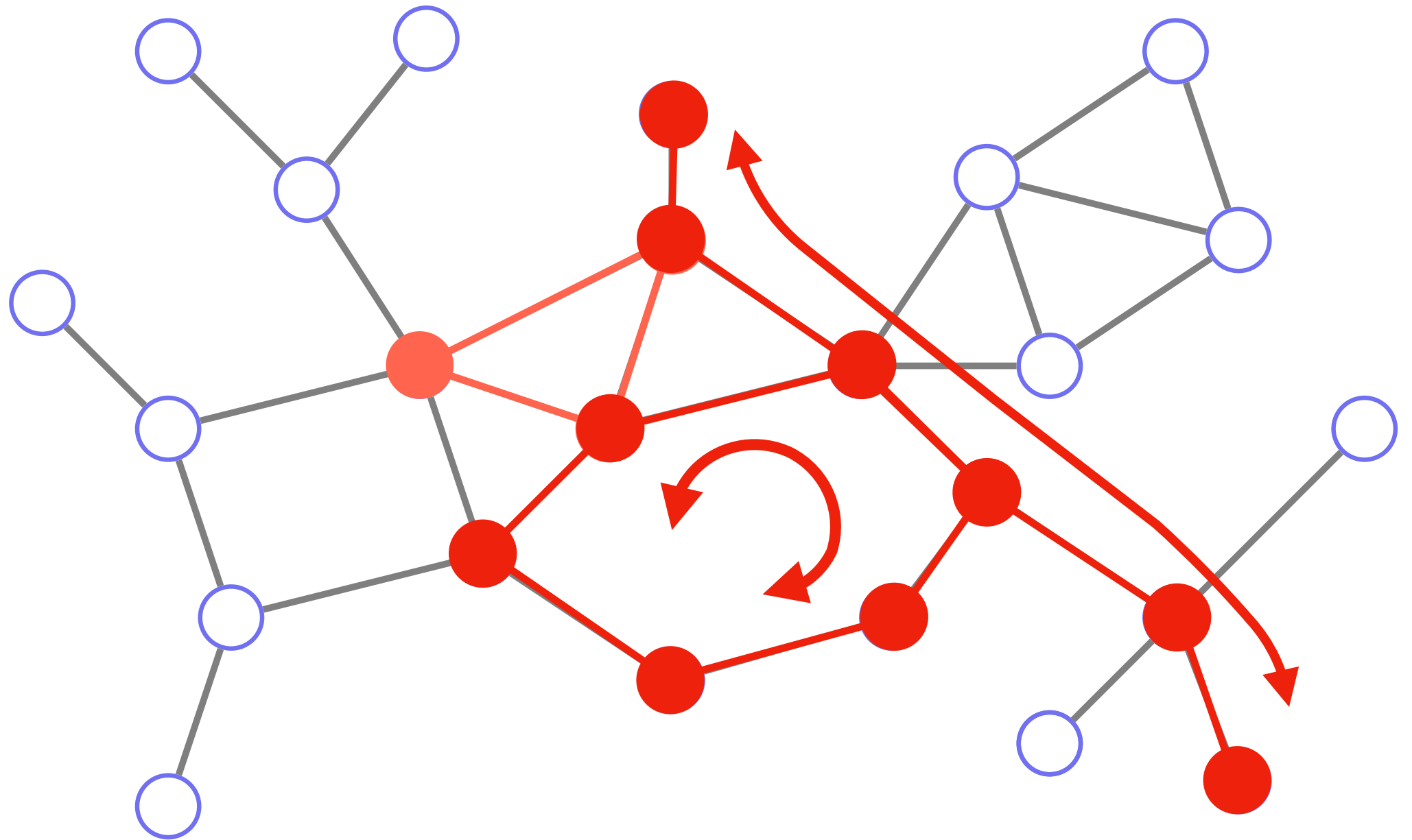
3. MPNNs only encode topology implicitly



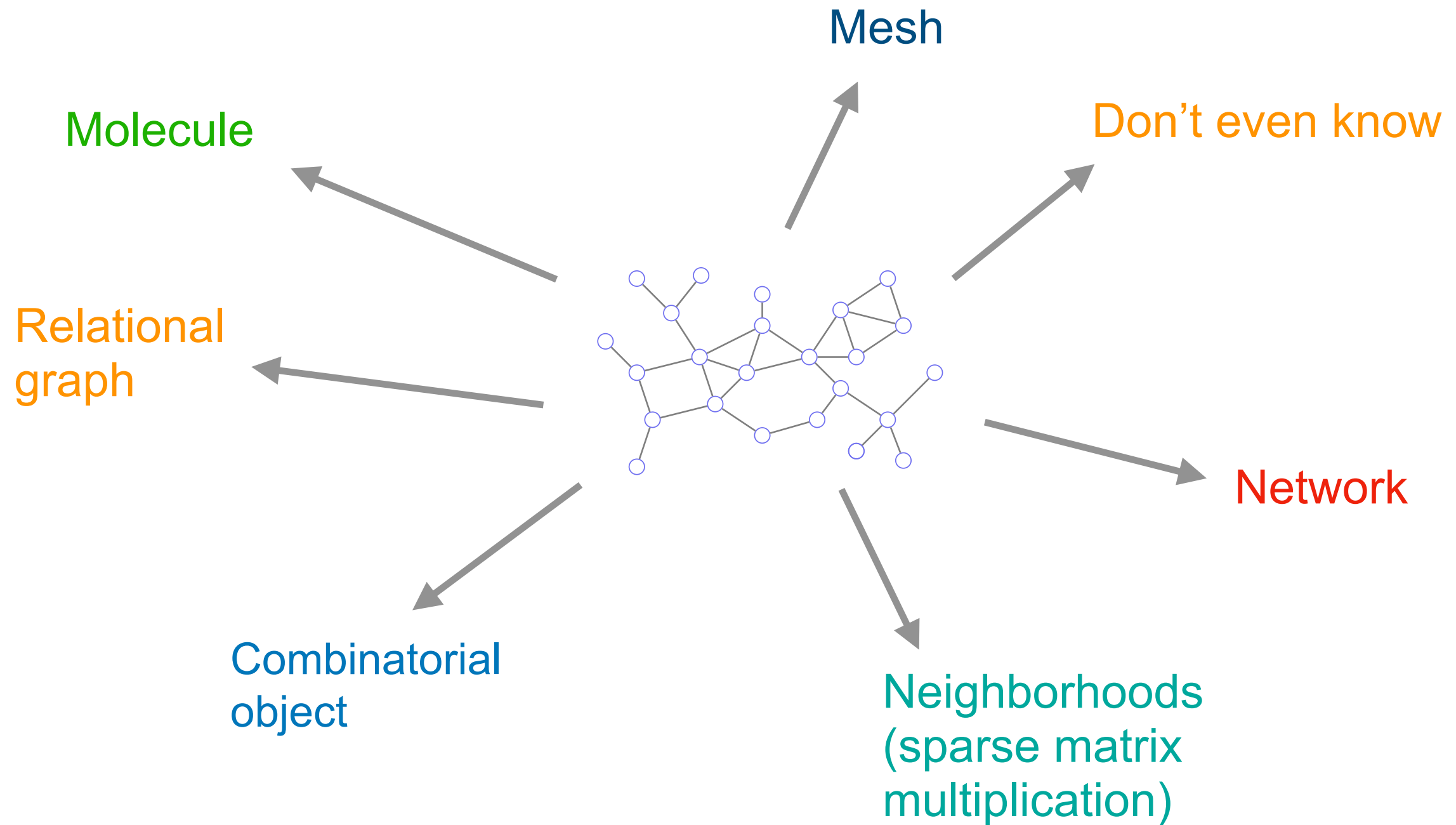
The graph topology determines the structure of the compute graph.

There is no strong sense in which we can encode known substructures, e.g., functional groups.

