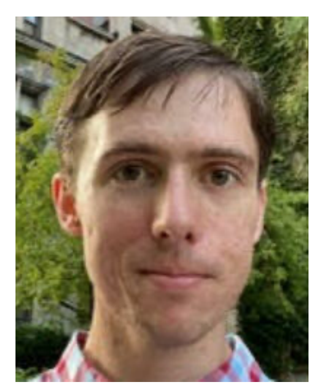
Higher order graph neural networks with P-tensors

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

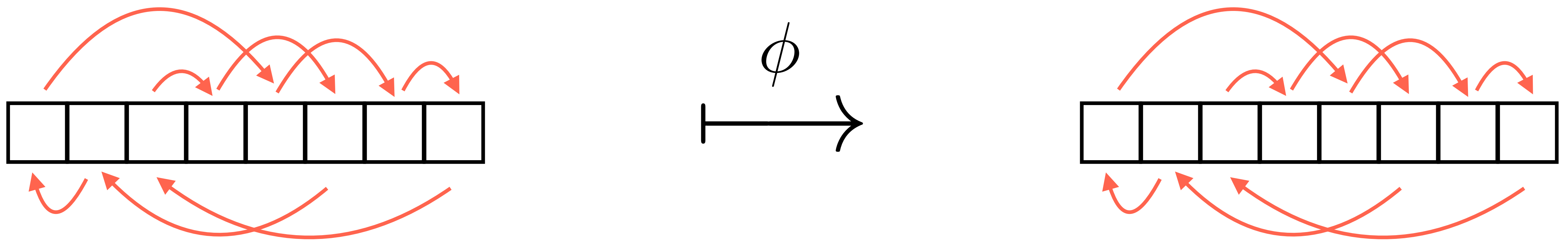
$$\begin{tikzcd} f^{\text{in}} \arrow[r, "T_g^{(1)}"] \arrow[d, "\phi"] & f^{\text{in}\prime} \arrow[d, "\phi"] \\ f^{\text{out}} \arrow[r, "T_g^{(2)}"] & f^{\text{out}\prime} \end{tikzcd}$$

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

First order permutation equivariance

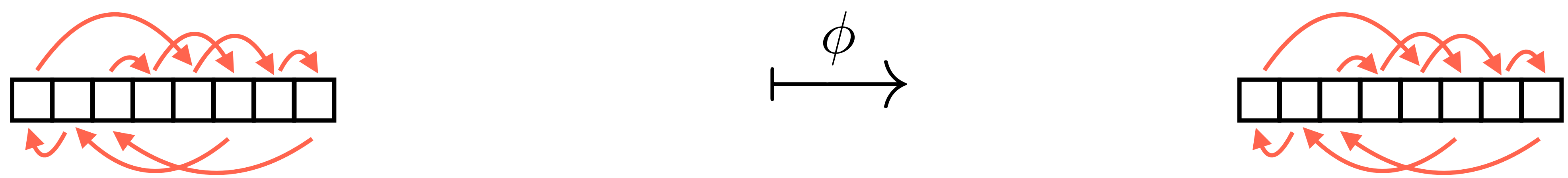
$$f \overset{\phi}{\longmapsto} f'$$

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



[Deep sets: Zaheer et al., 2017]

First order permutation equivariance



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

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

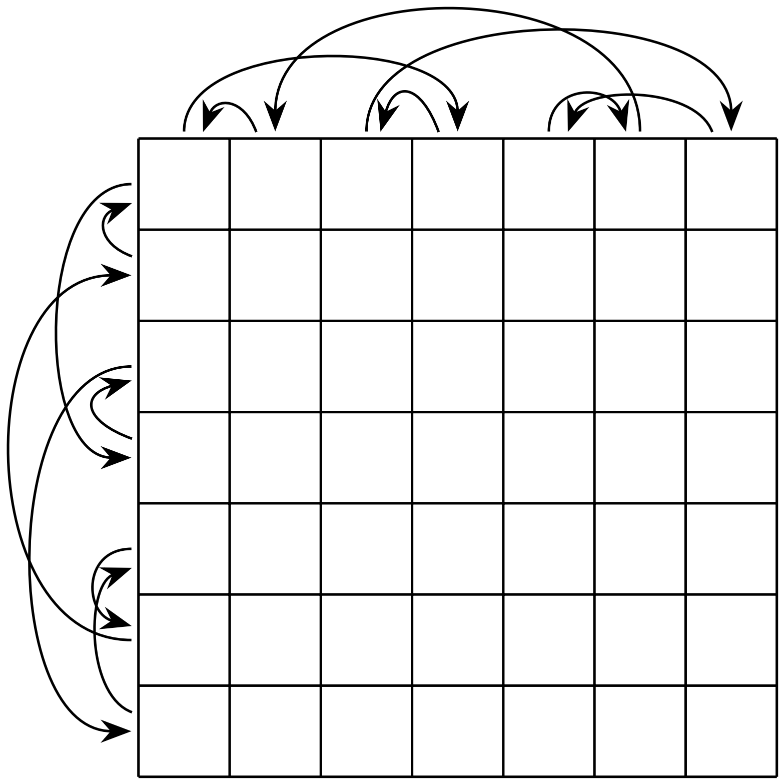
$$\boxed{f'_i = \alpha_1 f_i + \alpha_2 \frac{1}{n}\sum_j f_j}$$

[Deep sets: Zaheer et al., 2017]

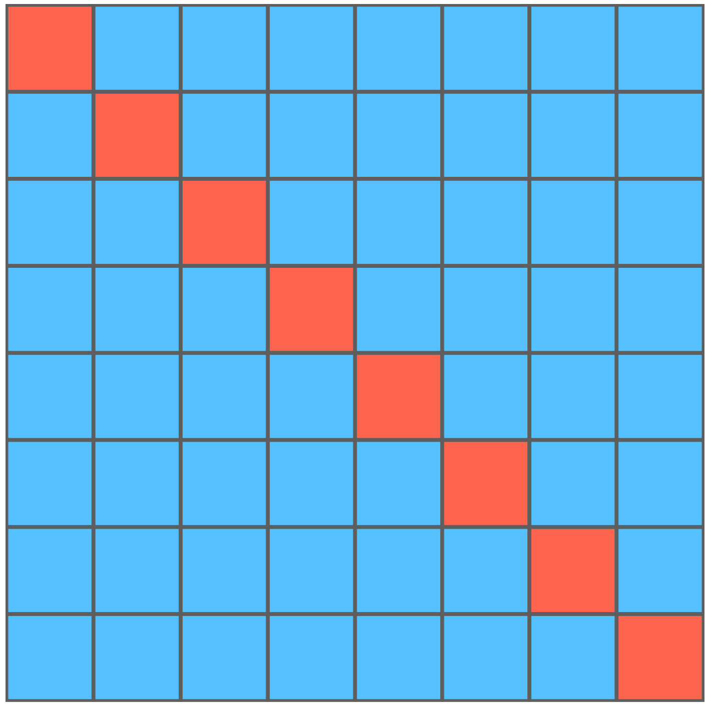
Second order permutation action

$$A \longmapsto A^{\sigma}$$

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



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

$$A_{i,i}$$

$$\overline{\mathrm{diag}(A)} = \tfrac{1}{n}\textstyle\sum_i A_{i,i}$$

$$A_{i,j}$$

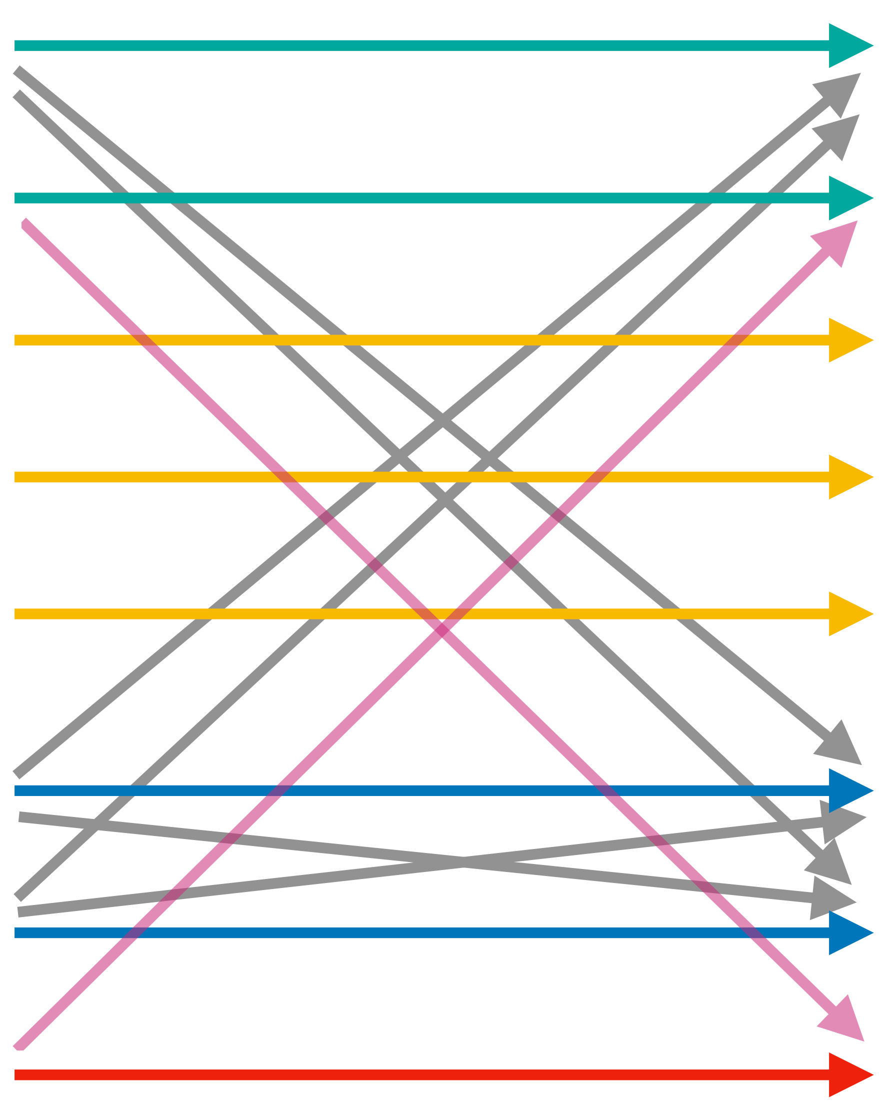
$$A^{\mathrm{symm}}_{i,j} = A_{i,j} + A_{j,i}$$

$$A^{\mathrm{asymm}}_{i,j} = A_{i,j} - A_{j,i}$$

$$\overline{A_{i,*}} = \tfrac{1}{n}\textstyle\sum_j A_{i,j}$$

$$\overline{A_{*,j}} = \tfrac{1}{n}\textstyle\sum_i A_{i,j}$$

$$\overline{A_{*,*}} = \tfrac{1}{n^2}\textstyle\sum_{i,j} A_{i,j}$$



$$A'_{i,i}$$

$$\overline{\mathrm{diag}(A')} = \tfrac{1}{n}\textstyle\sum_i A'_{i,i}$$

$$A'_{i,j}$$

$$A'^{\mathrm{symm}}_{i,j} = A'_{i,j} + A'_{j,i}$$

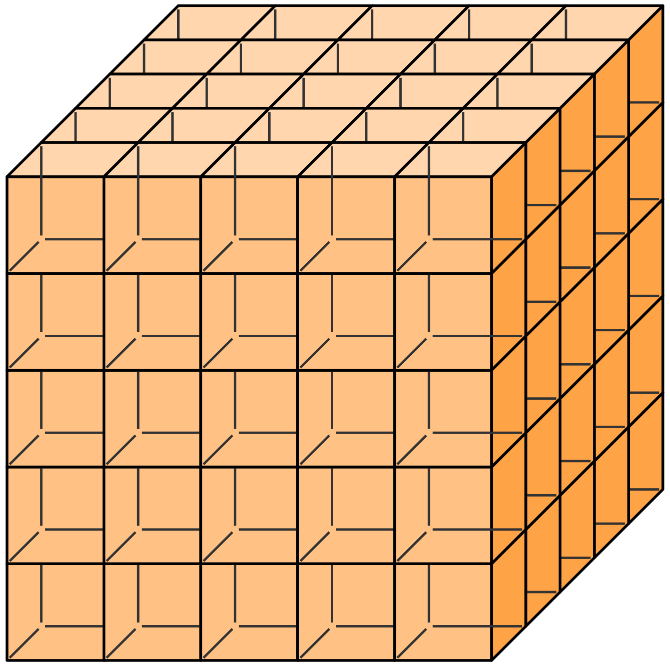
$$A'^{\mathrm{asymm}}_{i,j} = A'_{i,j} - A'_{j,i}$$

