Qualcomm

First Italian GDL Summer School 2022-07-27

From Equivariance to Naturality

Pim de Haan

Reseach Associate @ Qualcomm Technologies Netherlands B.V.

PhD student @ University of Amsterdam

pim@qti.qualcomm.com

Why Equivariance?

- Symmetry transformations relate “equivalent” descriptions of an object
- Equivariant maps respect symmetries, i.e. process equivalent inputs in essentially the same way
  - Fundamental idea of GDL
    - Bronstein, Bruna, Cohen, Velickovic, *Geometric Deep Learning: Grids, Groups, Graphs, Geodesics, and Gauges*, 2021
- Symmetry groups have some particular properties:
  - Invertibility: no information is lost; transformations can be undone
  - Composability: *any* two symmetry transformations can be composed
- In many cases we may want to loosen these assumptions
  - → This leads us to Categories, Functors, and Natural Transformations
  - → Generalizations of Groups, Representations, and Equivariant Maps
  - → Potential for less restricted network architectures & new application areas

“This may be regarded as a continuation of the Klein Erlanger Program, in the sense that a geometrical space with its group of transformations is generalized to a category with its algebra of mappings.”

S. Eilenberg & S. Mac Lane *A General Theory of Natural Equivalences* (1945, first paper on categories)

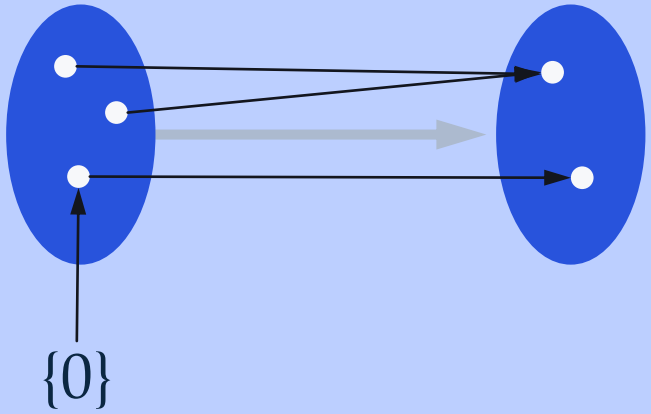
First example: the category of sets & mappings

- Categories consist of objects and arrows / maps / morphisms
- Arrows f : A → B with appropriate domain/codomain can be composed

The category only encodes how maps compose; doesn't care what objects/maps are "made of"

Set

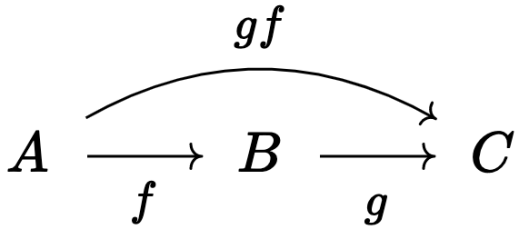
Ob(Set) = "all sets"
Hom(Set) = "all functions"



Lawvere, *Sets for Mathematics*

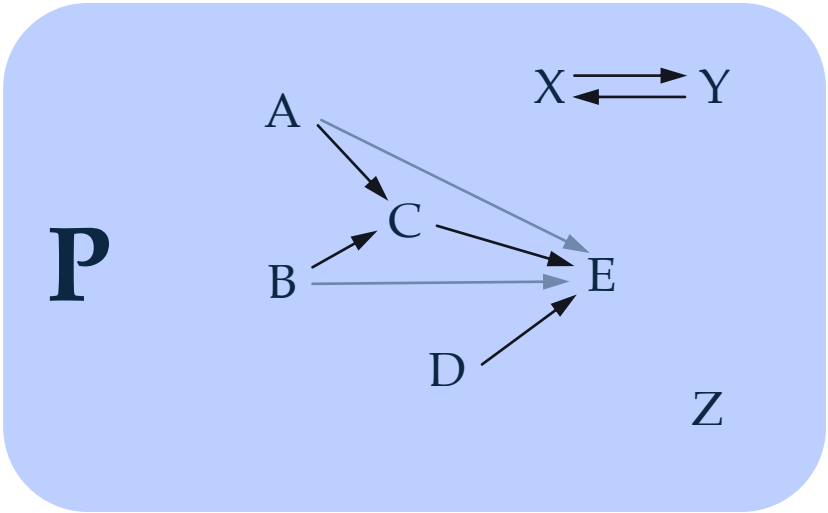
General Definition of Category

- A Category $\mathbb{C}$ consists of Objects and Arrows, such that:
  - For any object A there is an identity map $id_A : A \to A$
  - For any two arrows $f : A \to B$ and $g : B \to C$, there is a composite arrow $g\,f : A \to C$
    - A is called the domain of f, and B the codomain
  - Such that id acts as a unit: $f\,id_A = id_B\,f = f$ (for all f)
  - Composition is associative: $(f\,g)\,h = f\,(g\,h)$