4. MPNNs don't reduce to classical convolution

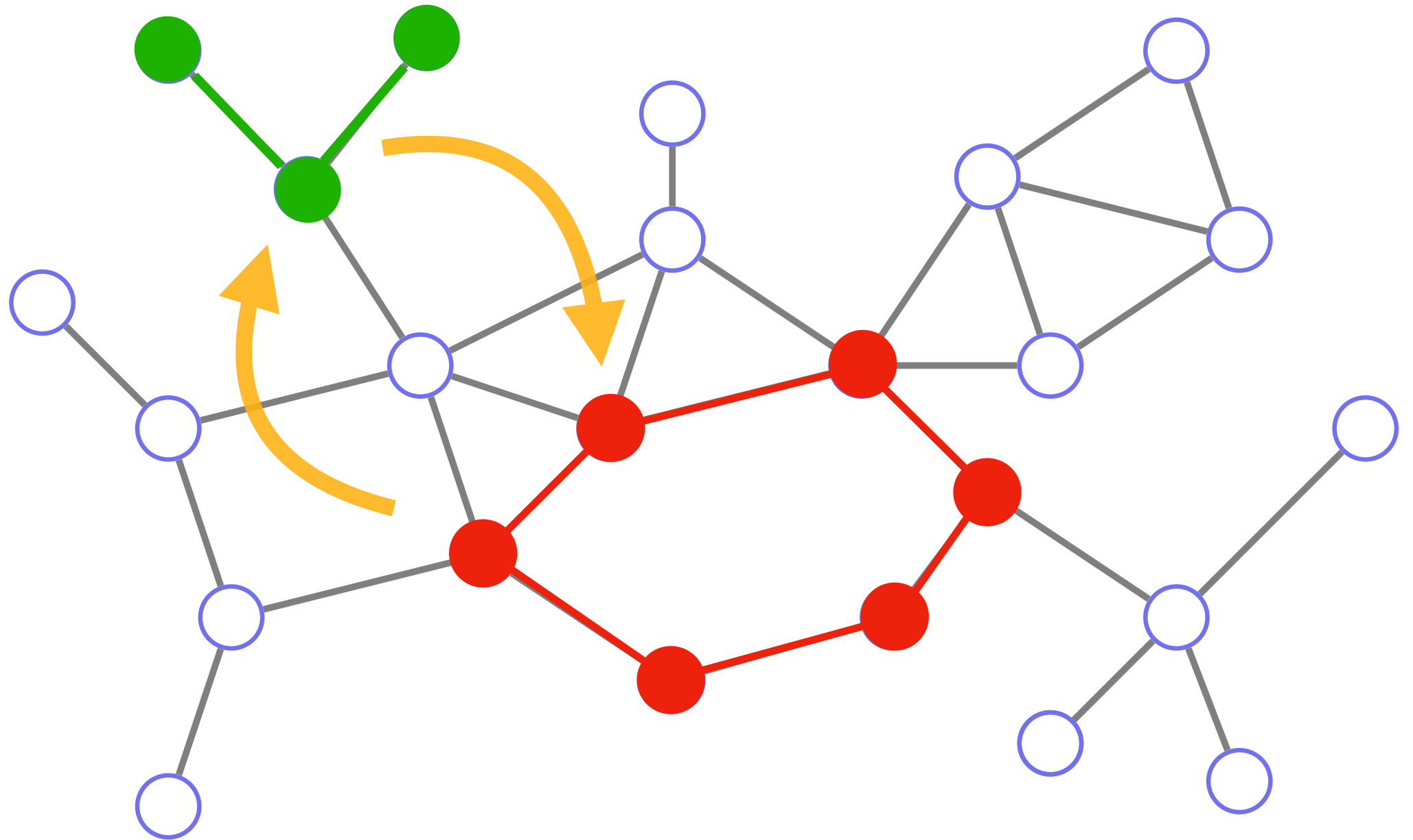


5. What is a graph, anyway?



Higher order message passing

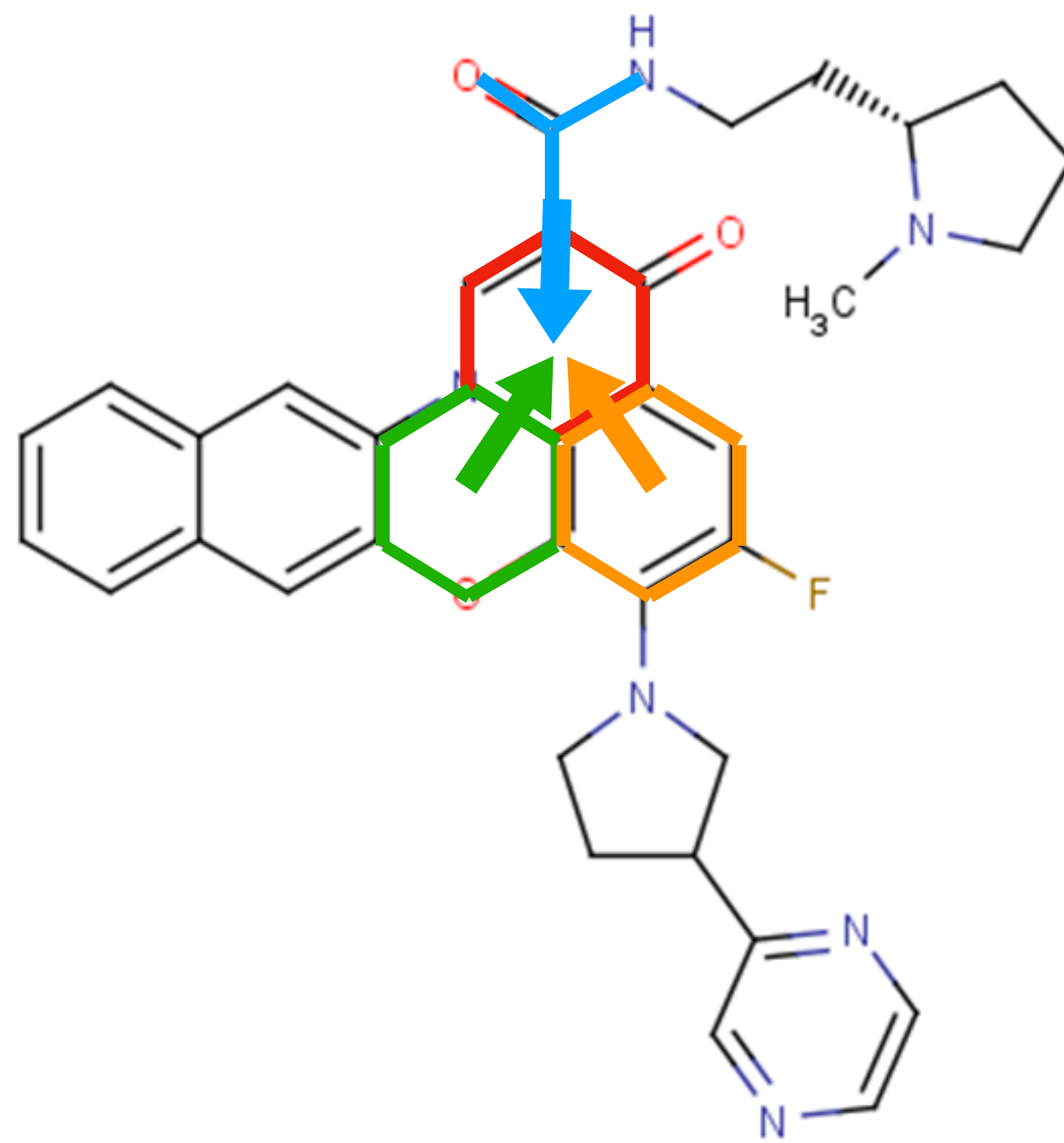
Subgraph neural nets



Define some “policy” to identify interesting subgraphs and use specialized rules to send messages to/from these subgraphs.

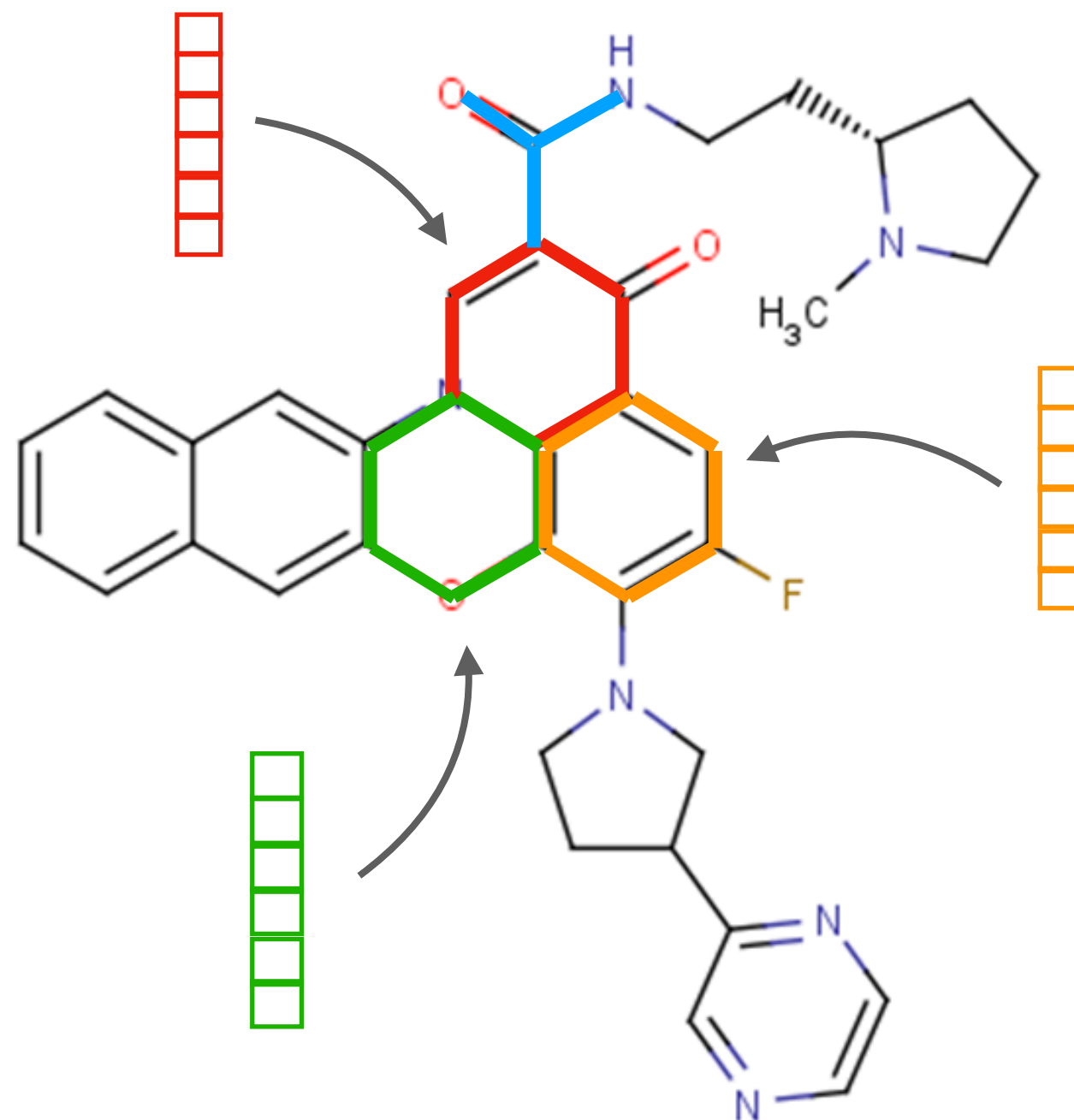
How can we do this in an equivariant way?

[Frasca, Bevilacqua, Bronstein & Maron, 2022]