$$\overline{A'_{i,*}} = \tfrac{1}{n}\textstyle\sum_j A'_{i,j}$$

$$\overline{A'_{*,j}} = \tfrac{1}{n}\textstyle\sum_i A'_{i,j}$$

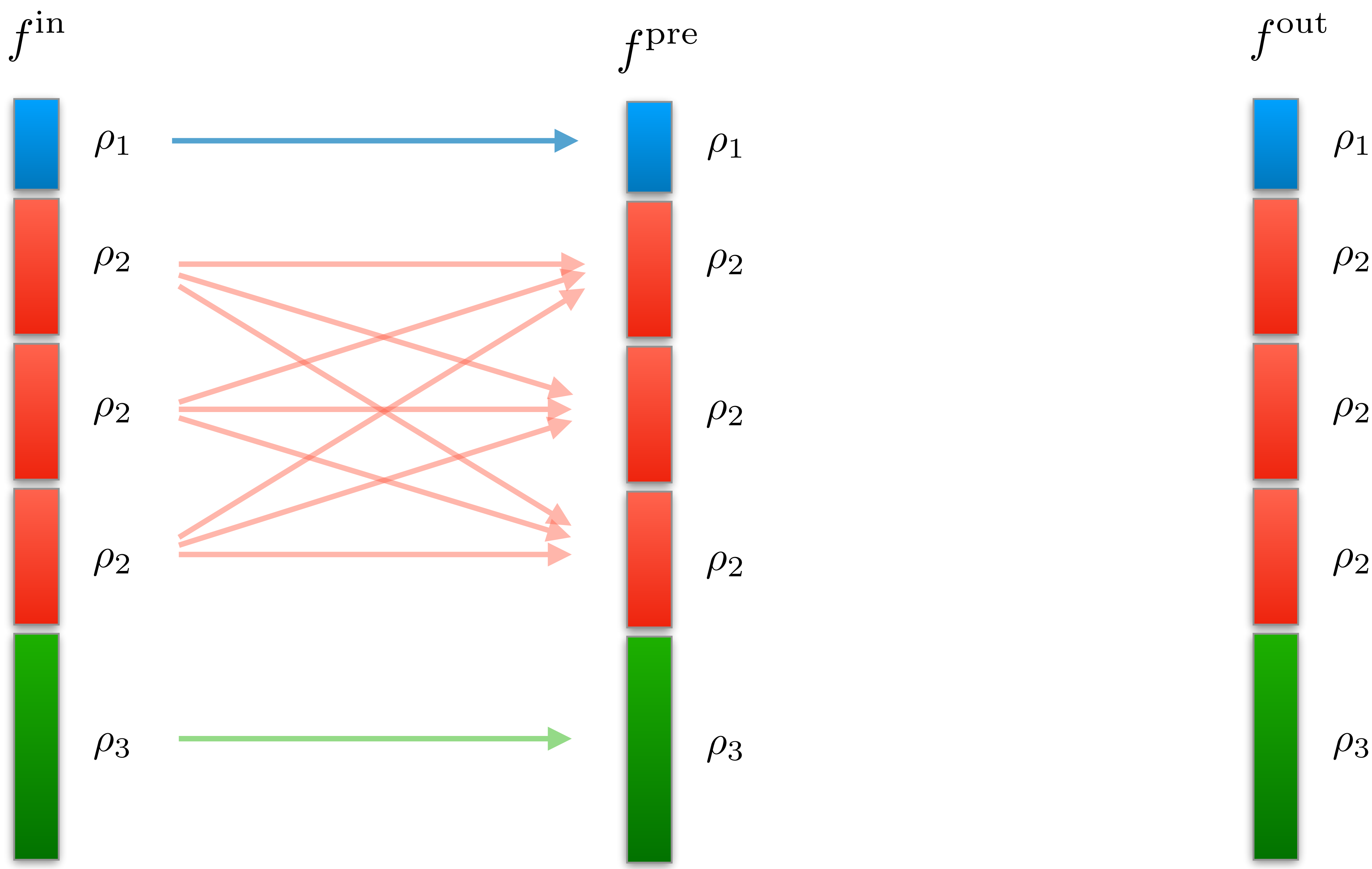
$$\overline{A'_{*,*}} = \tfrac{1}{n^2}\textstyle\sum_{i,j} A'_{i,j}$$

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



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

Equivariant neuron



Learned equivariant *linear* transformation.

Fixed equivariant nonlinearity

Representations of $\mathbb{S}_n$

The **permutation matrices**

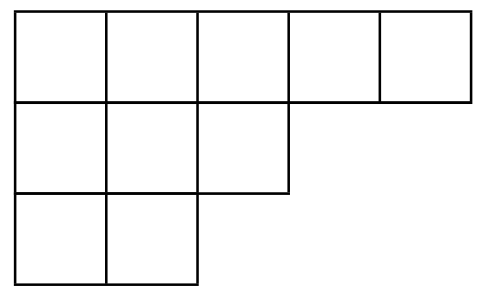
$$P^{\sigma}_{i,j} = \begin{cases} 1 & \text{if } \sigma(j) = i \\ 0 & \text{otherwise} \end{cases}$$

form a representation ρ_{def} of $\mathbb{S}_n$.

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.,

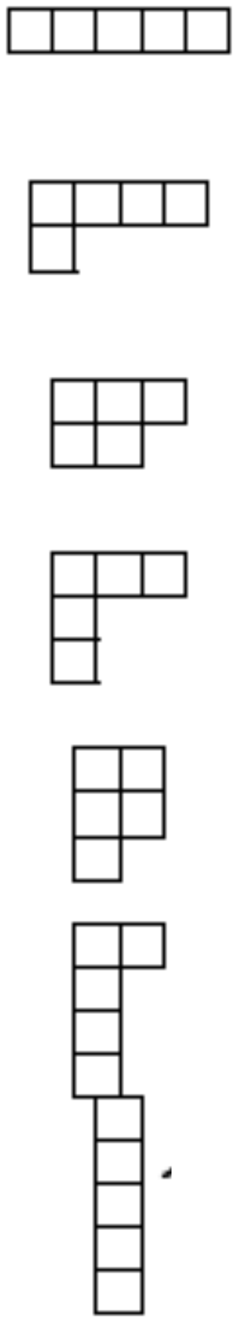


A Young diagram is really just the pictorial representation of an **integer partion**, i.e., a sequence $(\lambda_1, \lambda_2, \ldots, \lambda_k)$ such that $\sum_i \lambda_i = n$.

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 $\{1,2,\ldots,n\}$ 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,

$$\begin{array}{|c|c|c|c|c|}\hline 1 & 2 & 4 & 6 & 7 \\ \hline 3 & 5 & 8 \\ \cline{1-3} 9 \\ \cline{1-1}\end{array}$$

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 $i+1$.

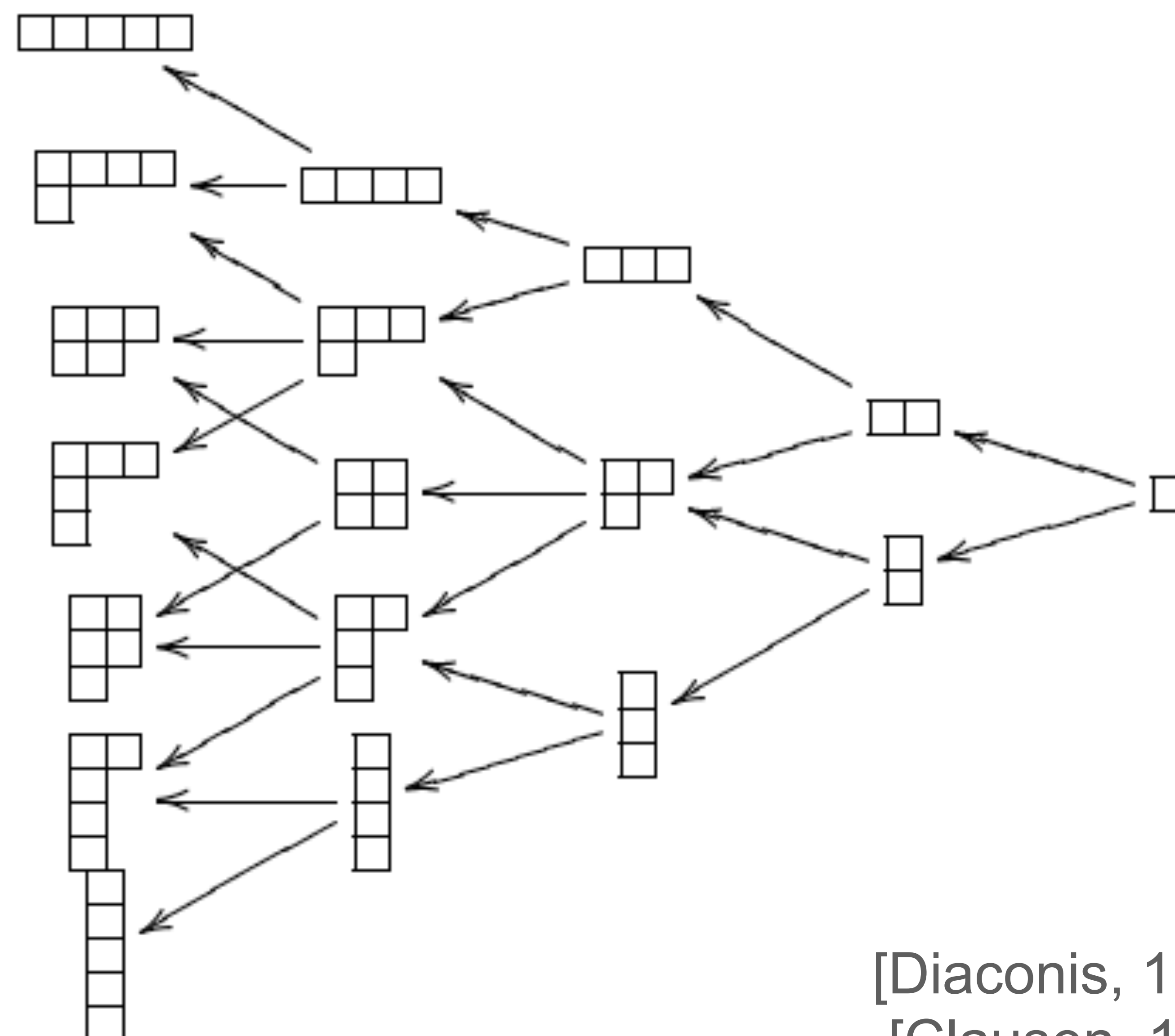
Sizes of the irreps

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

FFTs on the symmetric group

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

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



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

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

broadcast

reduction

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

Proposition (Maron et al.). *The space of linear maps* $\phi\colon \mathbb{R}^{d^{k_1}} \to \mathbb{R}^{d^{k_2}}$ *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* $\{1, 2, \ldots, k_1 + k_2\}$.

[Maron, Hamu, Shamir & Lipman, 2019]

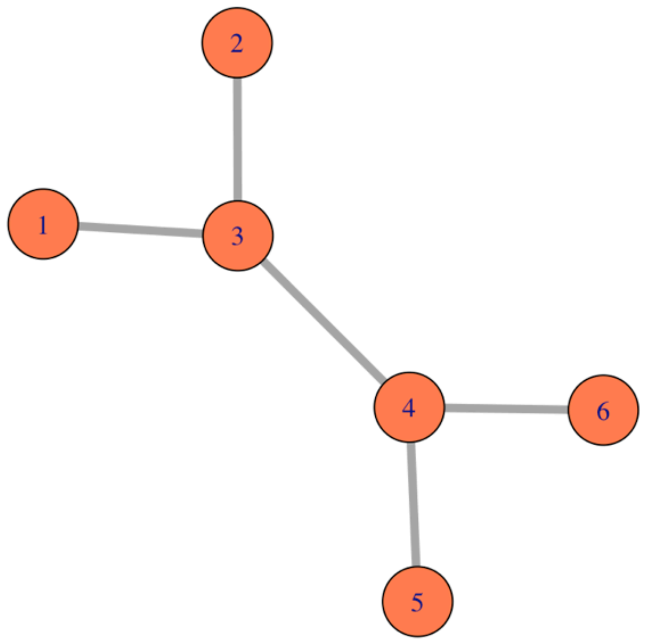
Do we really want equivariance to $\mathbb{S}_n$?

$$
\begin{tikzcd}
f^{\mathrm{in}} \arrow[r, "T_g^{(1)}"] \arrow[d, "\phi"] & f^{\mathrm{in}\prime} \arrow[d] \\
f^{\mathrm{out}} \arrow[r, "T_g^{(2)}"] & f^{\mathrm{out}\prime}
\end{tikzcd}
$$

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

NO!

Automorphism groups



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

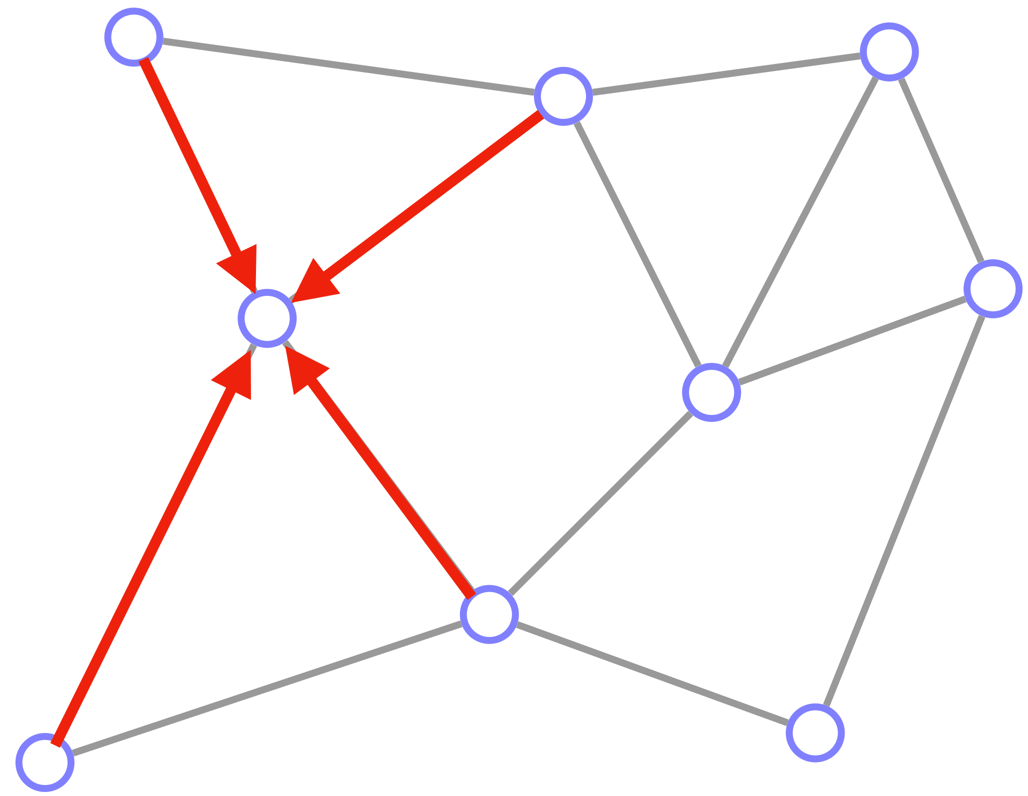
$$\sigma \circ A = A \qquad \Longleftrightarrow \qquad \sigma \in \mathrm{Aut}(\mathcal{G})$$