- Examples:
  - Categories of mathematical gadgets: Set, Vect, Top, Grp
  - Mathematical gadgets as categories: Groups, Groupoids, Preorders,
  - Applied category theory: Resource theories, Markov Categories, Bayes Nets, Causal Theories, Monoidal Categories, Chemical Reaction Networks, …

Example: Preorders as Categories



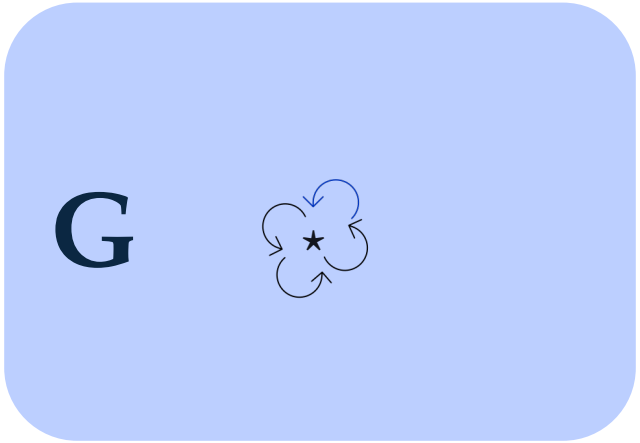
- Gadget as a category
- Classical definition: a relation A ≤ B that is:
  - Reflexive: A ≤ A
  - Transitive: A ≤ B and B ≤ C then A ≤ C
    - (Not necessarily antisymmetric as in a partial order; A ≤ B and B ≤ A then not necessarily A = B)
- Examples:
  - Reachability in graphs, ordering of numbers, ordering of subsets, …
- Categorical definition: a category with at most one arrow from any A to any B
  - Interpretation: we have A ≤ B if and only if there is an arrow A → B
  - ✅ Reflexive: in any category we have an identity arrow A → A
  - ✅ Transitive: the composition operation takes arrows A → B and B → C and produces an arrow A → C
    - Since a preorder category has at most one arrow A → C there is nothing to choose

Example: Groups as Categories

- Isomorphism: a map f : A → B in a category is called an isomorphism if there exists g : B → A s.t.

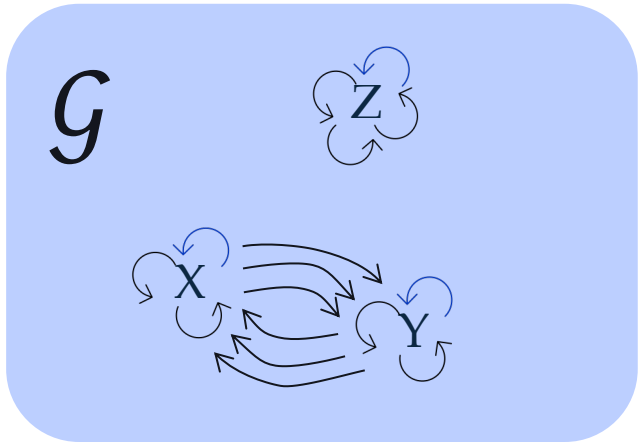
$$g\ f = id_A \quad \text{and} \quad f\ g = id_B$$

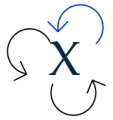
- Classical definition: a set with an associative binary relation, identity maps, and inverses (…)
- Categorical definition: a category with one object $\star$, where each morphism $g: \star \to \star$ is an isomorphism
  - ✅ All arows $g: \star \to \star$ and $\mathrm{h}: \star \to \star$ can be composed and composition is associative
  - ✅ G has an identity $\mathrm{id}_\star: \star \to \star$
  - ✅ Since all arrows are iso, we have inverses



Groupoids

- A Groupoid $\mathcal{G}$ is a category where every morphism is an isomorphism
  - A group is a groupoid with one object
- Example: the category of graphs, restricted to only contain graph isomorphisms
- A groupoid has two kinds of maps: general isomorphisms A → B and automorphisms A → A
- Automorphisms are symmetries. The automorphisms of an object X form a group
- Isomorphism classes are connected by isomorphisms: $\{\{X, Y\}, \{Z\}\}$



$Aut_{\mathcal{G}}(X) =$ 

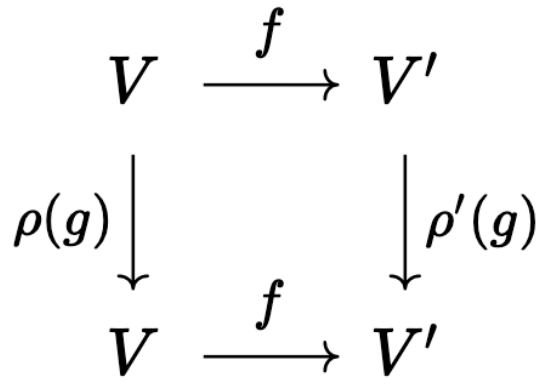
Group Actions & Representations as Functors