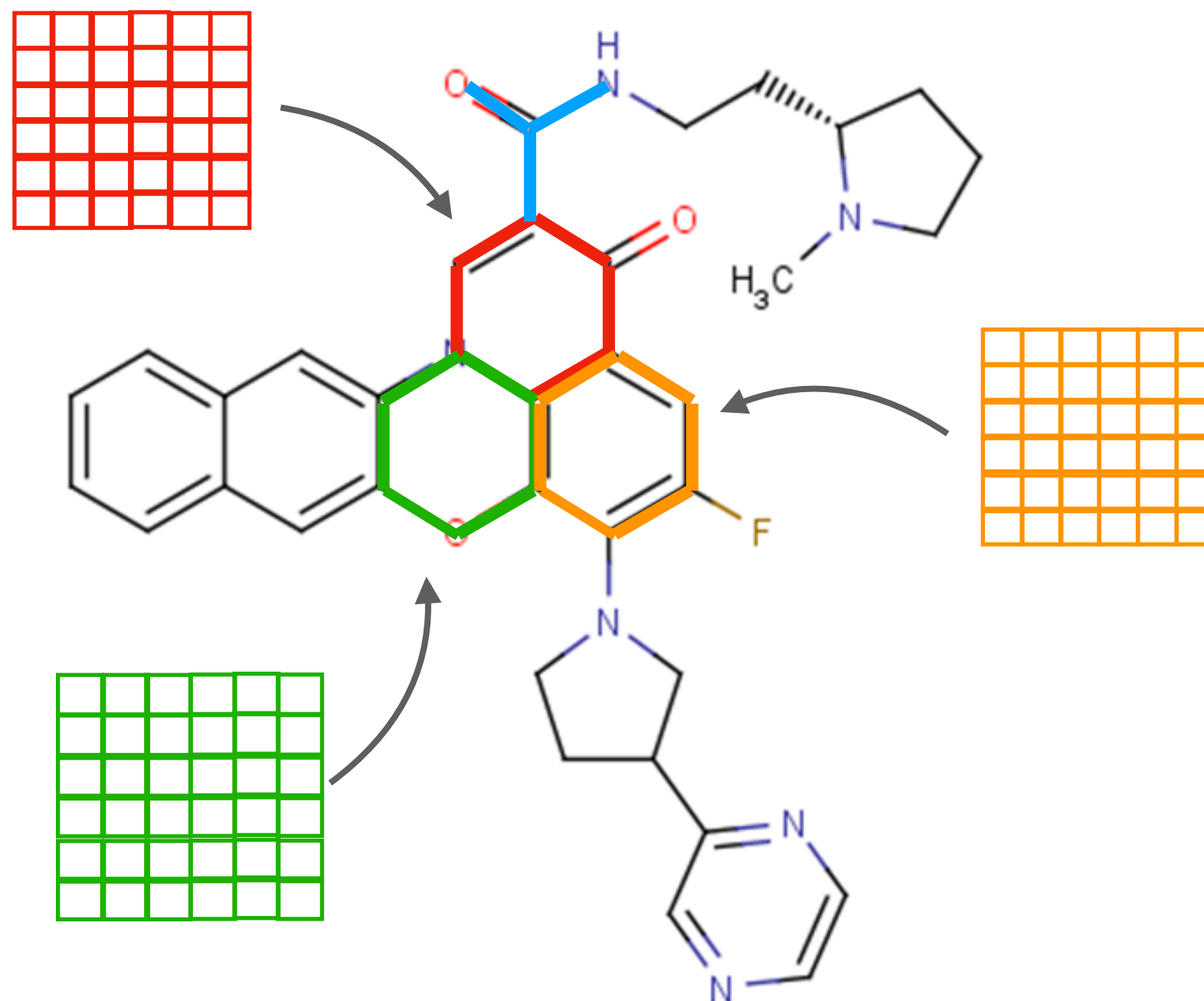


quarfloxin

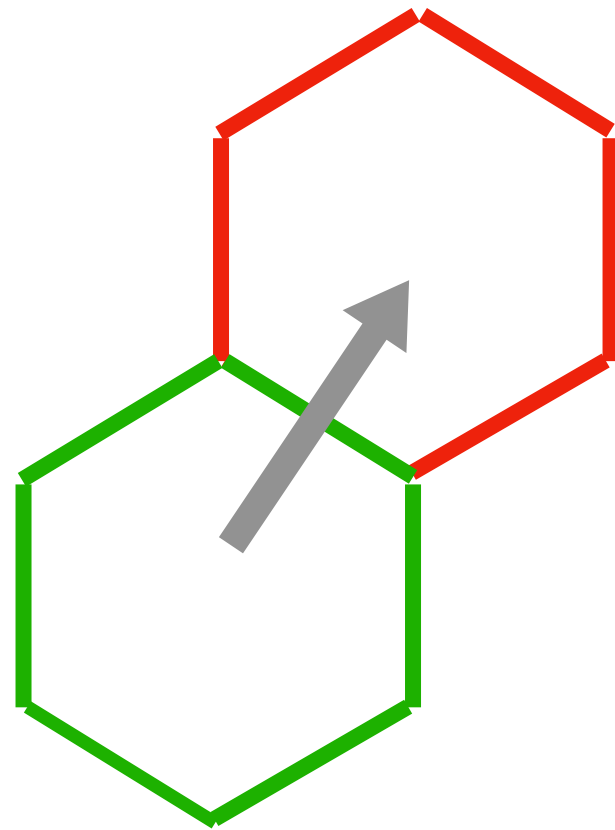
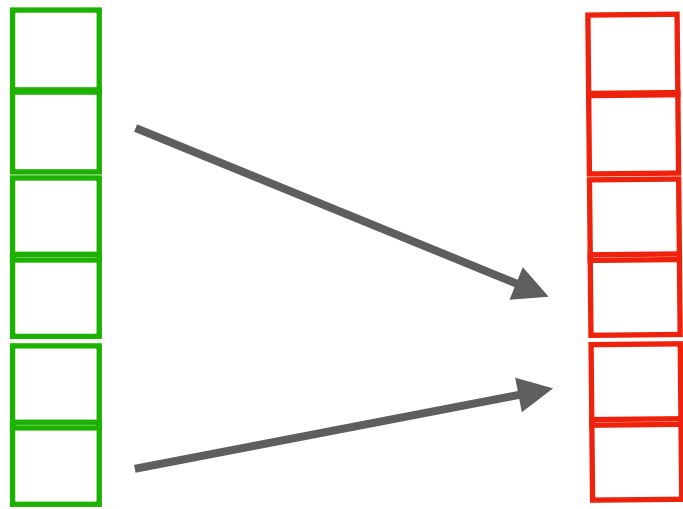
First order subgraph neural nets



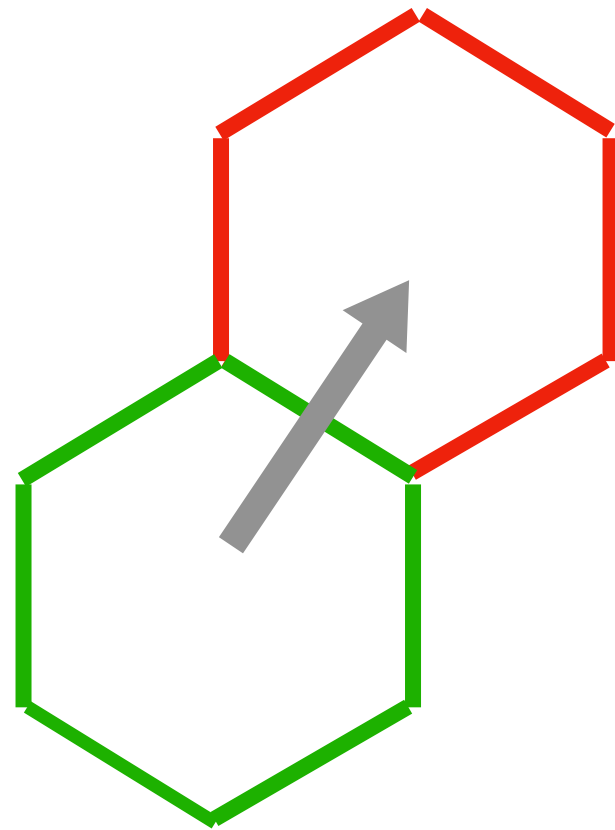
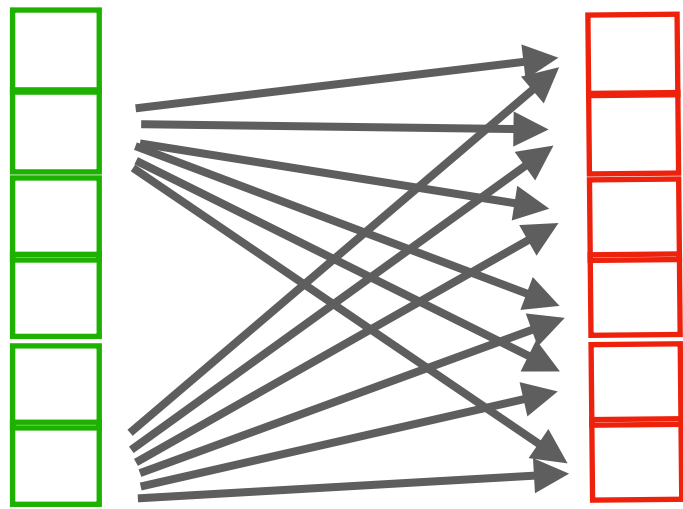
Second order subgraph neural nets



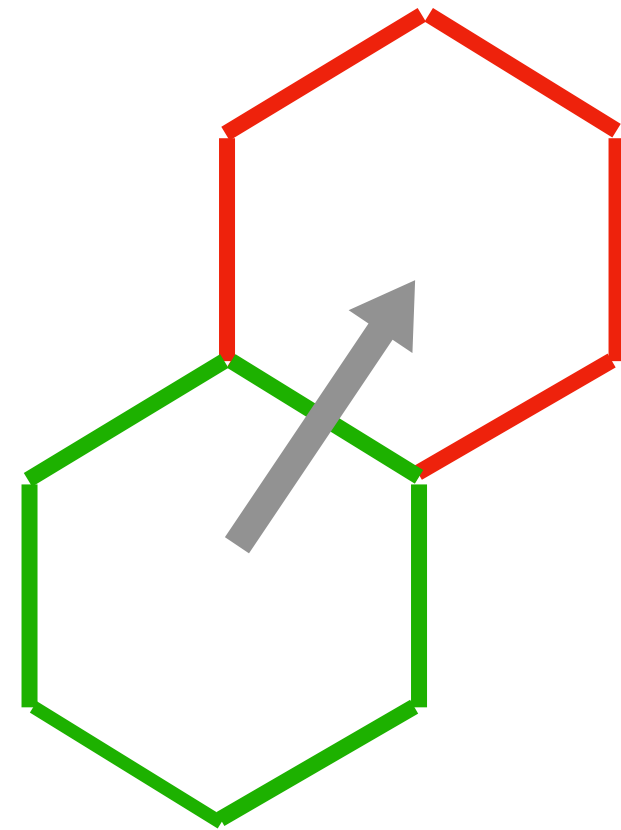
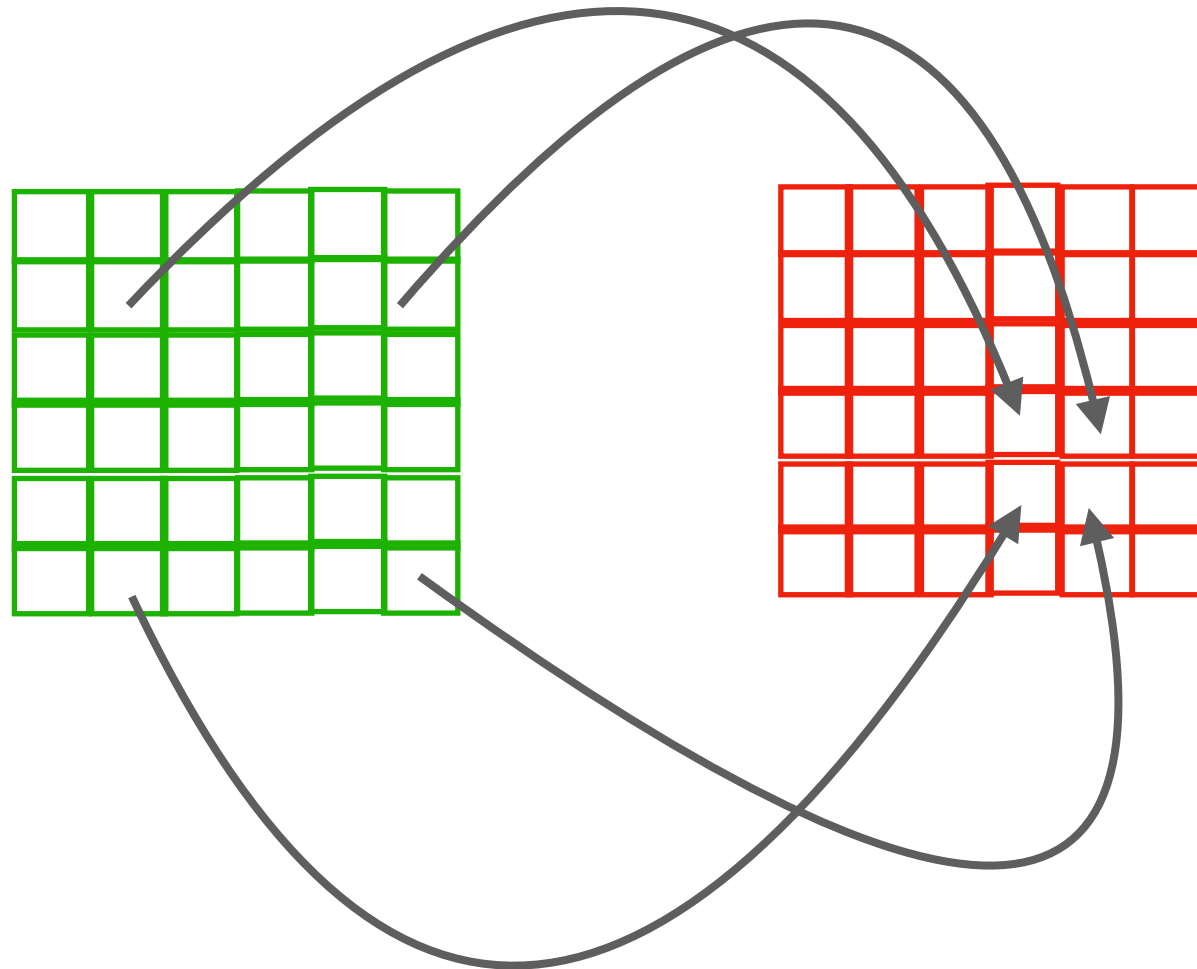
What is the correct generalization of message passing?