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

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

2. Why I don’t like GNNs

MPNN

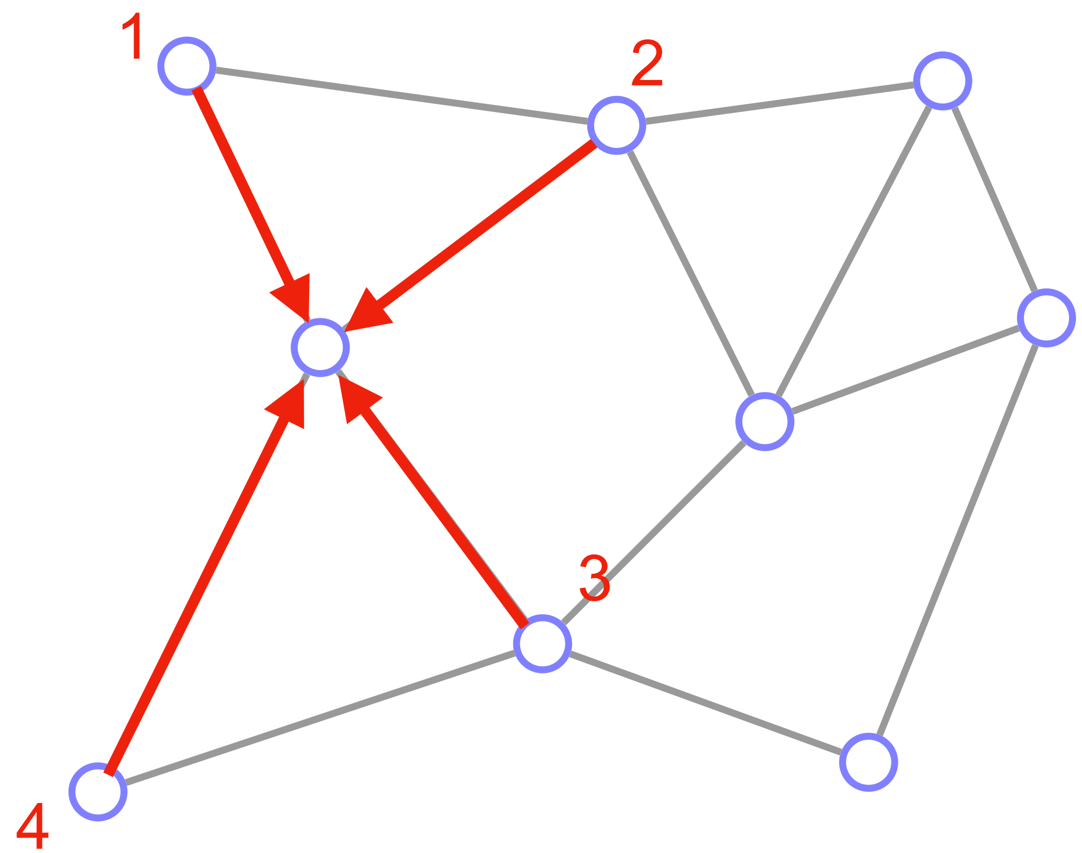


$$f_i^{\ell+1} = \xi\left(W \sum_{j \in \mathcal{N}(i)} f_j^{\ell} + b\right)$$

Associative & commutative

[Micheli, 2009] [Atwood & Towsley, 2016] [Gilmer et al., ICML 2017]

1. MPNNs have amnesia



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

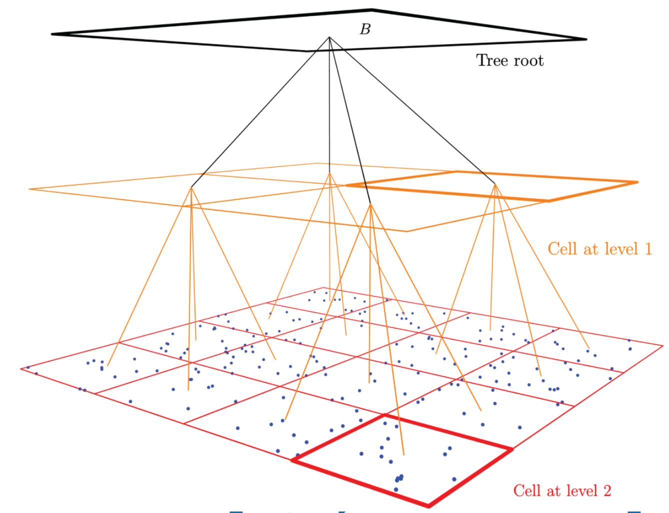
2. MPNNs have no sense of global structure


Spectral Networks and Deep Locally Connected Networks on Graphs

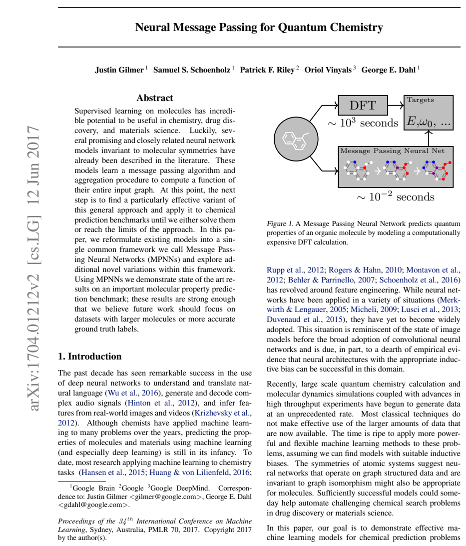
Joan Bruna, New York University
Wojciech Zaremba, New York University
Arthur Szlam, The City College of New York
Yann LeCun, New York University

Abstract

1 Introduction



Multiscale


Neural Message Passing for Quantum Chemistry

Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, George E. Dahl

Abstract

1. Introduction

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

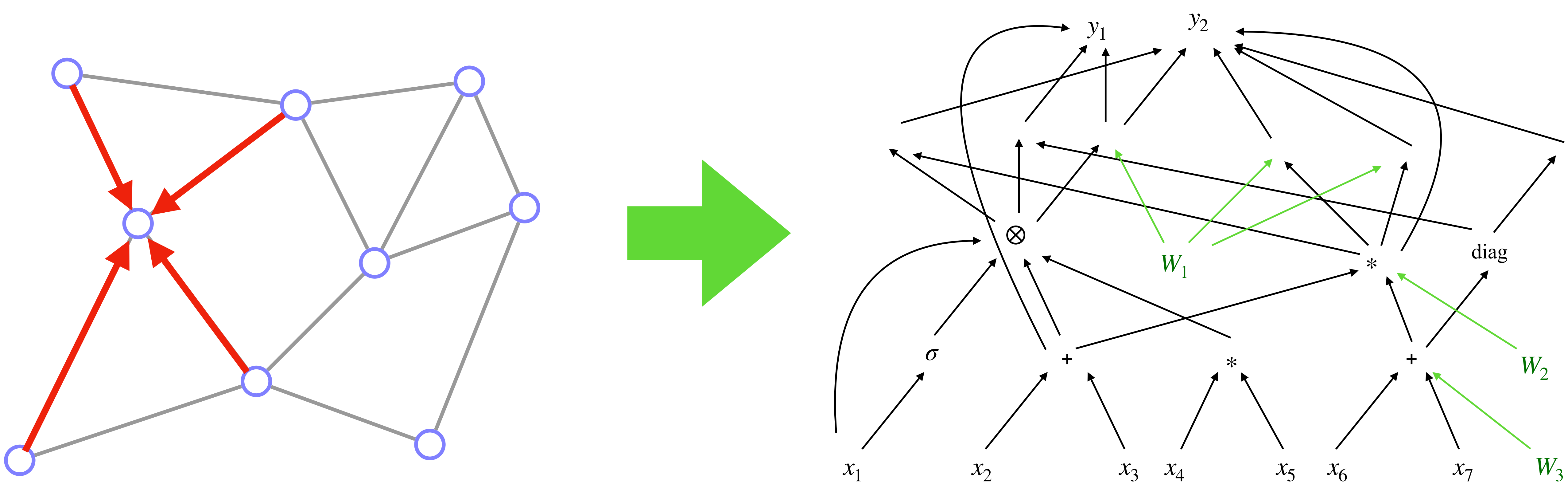
$$x_{k+1,j} = h\left(V \sum_{i=1}^{f_{k-1}} F_{k,i,j} V^T x_{k,i}\right) \quad (j = 1 \ldots f_k) \, ,$$

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

Global

Local

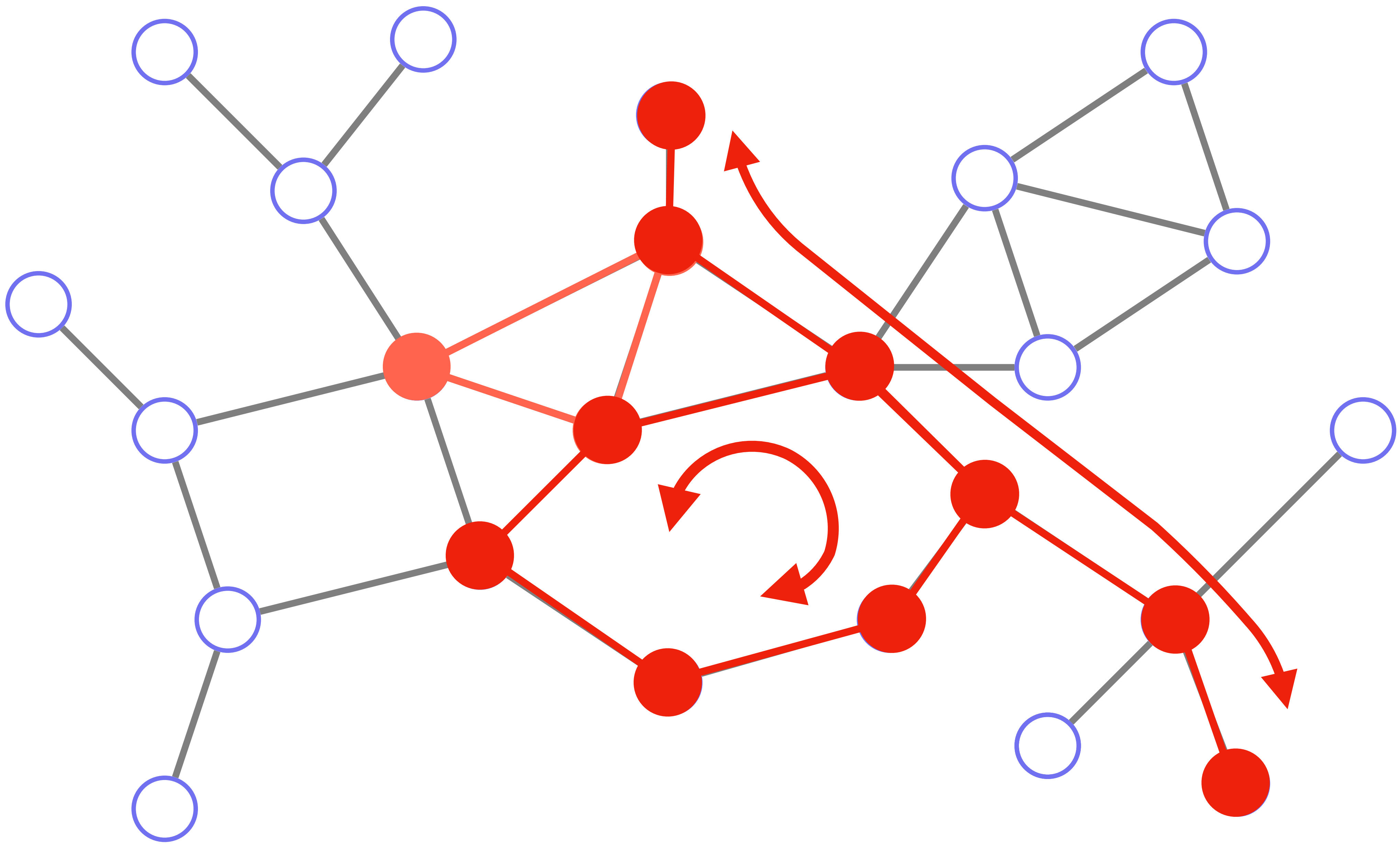
3. MPNNs only encode topology implicitly



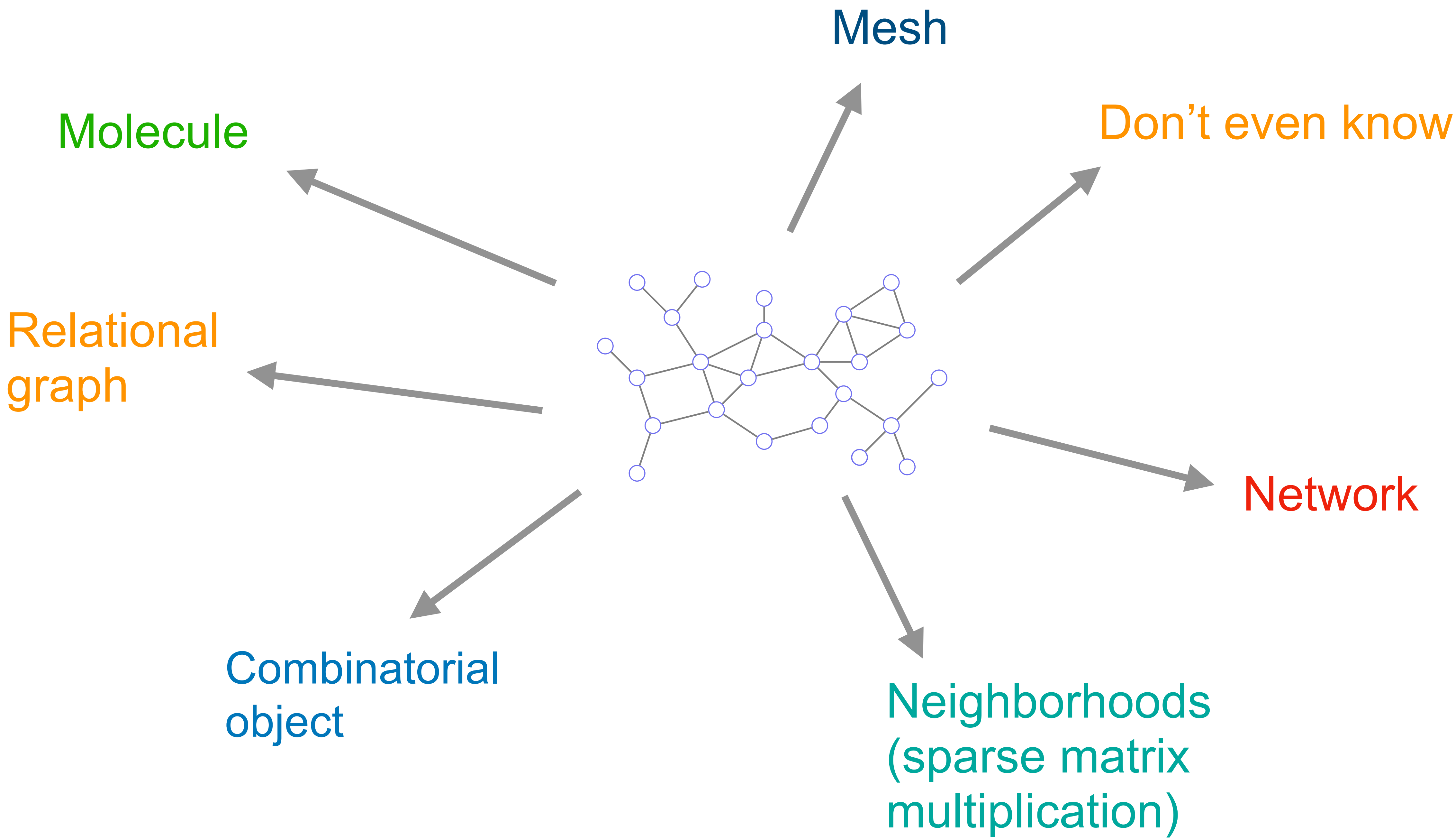
The graph topology determines the structure of the compute graph.