- A group action F of a group G on a set X,
  - Has for each element $g \in G$, a bijection $F(g): X \to X$
  - Respecting group multiplication: $F(g\ g') = F(g) \circ F(g')$
- A group representation ρ of a group G is a linear action on a vector space V
- A functor $F: C \to D$ between categories is:
  - A mapping on objects $F: Ob(C) \to Ob(D)$
  - A mapping on morphisms that takes $f: A \to B \in C$ to $F(f): F(A) \to F(B) \in D$
  - Such that $F(\mathrm{id}_X) = \mathrm{id}_{F(X)}, \mathrm{F}(f \circ g) = F(f) \circ F(g)$
- Examples:
  - A functor between groups is a group homomorphism
  - A functor between preorders is a monotone map
  - A group action is a functor $\rho: G \to Set, \quad \rho(\star) = X, \ \rho(g): X \to X$
  - A group representation is a functor $\rho: G \to Vec, \ \rho(\star) = V, \ \rho(g): V \to V$

Equivariant Maps as Natural Transformations

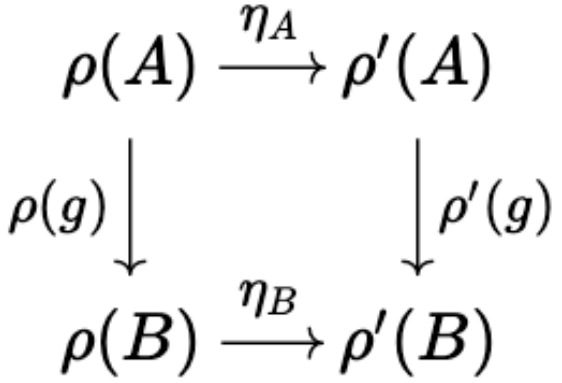
- An equivariant linear map $f: V \to V'$ between group representations $(\rho, V), (\rho', V')$ is a map satisfying for each $g \in G$:

$$f \circ \rho(g) = \rho'(g) \circ f$$



- A Natural Transformation η between Functors $\rho, \rho': C \to D$
  - Written $\eta: \rho \Rightarrow \rho'$
  - For each *object* A of C, a mapping $\eta_A: \rho(A) \to \rho'(A)$ such that for every *arrow* $g : A \to B$ in C:

$$\eta_B \circ \rho(g) = \rho'(g) \circ \eta_A$$

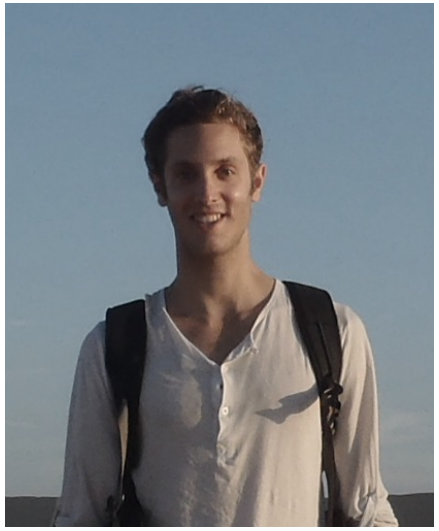


- Hence, an equivariant map is a natural transformation between functors $\rho, \rho': G \to Vec$:
  - For the one and only object $\star$, $\rho(\star) = V, \rho'(\star) = V'$
  - Single linear mapping $f = \eta_\star: \rho(\star) \to \rho'(\star)$, such that tor every $g: \star \to \star$ in G:

Natural Graph Networks

Team

Pim de Haan
Qualcomm AI Research
Qualcomm Technologies Netherlands B.V.
University of Amsterdam