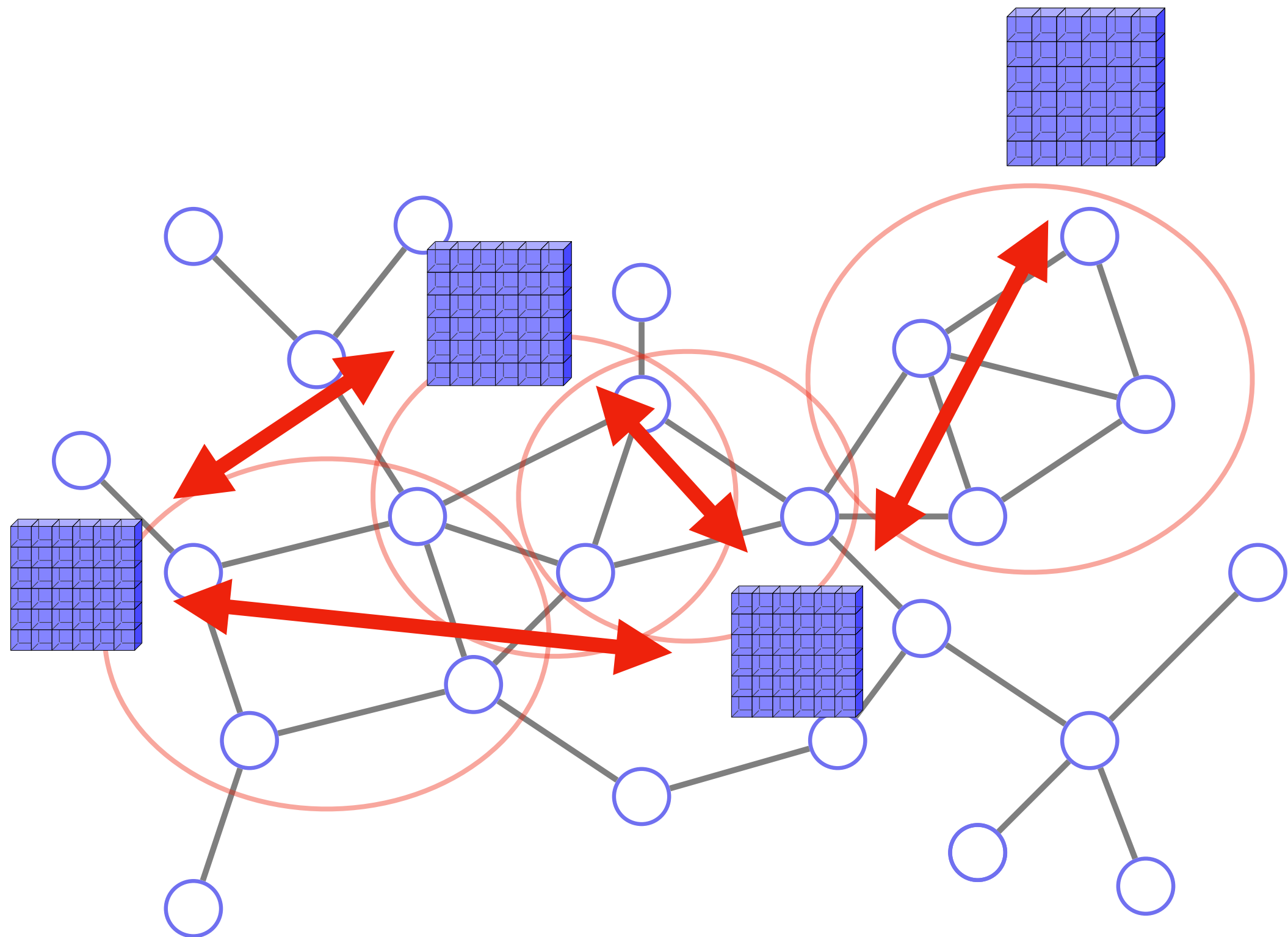


What is the correct generalization of message passing?



What is the correct generalization of message passing?

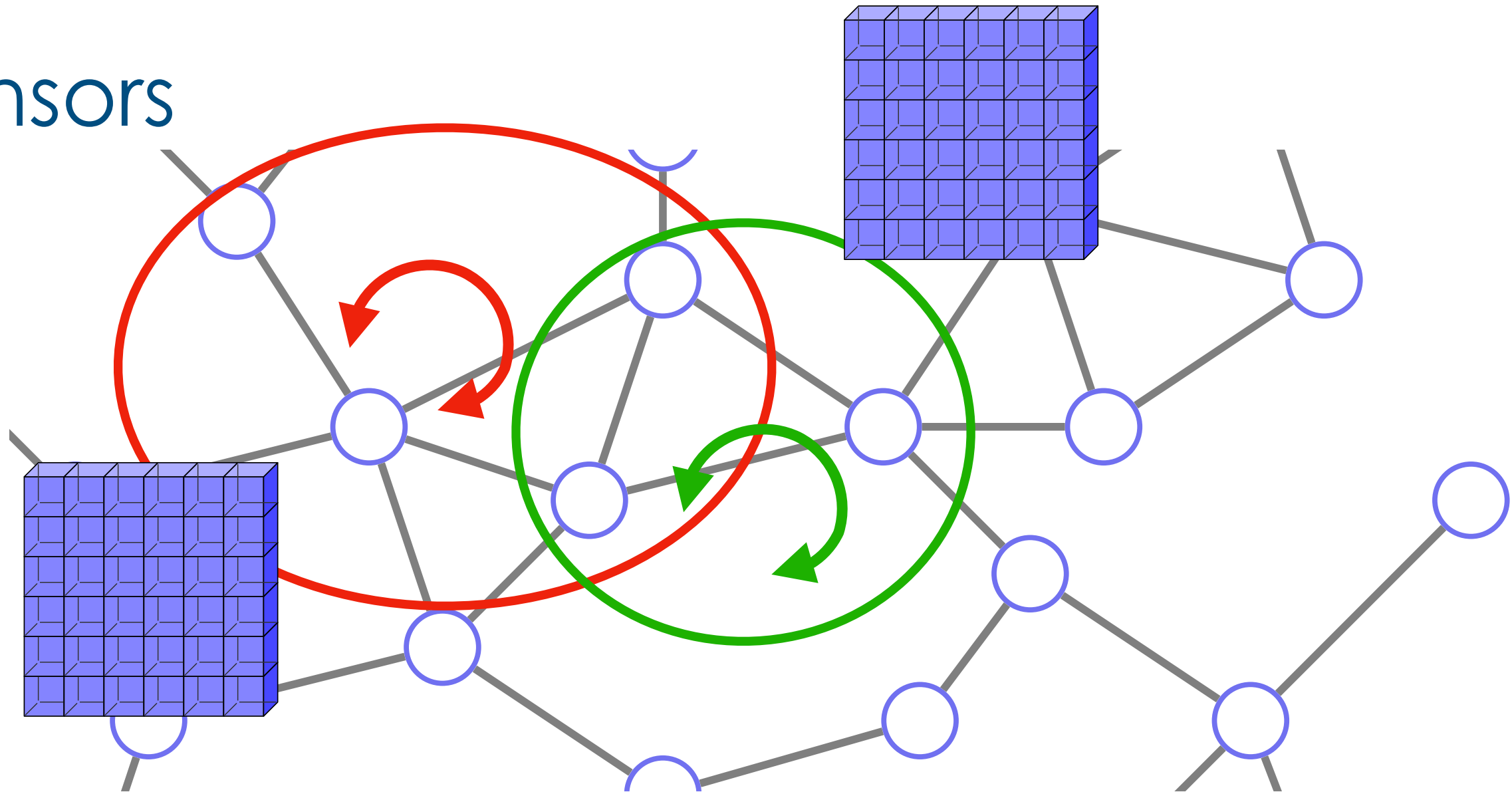




P-tensors

[Andrew Hands, Tiny Sun & K, 2024]