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

4. MPNNs don't reduce to classical convolution

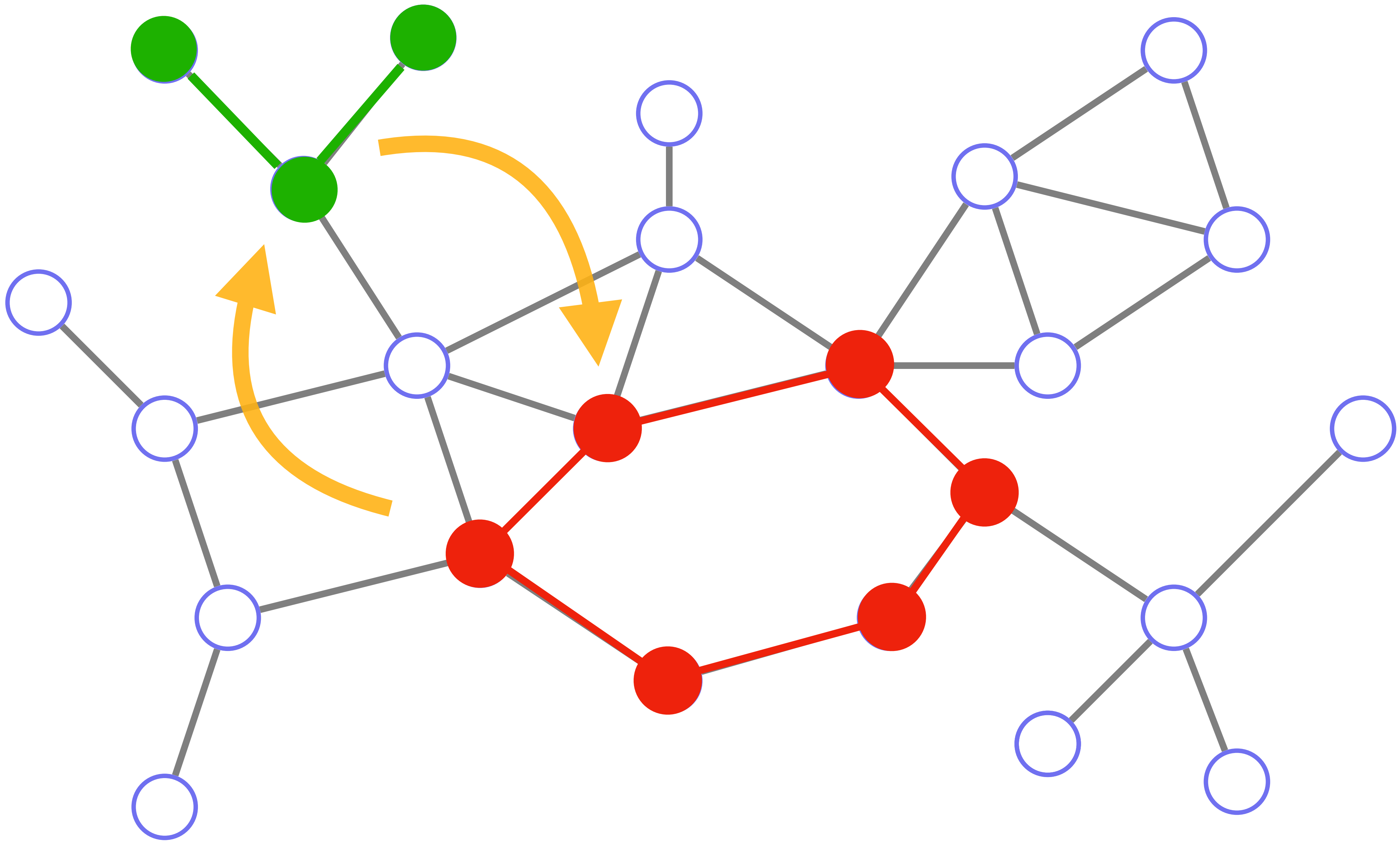


5. What is a graph, anyway?



Higher order message passing

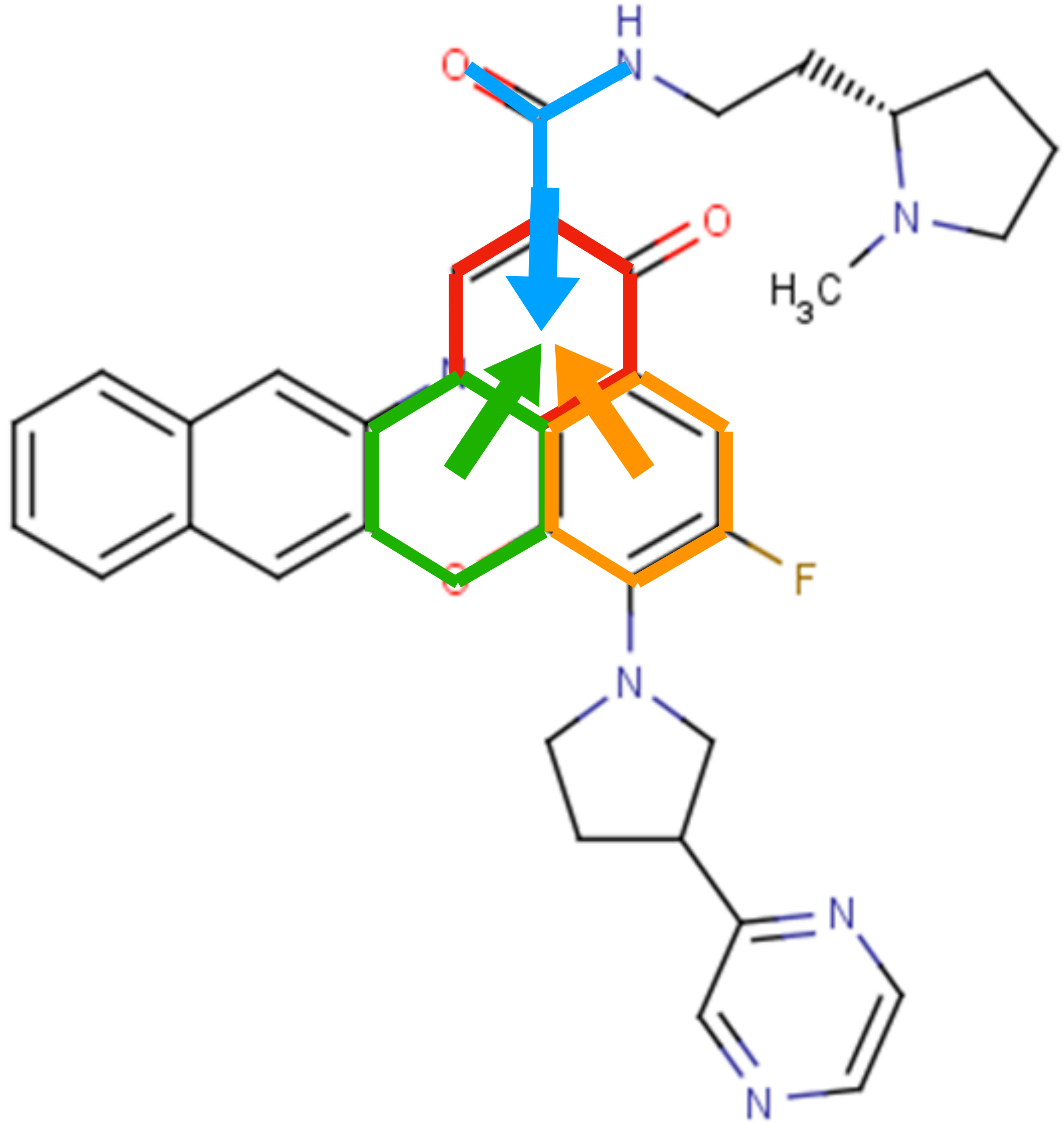
Subgraph neural nets



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

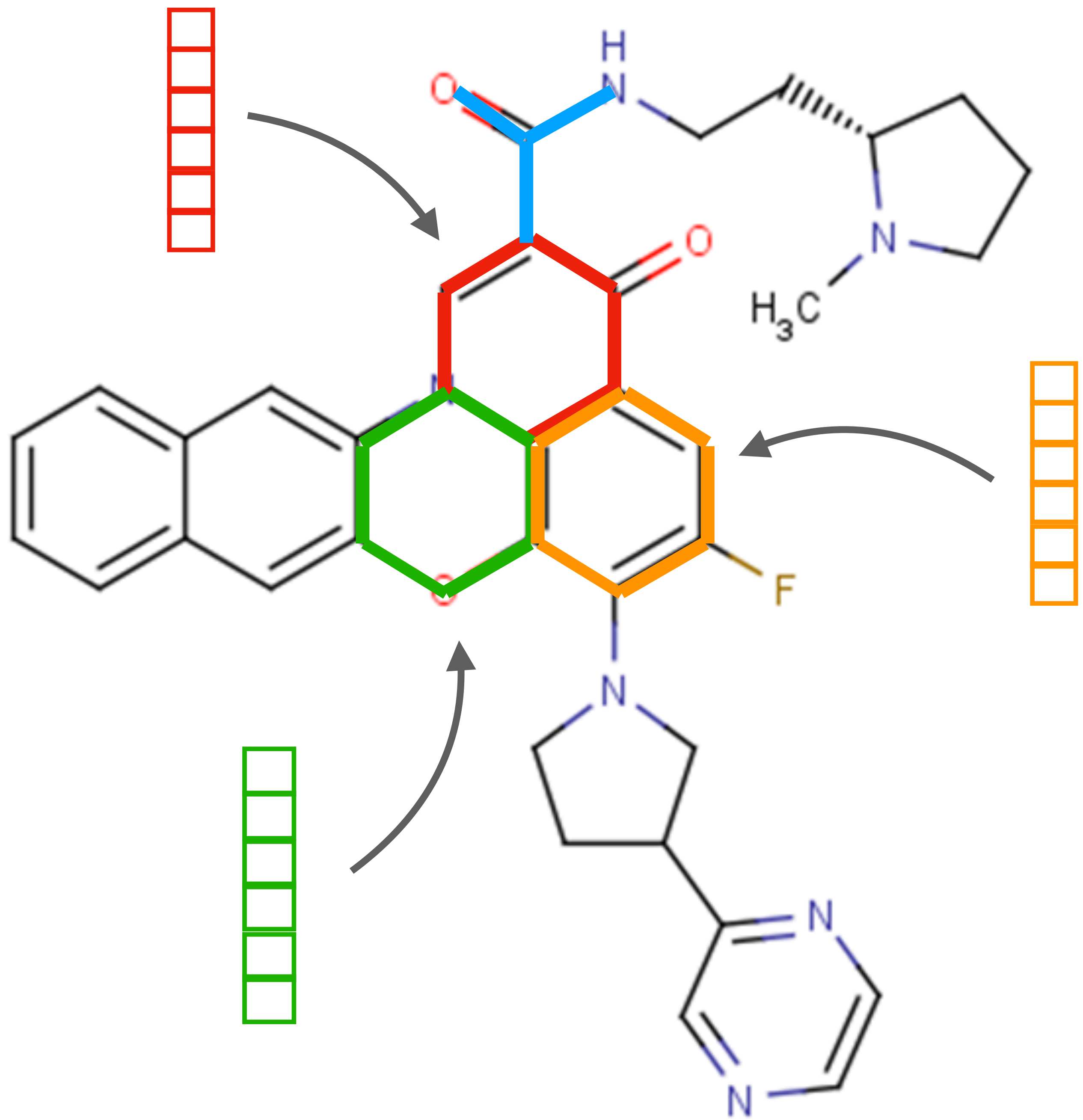
How can we do this in an equivariant way?