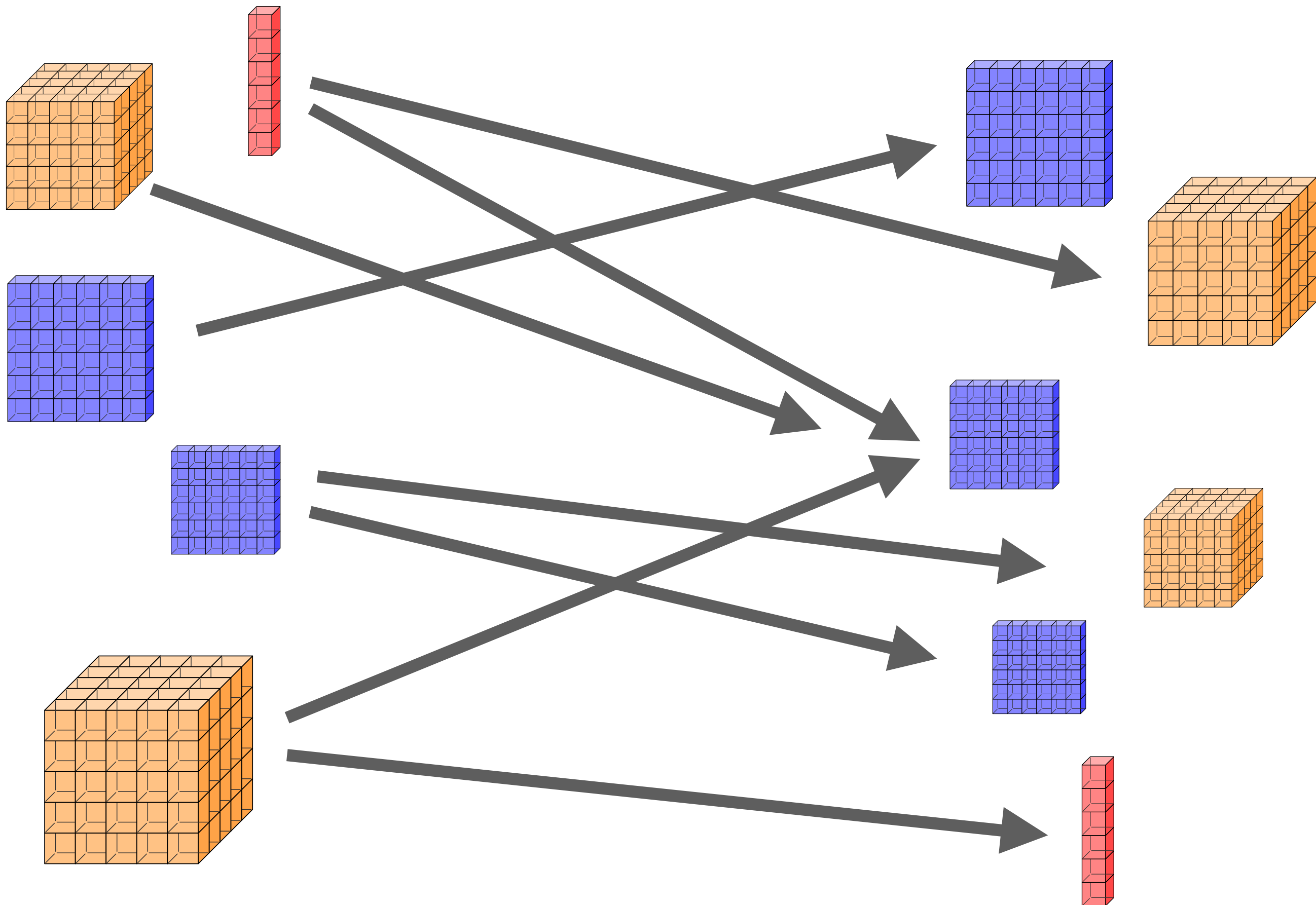
P-tensors



Given an ordered subset of d atoms (x_1, x_2, \dots, x_m) , a tensor $T \in \mathbb{R}^{d \times d \times \dots \times d}$ that transforms under permutations according to

$$T_{i_1, \dots, i_k} = T_{\tau^{-1}(i_1), \dots, \tau^{-1}(i_k)} \quad \tau \in \mathbb{S}_k$$

is called a k 'th order **P-tensor**.

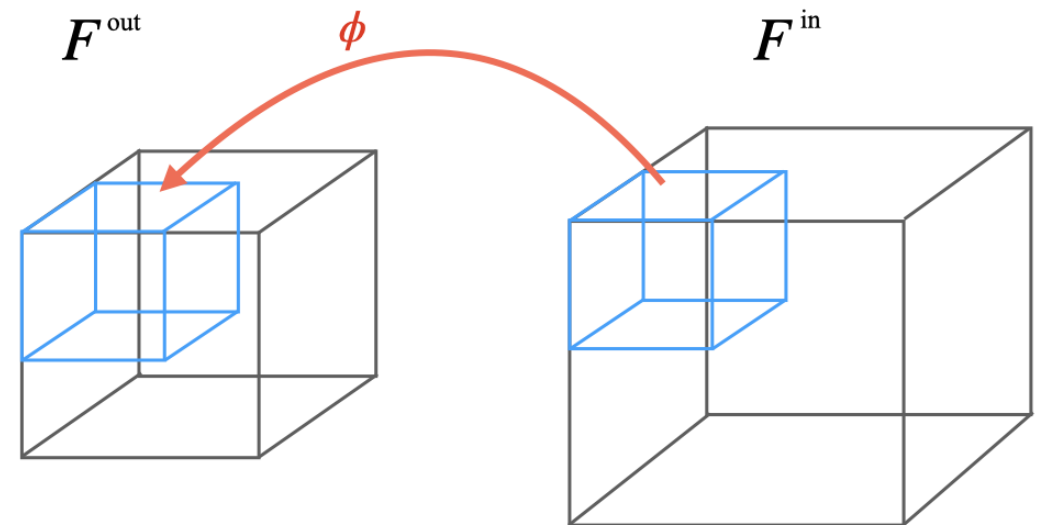


Message passing between P-tensors

Definition 6 (Equivariant map between P-tensors). Let T_1 and T_2 be two P-tensors with reference domains $\mathcal{D}_1 \subseteq \mathcal{U}$ and $\mathcal{D}_2 \subseteq \mathcal{U}$, respectively. We say that a linear map $\phi: T_1 \rightarrow T_2$ is permutation equivariant if

$$\phi(\sigma \downarrow_{\mathcal{D}_1} \circ T_1) = \sigma \downarrow_{\mathcal{D}_2} \circ \phi(T_1)$$

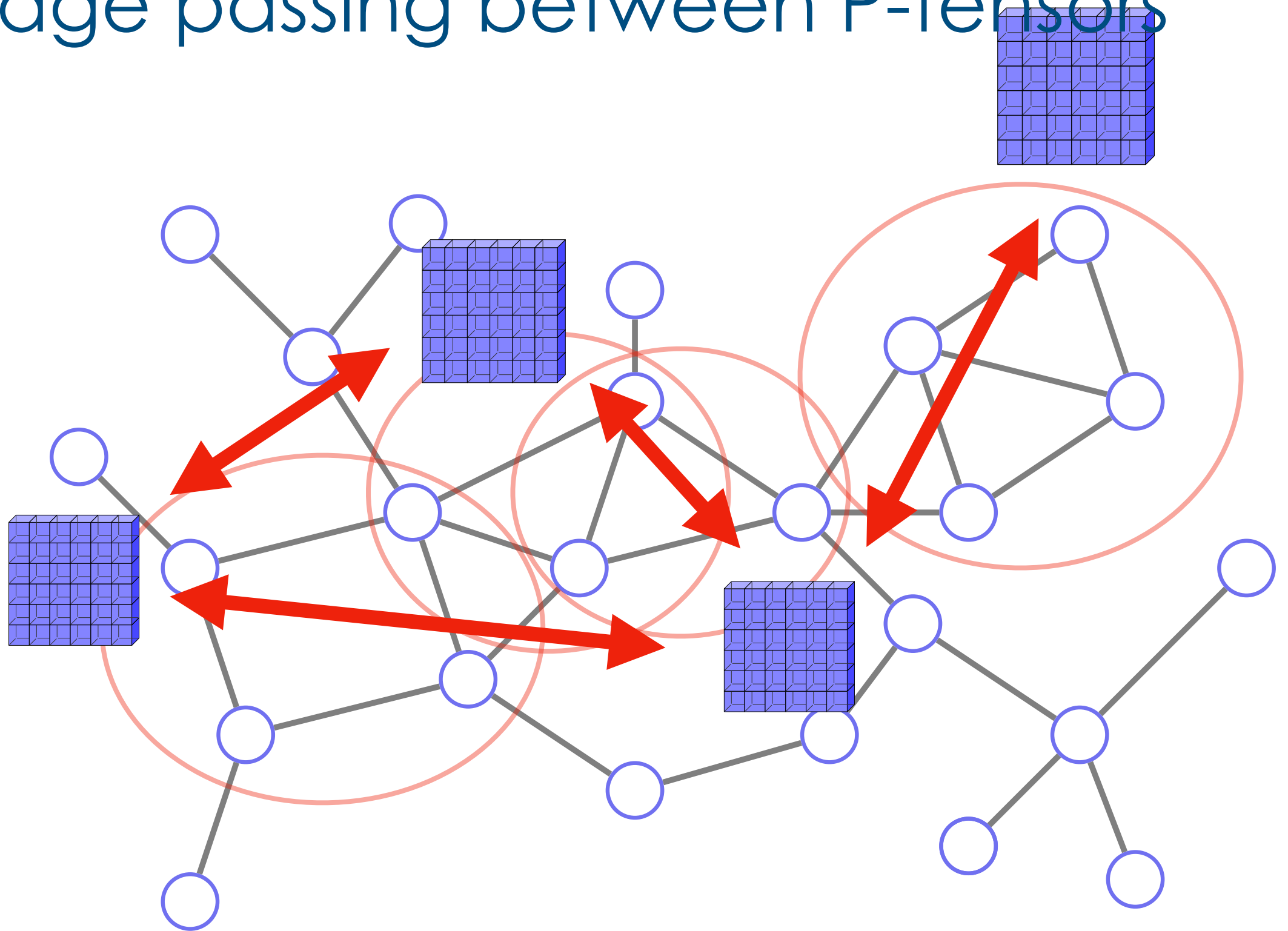
for any permutations σ of \mathcal{U} that fixes both \mathcal{D}_1 and \mathcal{D}_2 as sets.



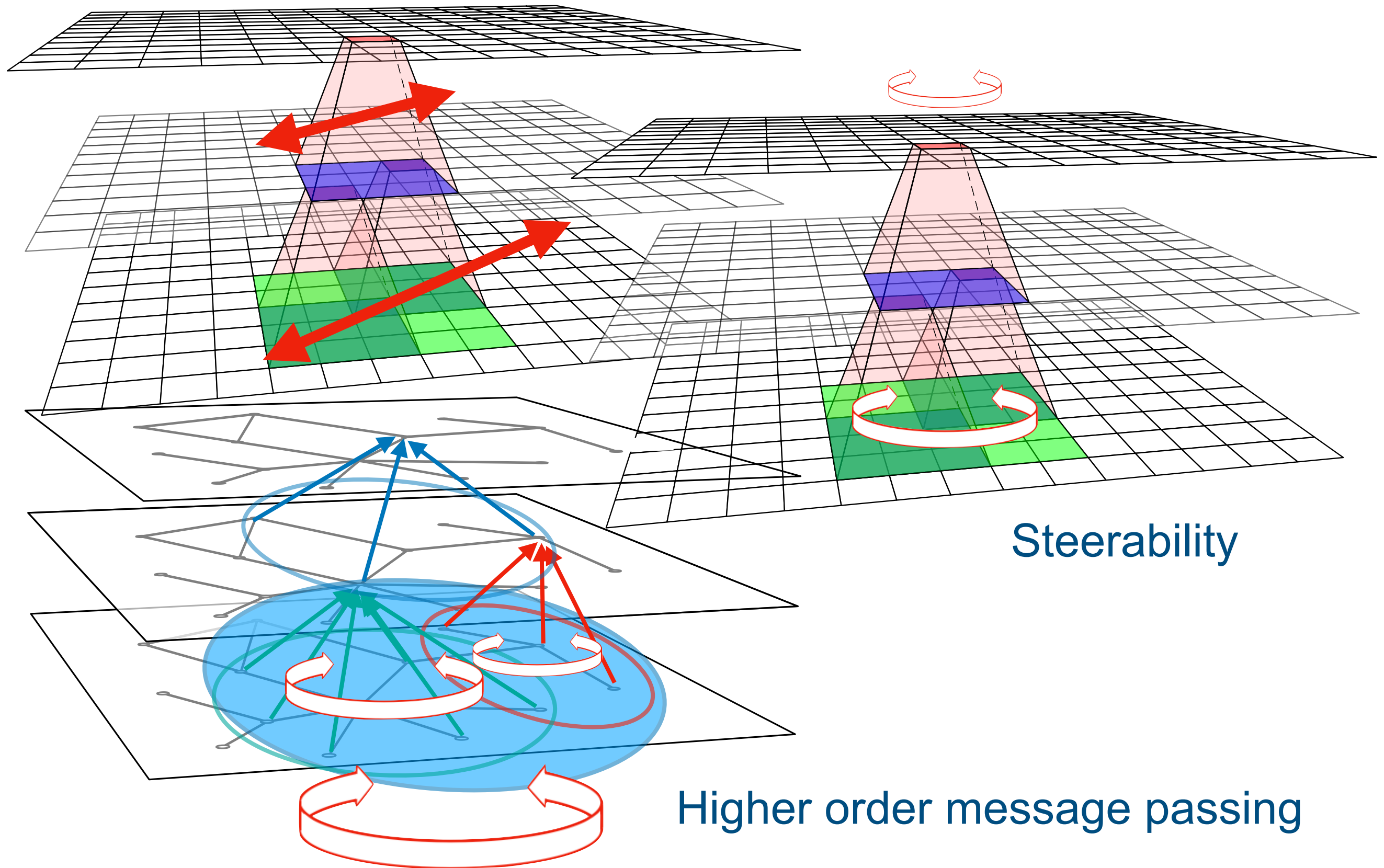
Theorem 2. Let T_1 and T_2 be two P-tensors with reference domains \mathcal{D}_1 and \mathcal{D}_2 such that $\mathcal{D}_1 \cap \mathcal{D}_2 \neq \emptyset$ and $\mathcal{D}_1 \not\subseteq \mathcal{D}_2$ and $\mathcal{D}_2 \not\subseteq \mathcal{D}_1$. Then for each partition \mathcal{P} of $\{1, \dots, k_1 + k_2\}$ of type (p_1, p_2, p_3) there are $2^{p_1 + p_3}$ independent permutation equivariant maps $\phi: T_1 \mapsto T_2$.

(k_1, k_2)	# of maps in $\mathcal{D}_1 = \mathcal{D}_2$ case	# of maps in $\mathcal{D}_1 \neq \mathcal{D}_2$ case
$(0, 0)$	1	1
$(1, 1)$	2	5
$(1, 2)$	5	17
$(2, 2)$	15	61
$(2, 3)$	52	321
$(3, 3)$	203	769