[Frasca, Bevilacqua, Bronstein & Maron, 2022]

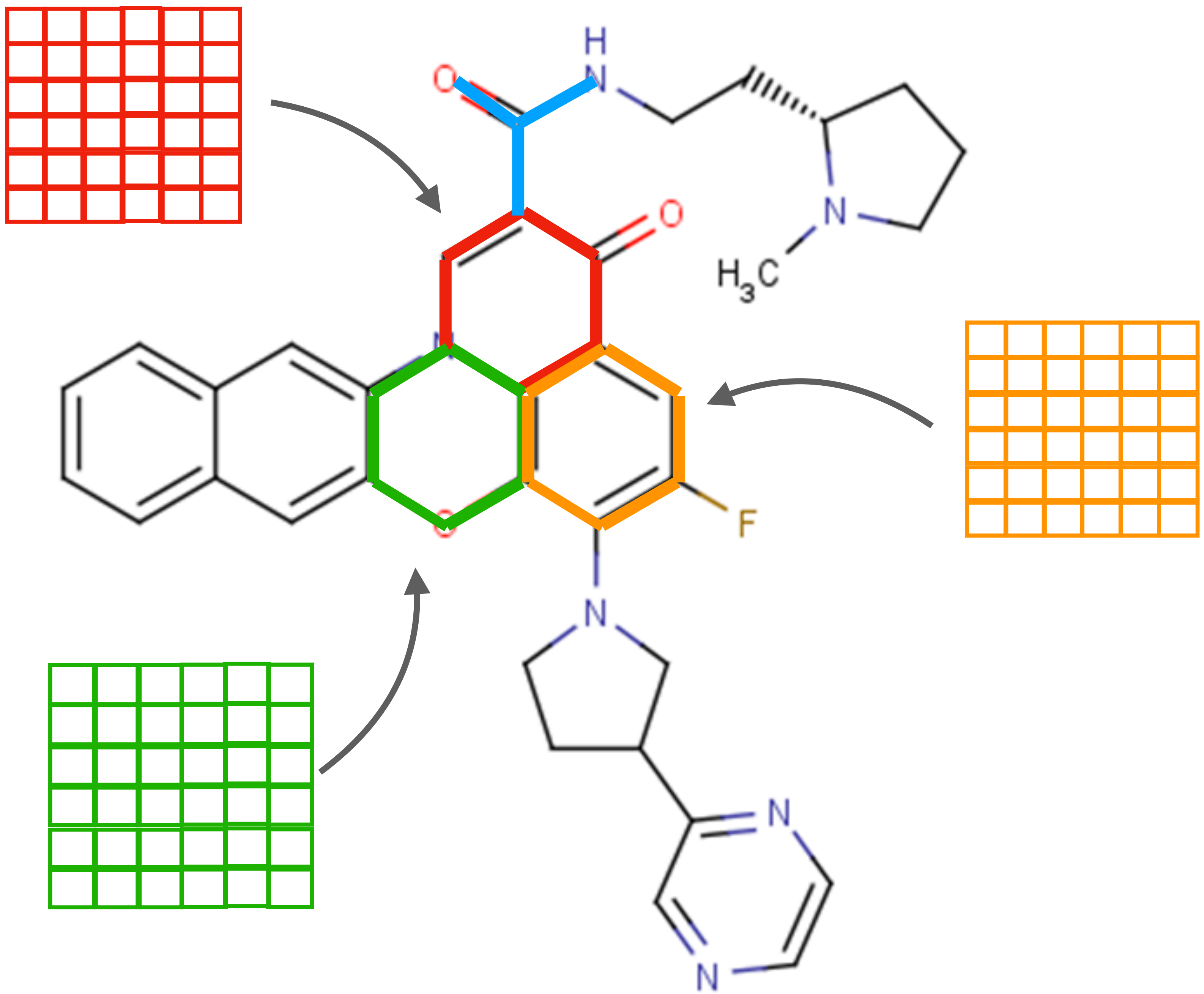


quarfloxin

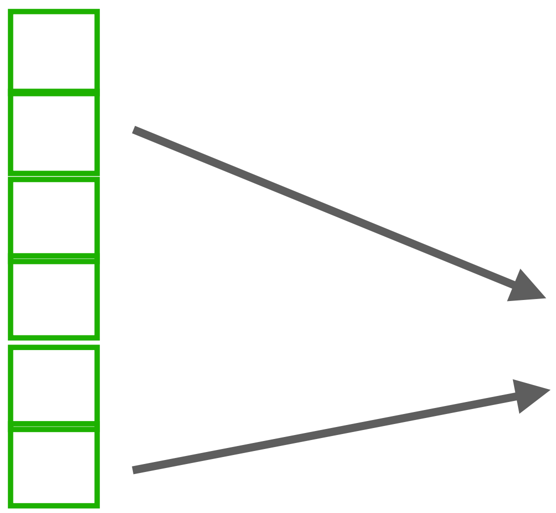
First order subgraph neural nets

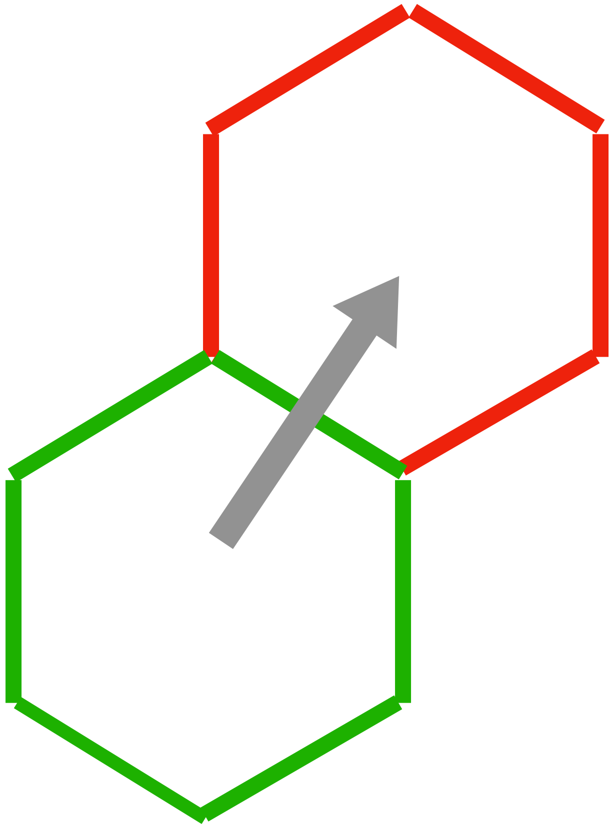


Second order subgraph neural nets

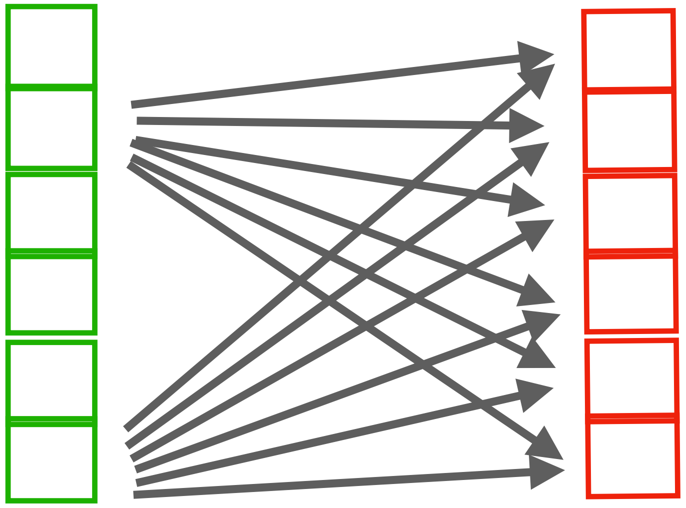


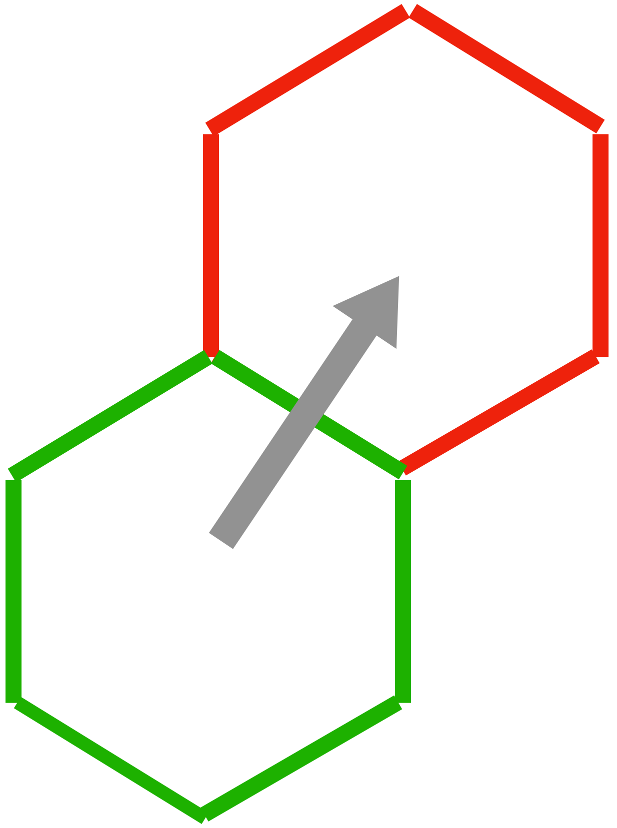
What is the correct generalization of message passing?