Message passing between P-tensors



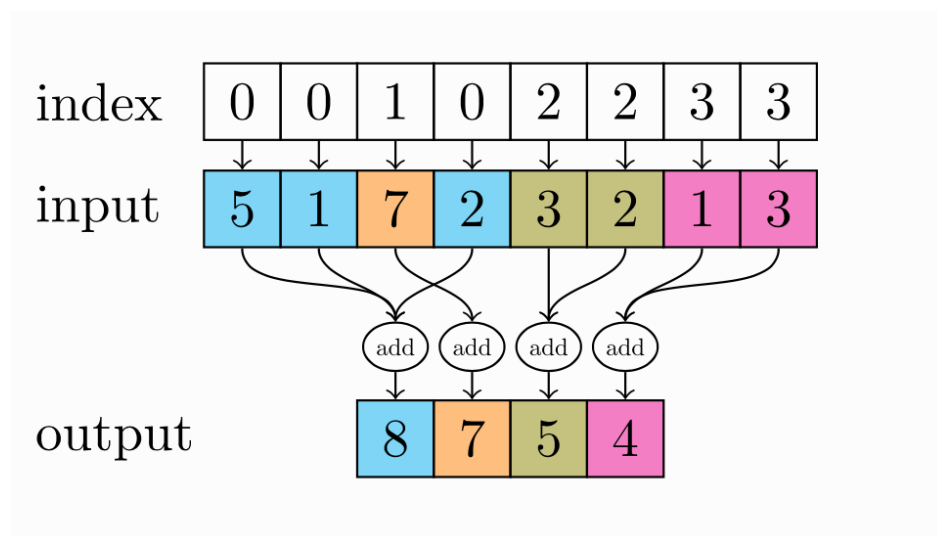
Convolution



P-tensor software

pytorch-geometric

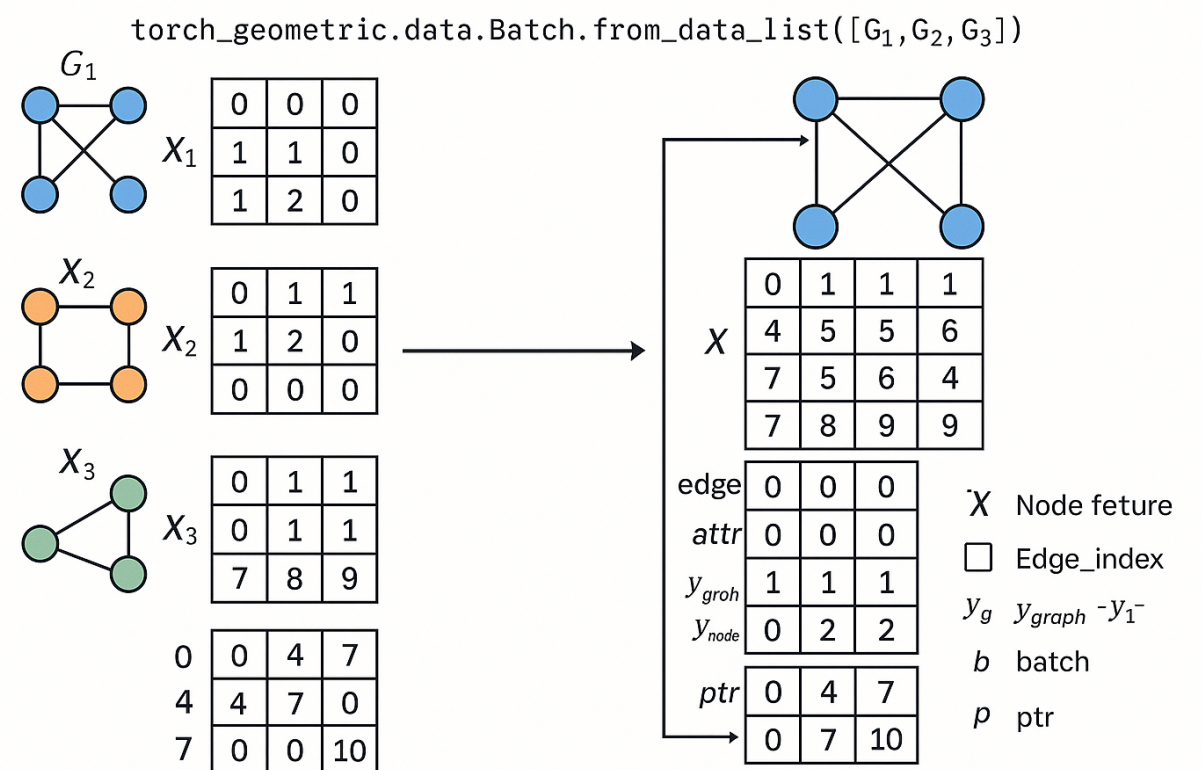
Efficient gather/scatter



Batching

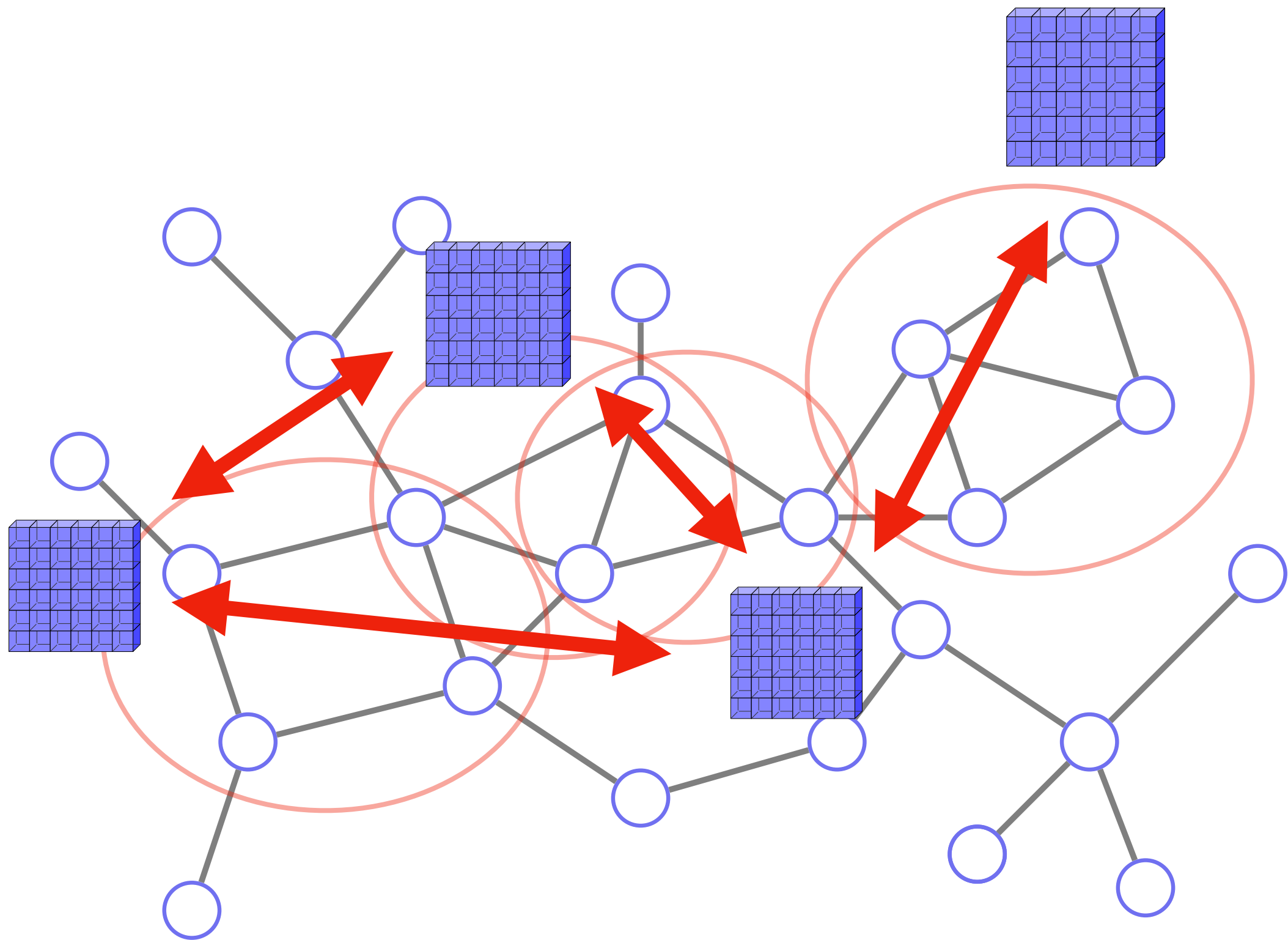
PyTorch Geometric Batching

(Data \rightarrow Batch)

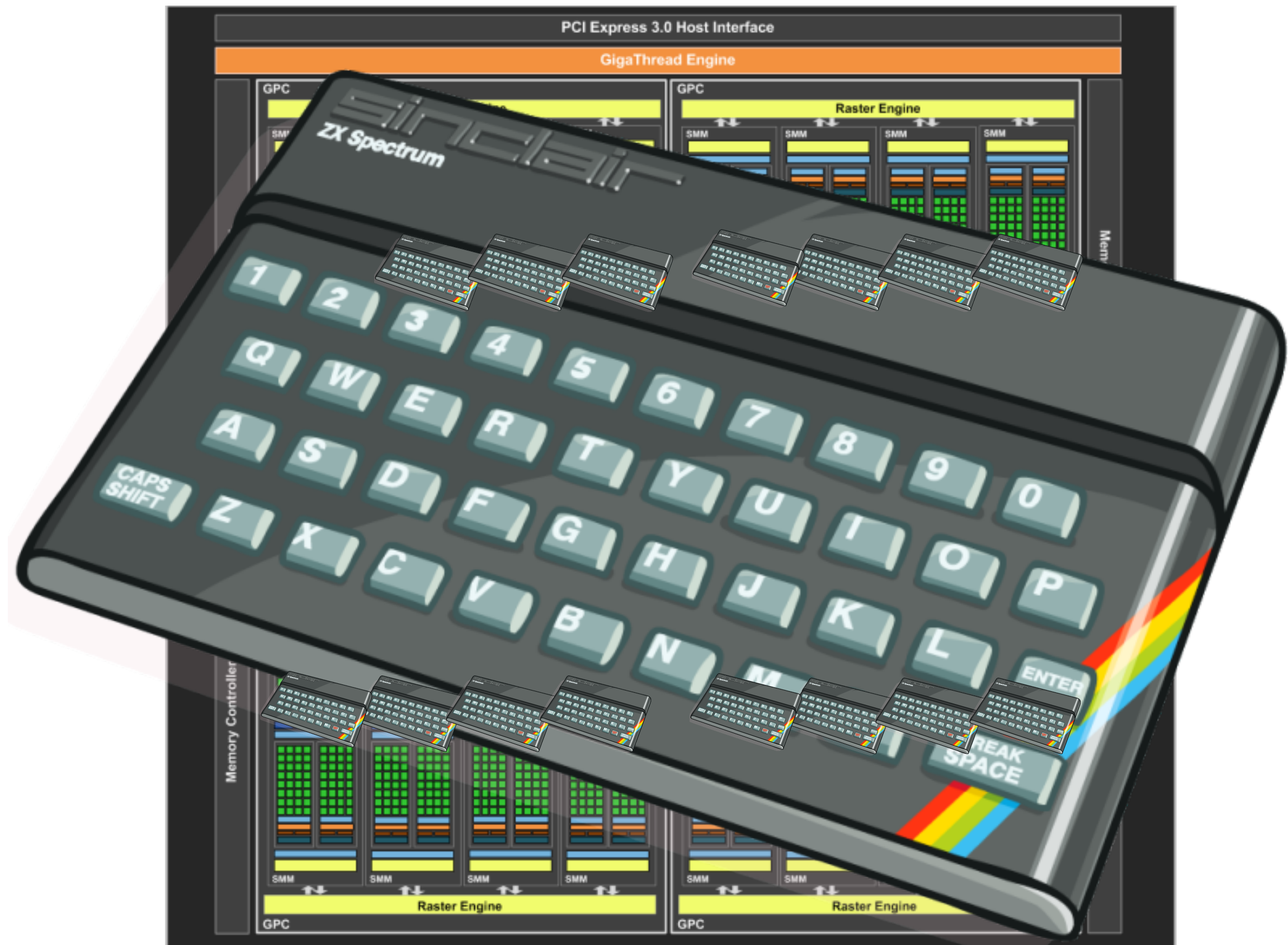


pytorch scatter

```
2
3  #define ATOMIC(NAME) \
4      template <typename scalar, size_t size> struct Atomic##NAME##IntegerImpl; \
5      \
6      template <typename scalar> struct Atomic##NAME##IntegerImpl<scalar, 1> { \
7          inline __device__ void operator()(scalar *address, scalar val) { \
8              uint32_t *address_as_ui = (uint32_t *) (address - ((size_t) address & 3)); \
9              uint32_t old = *address_as_ui; \
10             uint32_t shift = ((size_t) address & 3) * 8; \
11             uint32_t sum; \
12             uint32_t assumed; \
13             \
14             do { \
15                 assumed = old; \
16                 sum = OP(val, scalar((old >> shift) & 0xff)); \
17                 old = (old & ~(0x000000ff << shift)) | (sum << shift); \
18                 old = atomicCAS(address_as_ui, assumed, old); \
19             } while (assumed != old); \
20         } \
21     }; \
22
```







ref domains



communication graph



```
>>> A=ptens.ptensors1.randn([[1,2],[3]],3)
>>> G=ptens.graph.from_matrix(torch.ones(3,2))
>>> print(A)
```

```
Ptensor1 [1,2]:
[ -1.23974 -0.407472  1.61201 ]
[  0.399771  1.3828  0.0523187 ]
```

```
Ptensor1 [3]:
[ -0.904146  1.87065 -1.66043 ]
```

ref. domains



```
>>> B=ptens.transfer1(A,[[1],[2,3],[1,3]],G)
>>> print(B)
```

```
Ptensor1 [1]:
[ -1.23974 -0.407472  1.61201 -1.23974 -0.407472  1.61201 ]
```

```
Ptensor1 [2,3]:
[  0.399771  1.3828  0.0523187  0.399771  1.3828  0.0523187 ]
[ -0.904146  1.87065 -1.66043 -0.904146  1.87065 -1.66043 ]
```

https://github.com/arhands/topological_model

<https://github.com/risi-kondor/ptens>

```
x=subgraphlayer0(G,x_in)

a=p.subgraphlayer1.gather(x,self.nodes)
a=self.linear(a,w0,b0)

b=p.subgraphlayer1.gather(x,self.edges)
b=self.linear(b,w1,b1)

c=p.subgraphlayer1.gather(x,self.cycle5)
b=self.linear(c,w2,b2)

d=p.subgraphlayer1.gather(x,self.cycle6)
d=self.linear(d,w3,b3)

z=sugraphlayer1.cat(a,b,c,d)
z=ReLU(z)

y=subgraphlayer2(z,S)
```

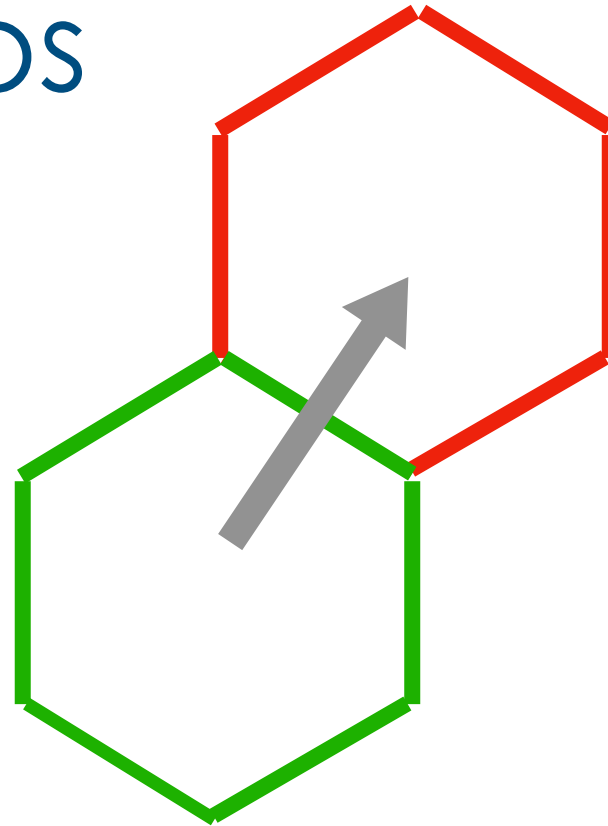
https://github.com/arhands/topological_model
<https://github.com/risi-kondor/ptens>

	ZINC-12K MAE(% ↓)	ZINC-Full MAE(% ↓)	OGBG-MOLHIV ROC-AUC(% ↑)	TOX21 ROC-AUC(% ↑)
RP-NGF (Murphy et al., 2019)	—	—	—	$0.79.4 \pm 1.00$
GCN (Kipf and Welling, 2017)	0.321 ± 0.009	—	76.07 ± 0.97	—
GIN (Xu et al., 2018)	0.408 ± 0.008	0.088 ± 0.002	75.58 ± 1.40	—
GINE (Hu et al., 2019)	0.252 ± 0.014	0.088 ± 0.002	75.58 ± 1.40	86.68 ± 0.77
PNA (Corso et al., 2020)	0.133 ± 0.011	0.320 ± 0.032	79.05 ± 1.32	—
HIMP (Fey et al., 2020)	0.151 ± 0.002	0.036 ± 0.002	78.80 ± 0.82	87.36 ± 0.50
CIN (Bodnar et al., 2021a)	0.079 ± 0.006	0.022 ± 0.002	80.94 ± 0.57	—
DS-GNN (EGO+) (Bevilacqua et al., 2022)	0.105 ± 0.003	—	77.40 ± 2.19	76.39 ± 1.18
DSS-GNN (EGO+) (Bevilacqua et al., 2022)	0.097 ± 0.006	—	76.78 ± 1.66	77.95 ± 0.40
GNN-AK+ (Zhao et al., 2022)	0.091 ± 0.011	—	79.61 ± 1.19	—
SUN (EGO+) (Frasca et al., 2022)	0.084 ± 0.002	—	80.03 ± 0.55	—
First order P-tensors (our model)	0.075 ± 0.003	0.024	80.47 ± 0.87	84.95 ± 0.58

Schur Nets

Equivariance to the automorphism group of subgraphs
[Qingqi Zhang, Ruize (Richard) Xu & K, 2024]

Automorphism groups

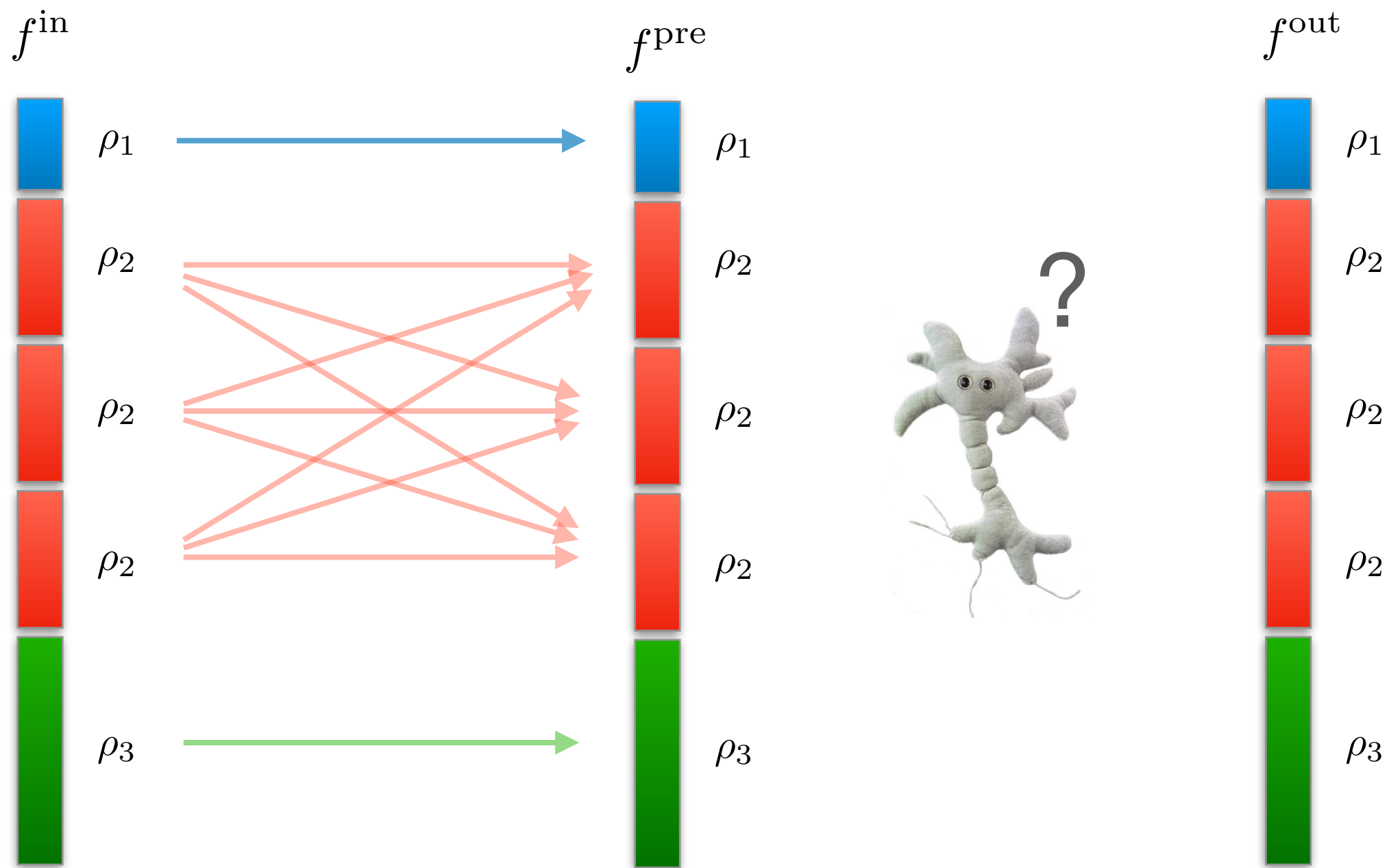


The automorphism group $\text{Aut}(\mathcal{G})$ of a graph is the subgroup of permutations that leave the adjacency matrix fixed:

$$\sigma \circ A = A \quad \Longleftrightarrow \quad \sigma \in \text{Aut}(\mathcal{G})$$

We want the operations on each subgraph to be equivariant to the automorphism group of the subgraph.

Equivariance

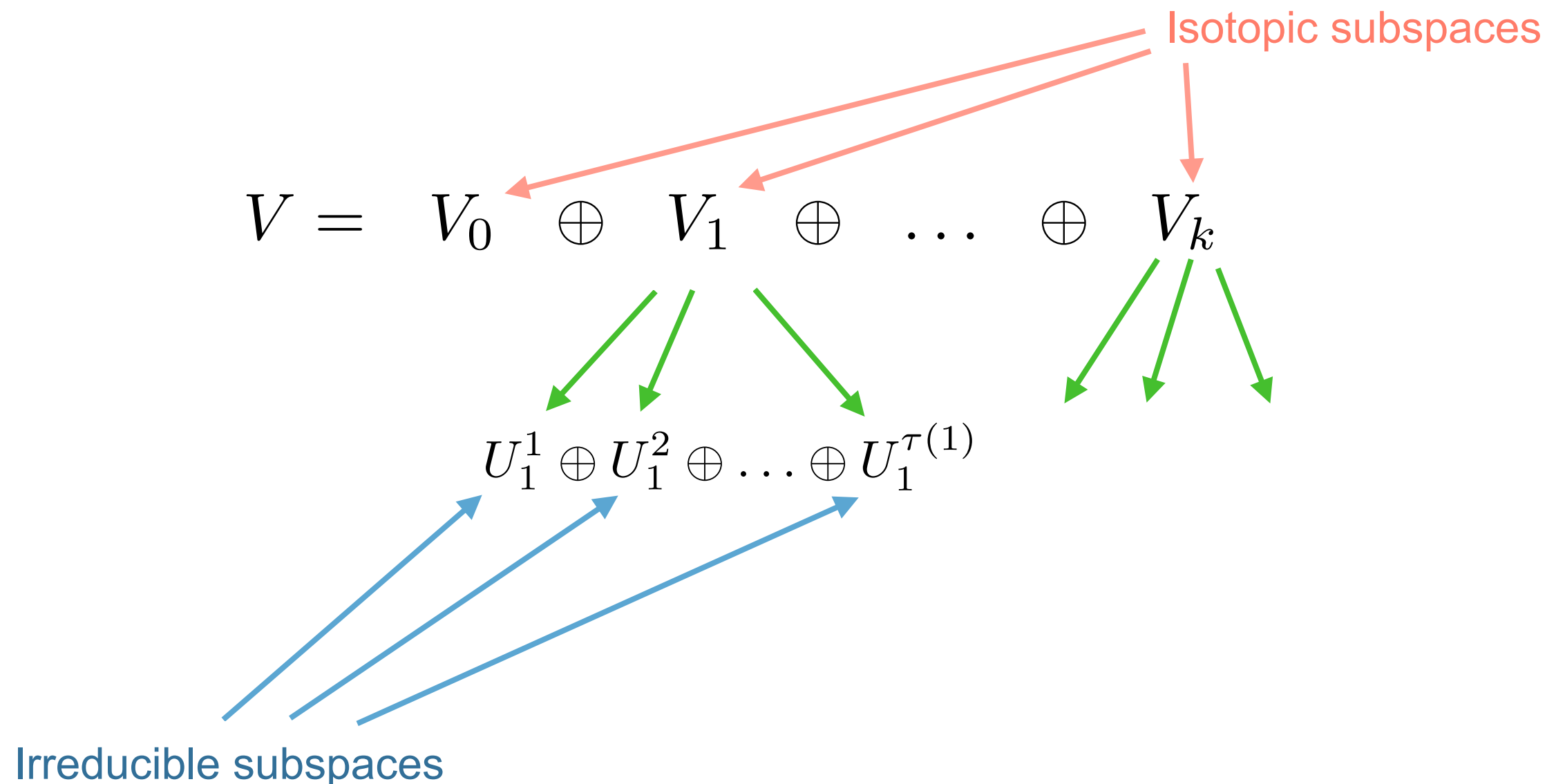


Learned equivariant
linear transformation.

Fixed equivariant
nonlinearity

Harmonic Analysis

Peter-Weyl Thm: Any finite dimensional representation of a compact group G reduces into a direct sum of irreducible representation.



Bare bones harmonic analysis

Ultimately we only need *some* decomposition into invariant subspaces

$$V = V_0 \oplus V_1 \oplus \dots \oplus V_k$$

not necessarily the finest. How about just using the eigenspaces of the Laplacian?

Letting Π_i denote the projector onto the i 'th eigenspace of L this gives (1st order case)

$$f^{\text{out}} = \sum_i \Pi_i^\top w_i \Pi_i f^{\text{in}}$$

 Learnable weights

[Schur Nets, NeurIPS 2024]

Graph	Aut_S	# of distinct Eigenvalues (<i>Schur</i> Layer)	$\sum_i (\kappa_i)^2$ <i>irreps</i> approach	$\sum_i \kappa_i$
6-cycle	D_6	4	4	4
5-cycle	D_5	3	3	3
4-cycle	D_4	3	3	3
3-cycle	D_3	2	2	2
5-star	S_4	3	5	3
4-star	S_3	3	5	3
3-path	S_2	3	5	3
n-cliques	S_n	2	2	2
5-cycle with one branch	S_2	6	20	6
6-cycle with one branch	S_2	7	29	7

Summary

P-tensors are an abstraction that can

- unify a wide range of graph, hypergraph and simplicial neural networks
- make it easy to construct domain specific networks tailored to particular types of substructures
- afford a unified efficient implementation on GPUs

Extensions:

- Attention
- Generative models
- Combine with spectral ideas
- Incorporating local topology