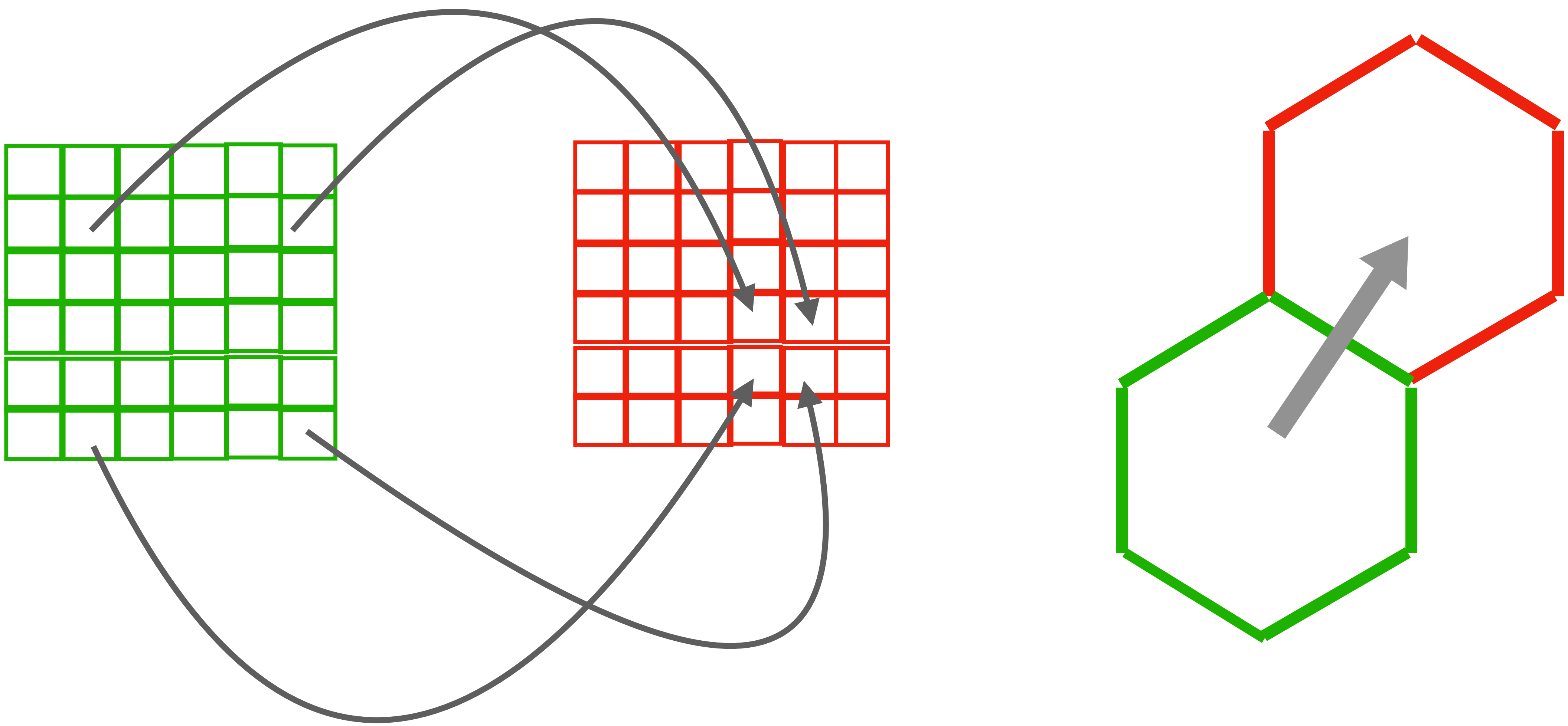


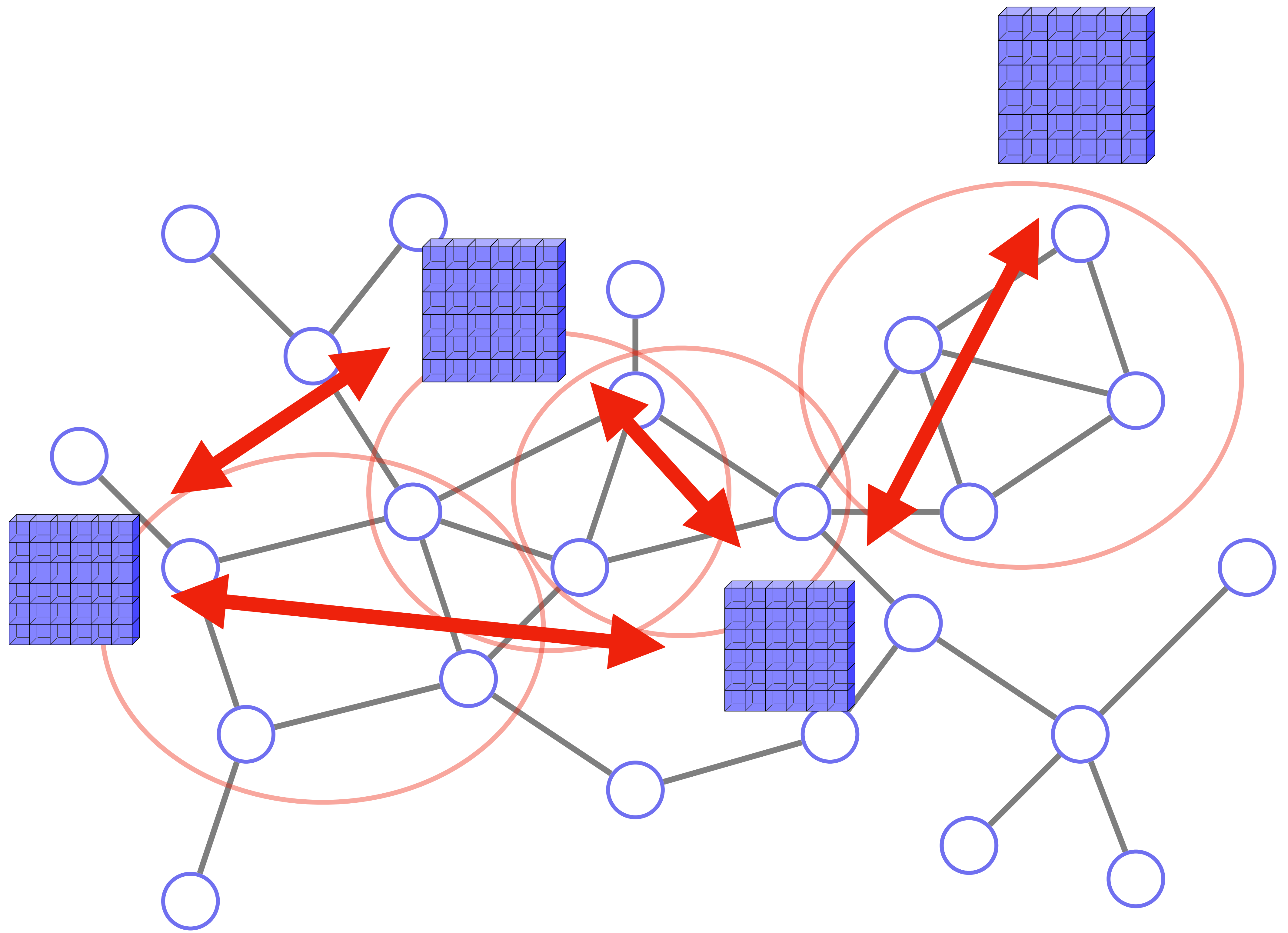
What is the correct generalization of message passing?





What is the correct generalization of message passing?

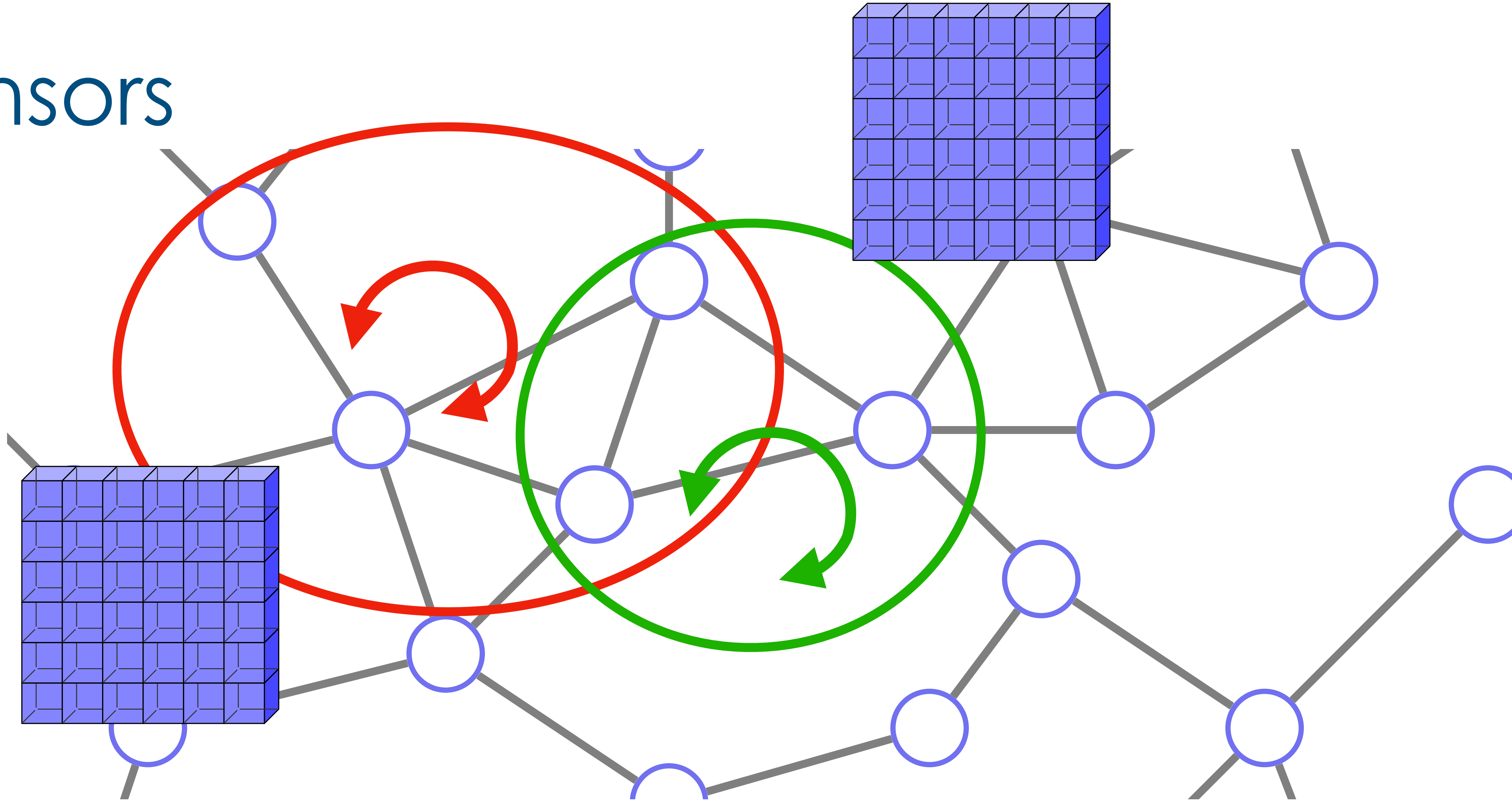




P-tensors

[Andrew Hands, Tiny Sun & K, 2024]

P-tensors

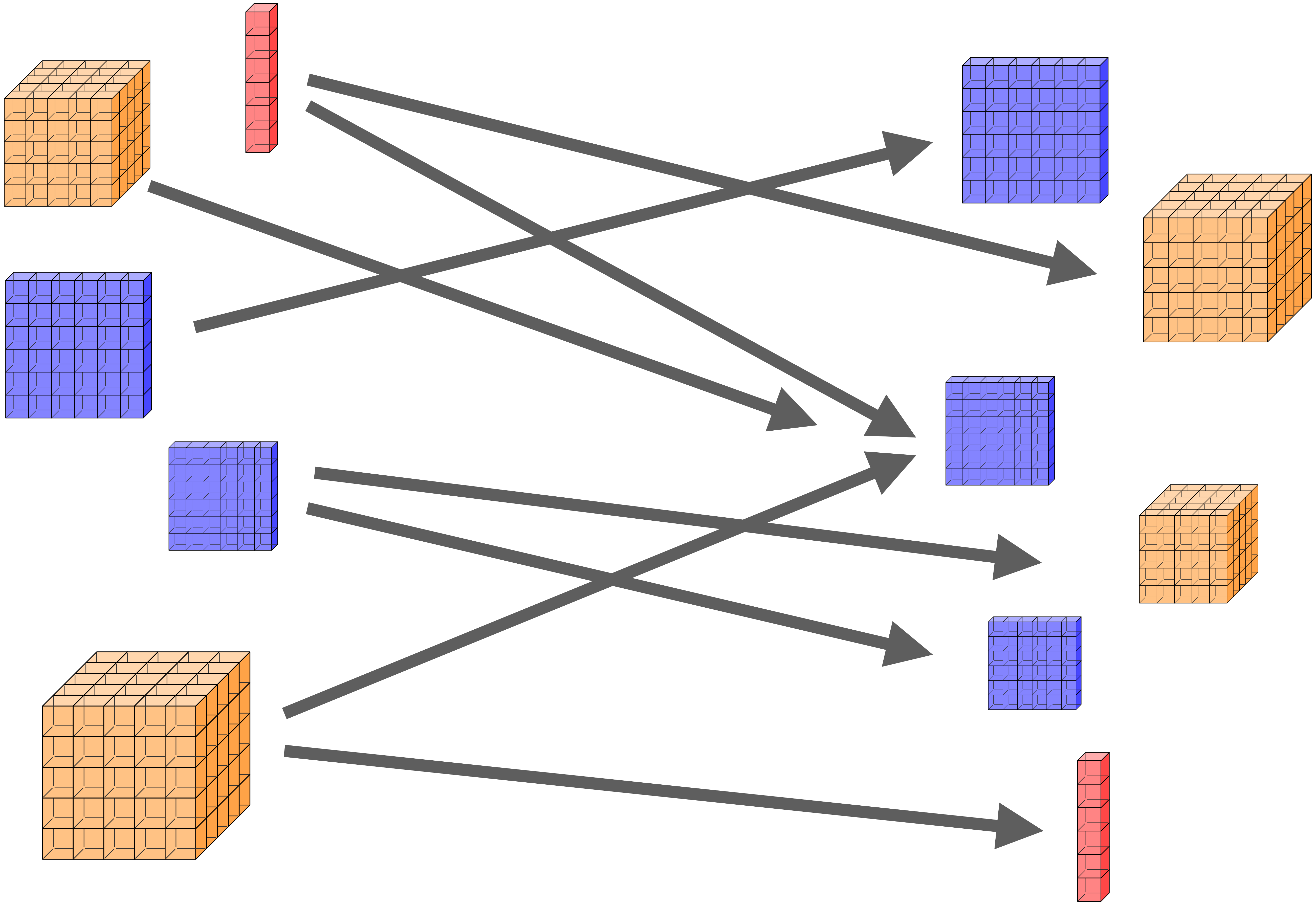


Given an ordered subset of d atoms $(x_1, x_2, \ldots, x_m)$, a tensor $T \in \mathbb{R}^{d \times d \times \ldots \times d}$ that transforms under permutations according to

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

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

[Hands, Sun & K, AISTATS 2024]

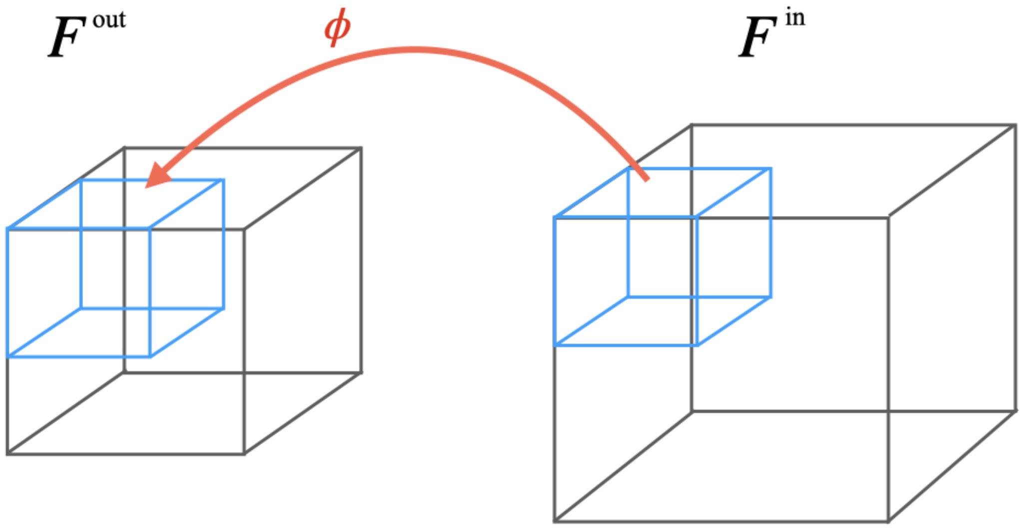


Message passing between P-tensors

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 $\phi\colon T_1 \to T_2$ is permutation equivariant if*

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

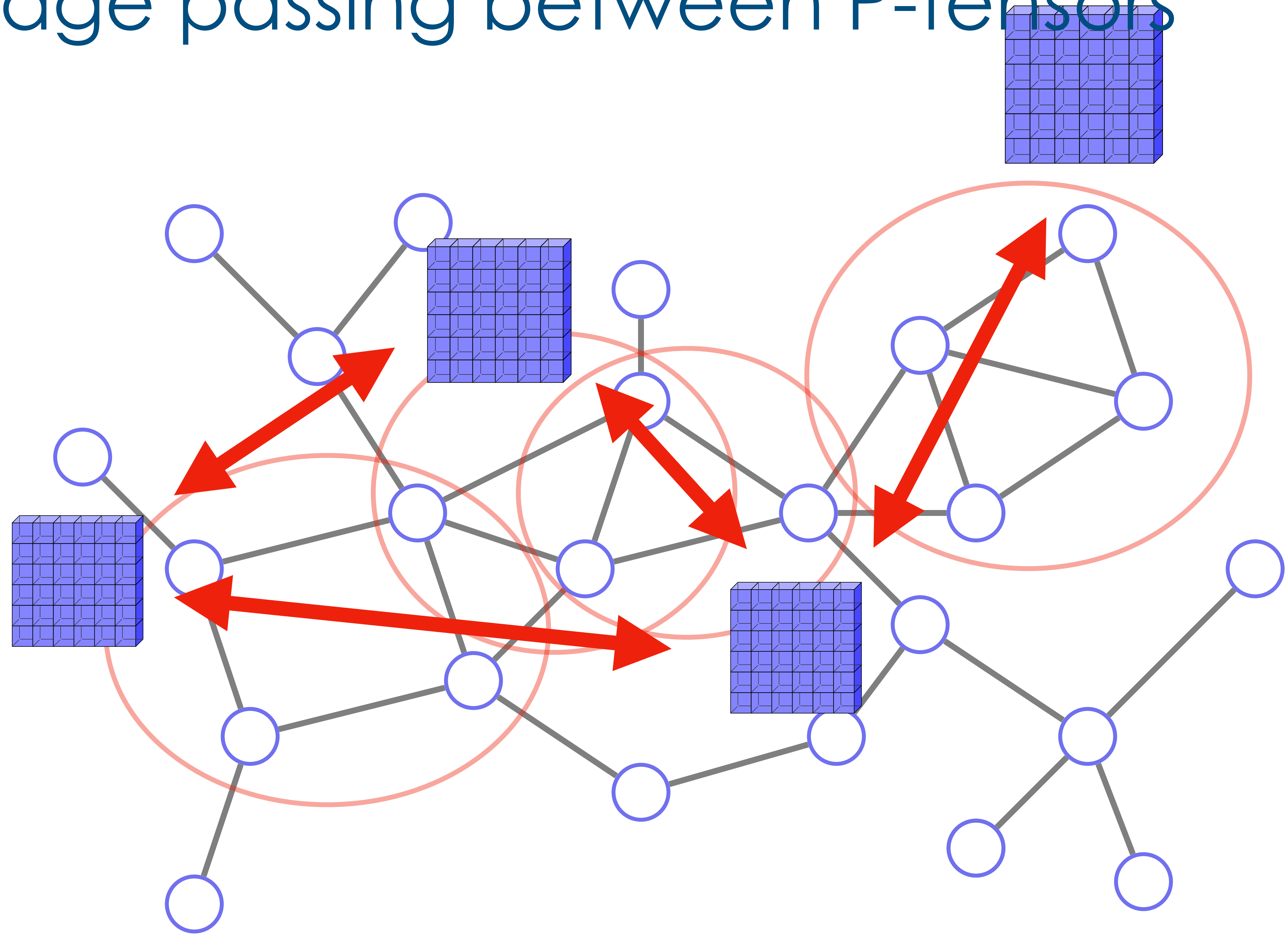
for any permutations σ of $\mathcal{U}$ that fixes both $\mathcal{D}_1$ and $\mathcal{D}_2$ as sets.

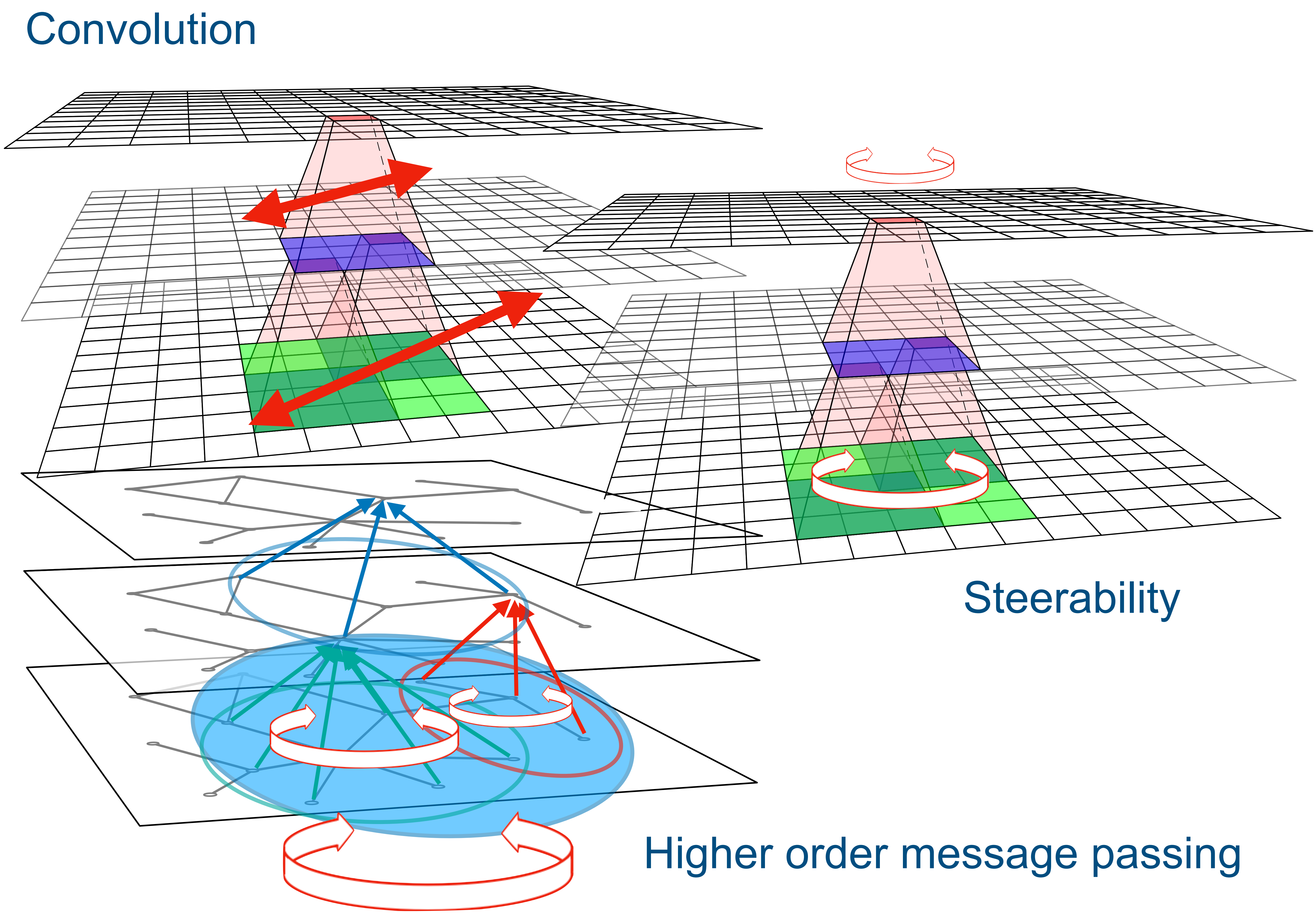


Theorem 2. *Let T_1 and T_2 be two P-tensors with reference domains $\mathcal{D}_1$ and $\mathcal{D}_2$ such that $\mathcal{D}_1 \cap \mathcal{D}_2 \neq \emptyset$ and $\mathcal{D}_1 \not\subseteq \mathcal{D}_2$ and $\mathcal{D}_2 \not\subseteq \mathcal{D}_1$. Then for each partition $\mathcal{P}$ of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

(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



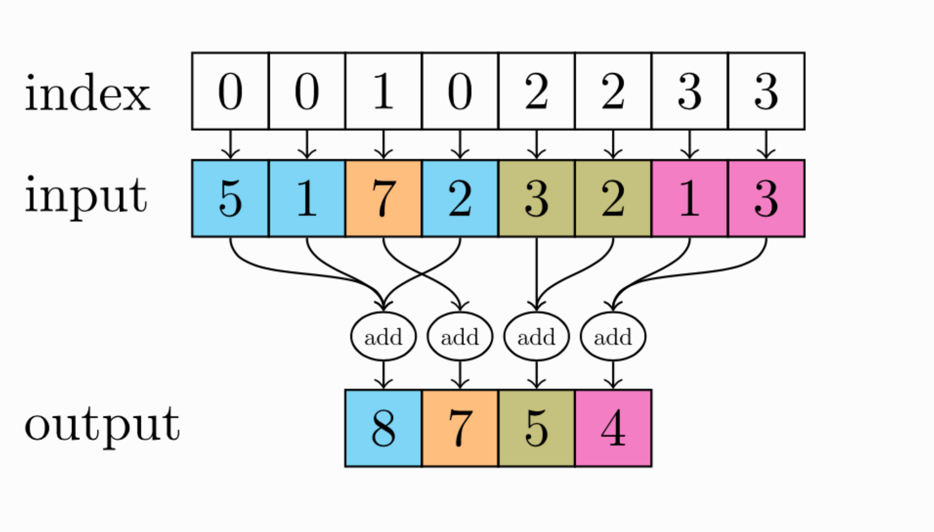


Convolution
Steerability
Higher order message passing

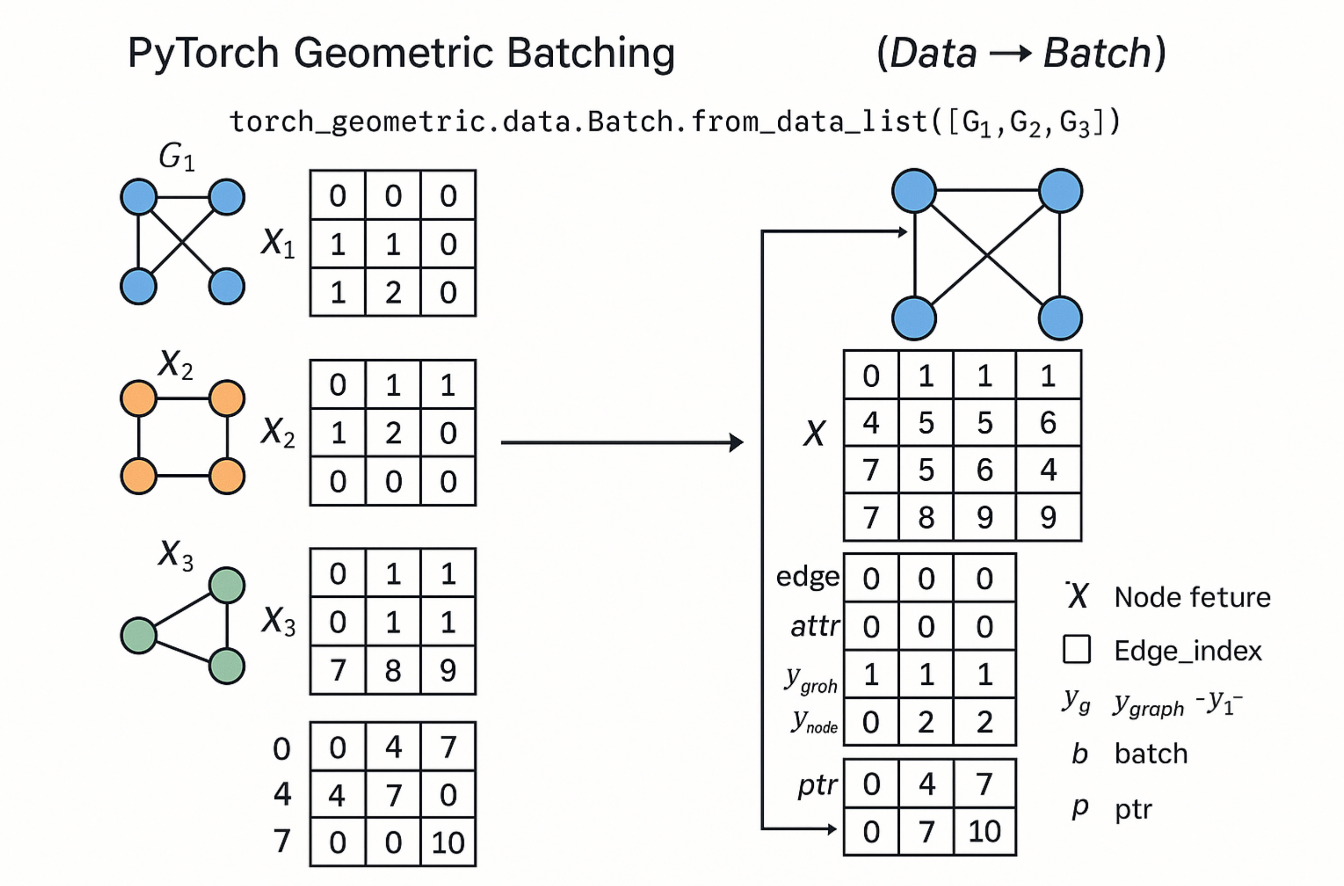
P-tensor software

pytorch-geometric

Efficient gather/scatter



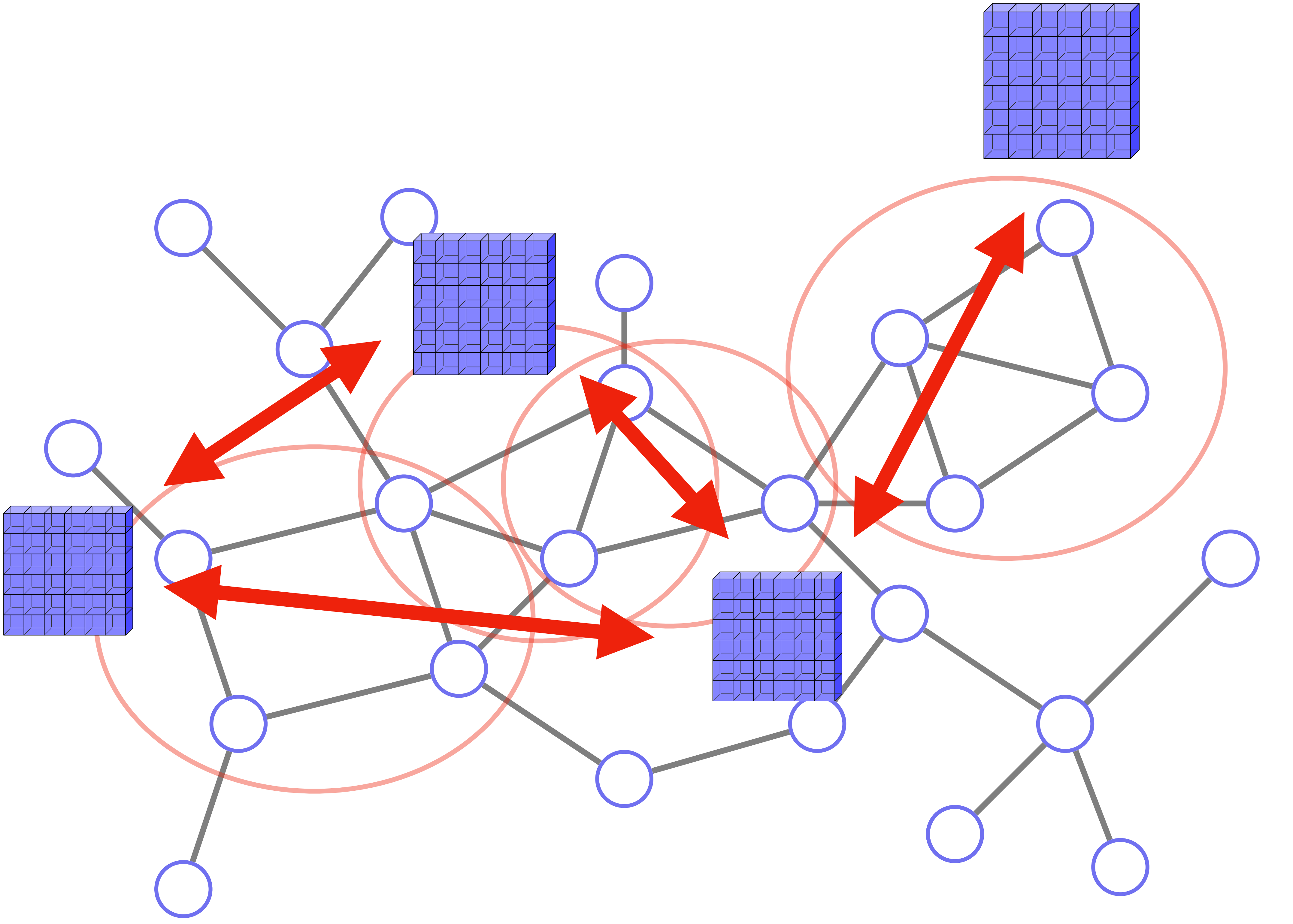
Batching



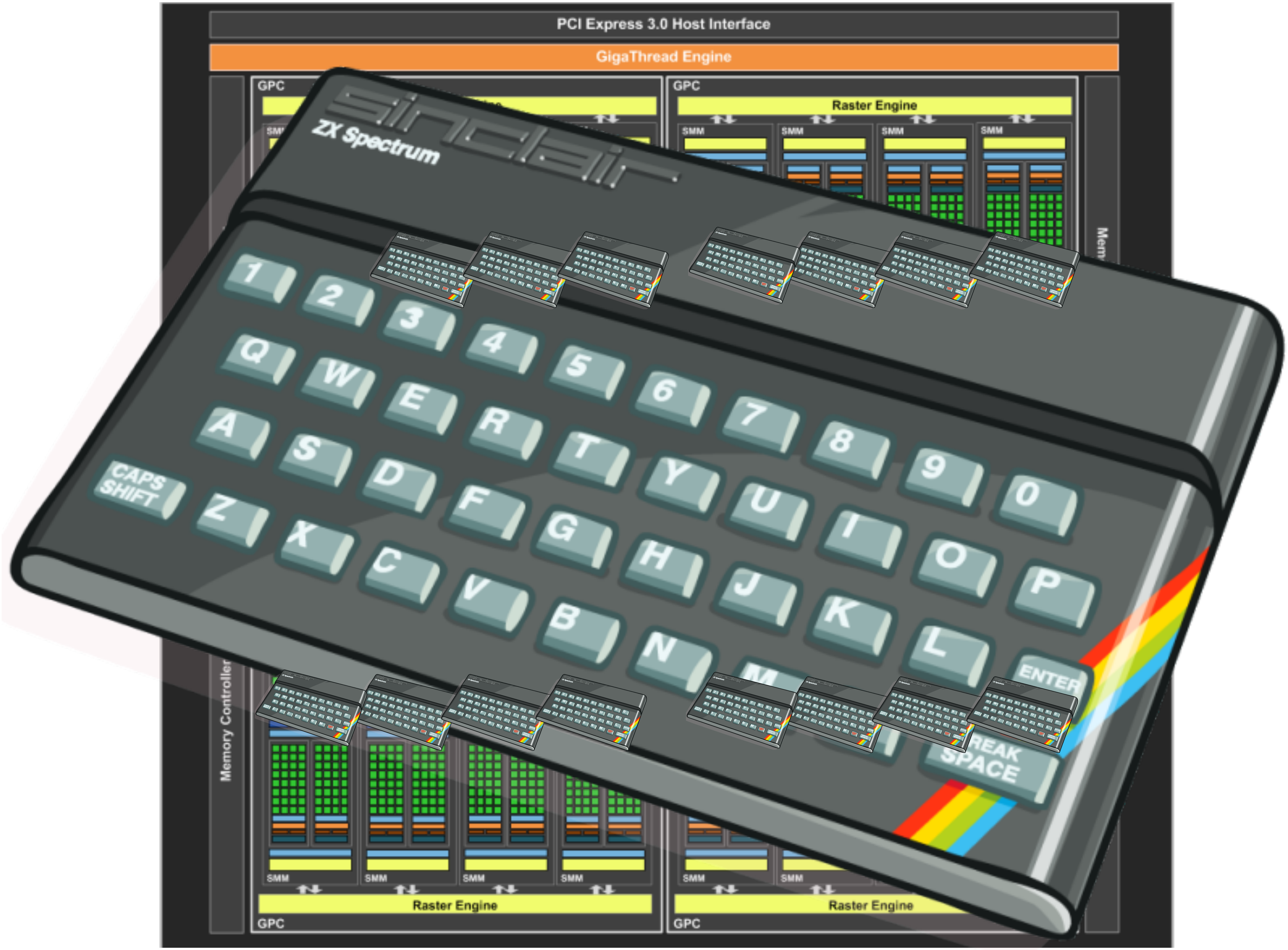
https://pyg.org/

pytorch scatter

```
#define ATOMIC(NAME)                                                          \
  template <typename scalar, size_t size> struct Atomic##NAME##IntegerImpl;   \
                                                                              \
  template <typename scalar> struct Atomic##NAME##IntegerImpl<scalar, 1> {    \
    inline __device__ void operator()(scalar *address, scalar val) {          \
      uint32_t *address_as_ui = (uint32_t *)(address - ((size_t)address & 3)); \
      uint32_t old = *address_as_ui;                                          \
      uint32_t shift = ((size_t)address & 3) * 8;                             \
      uint32_t sum;                                                           \
      uint32_t assumed;                                                       \
                                                                              \
      do {                                                                    \
        assumed = old;                                                        \
        sum = OP(val, scalar((old >> shift) & 0xff));                         \
        old = (old & ~(0x000000ff << shift)) | (sum << shift);                \
        old = atomicCAS(address_as_ui, assumed, old);                         \
      } while (assumed != old);                                               \
    }                                                                         \
  };                                                                          \
                                                                              \
```




PCI Express 3.0 Host Interface
GigaThread Engine
GPC
Raster Engine
SMM
Memory Controller
L2 Cache


PCI Express 3.0 Host Interface
GigaThread Engine
GPC
Raster Engine
SMM
Memory Controller
sinclair
ZX Spectrum
CAPS SHIFT
ENTER

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

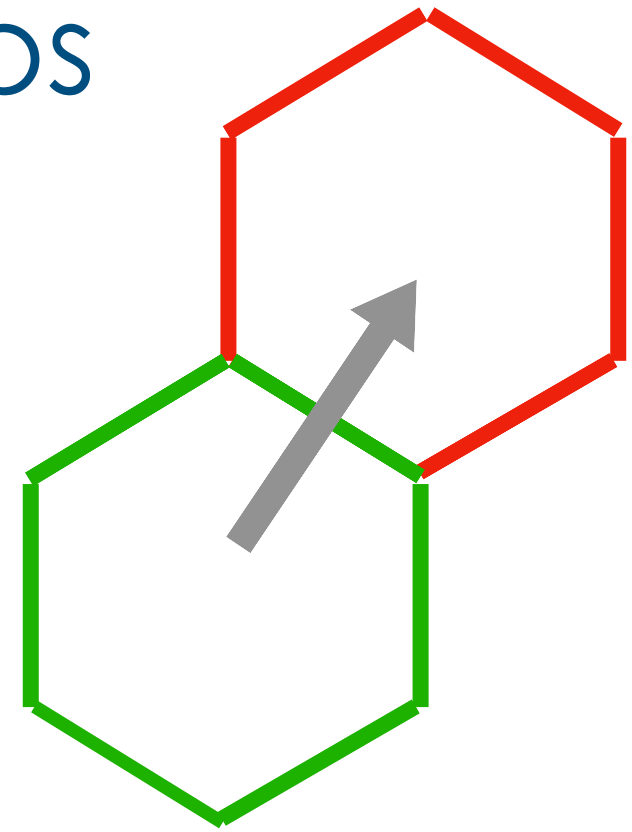
	ZINC-12K MAE(% ↓)	ZINC-Full MAE(% ↓)	OGBG-MOLHIV ROC-AUC(% ↑)	TOX21 ROC-AUC(% ↑)
RP-NGF (Murphy et al., 2019)	–	–	–	$0.79.4 \pm 1.00$
GCN (Kipf and Welling, 2017)	0.321 ± 0.009	–	76.07 ± 0.97	–
GIN (Xu et al., 2018)	0.408 ± 0.008	0.088 ± 0.002	75.58 ± 1.40	–
GINE (Hu et al., 2019)	0.252 ± 0.014	0.088 ± 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	$\mathbf{87.36 \pm 0.50}$
CIN (Bodnar et al., 2021a)	0.079 ± 0.006	$\mathbf{0.022 \pm 0.002}$	$\mathbf{80.94 \pm 0.57}$	–
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)	$\mathbf{0.075 \pm 0.003}$	**0.024**	$\mathbf{80.47 \pm 0.87}$	84.95 ± 0.58

[Hands, Sun & K, AISTATS 2024]

Schur Nets

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

Automorphism groups

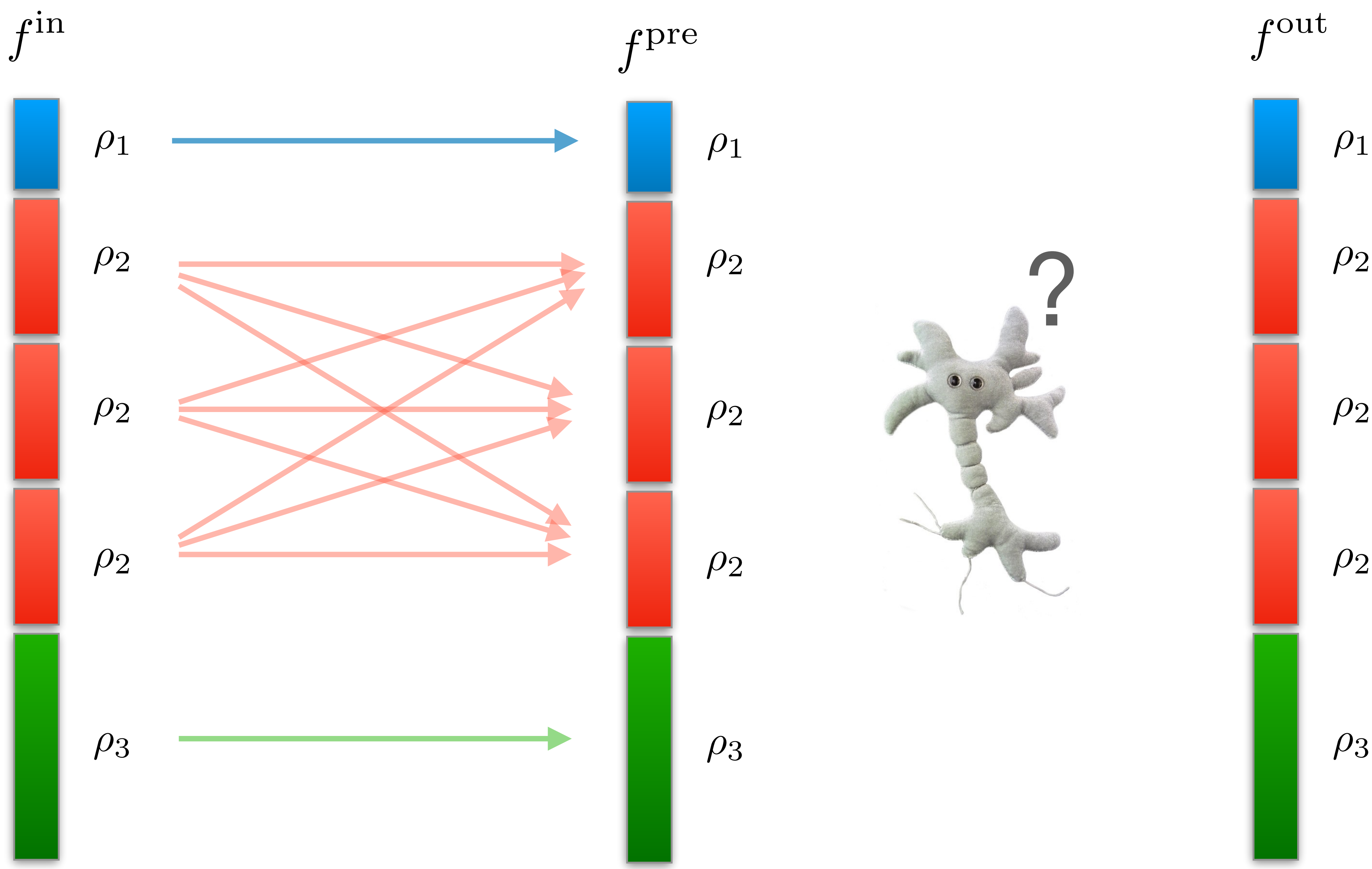


The automorphism group $\mathrm{Aut}(\mathscr{G})$ of a graph is the subgroup of permutations that leave the adjacency matrix fixed:

$$\sigma \circ A = A \qquad \Longleftrightarrow \qquad \sigma \in \mathrm{Aut}(\mathcal{G})$$

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.*

Isotopic subspaces

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

$$U_1^1 \oplus U_1^2 \oplus \ldots \oplus U_1^{\tau(1)}$$

Irreducible subspaces

Bare bones harmonic analysis

Ultimately we only need *some* decomposition into invariant subspaces

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

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)

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

Learnable weights

[Schur Nets, NeurIPS 2024]

Graph	Aut_S	**# of distinct Eigenvalues (*Schur* Layer)**	$\sum_i(\kappa_i)^2$ ***irreps* 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