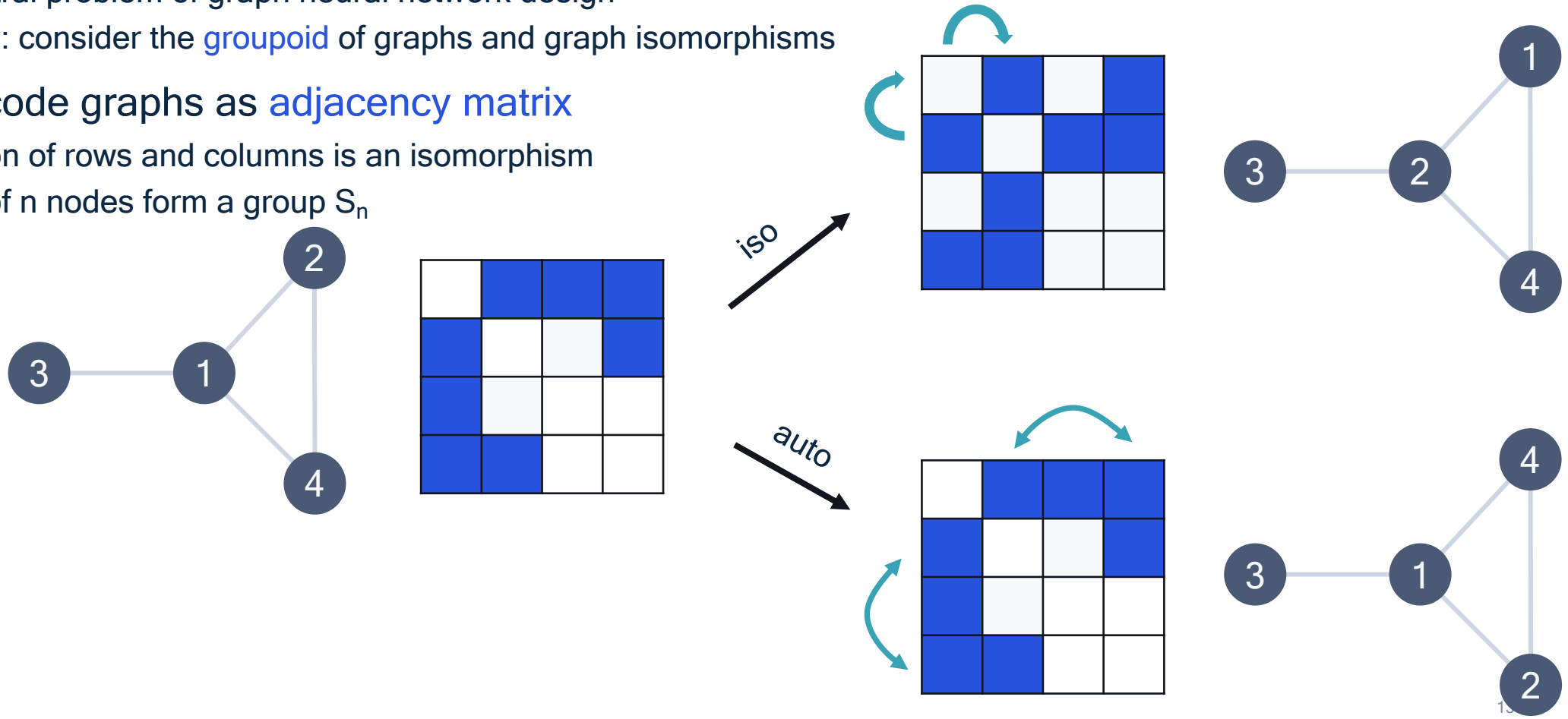
Taco Cohen
Qualcomm AI Research
Qualcomm Technologies Netherlands B.V.

Max Welling
Work done while at
Qualcomm AI Research
QUVA, University of Amsterdam
CIFAR
Qualcomm Technologies Netherlands B.V.

https://arxiv.org/abs/2007.08349

Graph Iso- and automorphisms

- There are many ways to encode the same graph!
  - All of these should be treated in an equivalent manner by the neural network
  - This is the central problem of graph neural network design
  - Mathematically: consider the groupoid of graphs and graph isomorphisms
- Example: encode graphs as adjacency matrix
  - Any permutation of rows and columns is an isomorphism
  - Permutations of n nodes form a group S_n



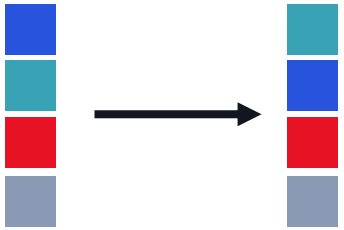
Graph Features

Group / groupoid representations

- Graph feature == representation of S_n
  - S_n is the group of permutations of n nodes

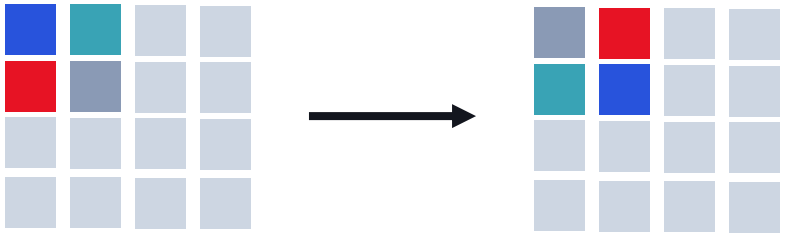
$$\rho_0(P_{12})s = 1 \cdot s$$

Scalar feature
e.g network output in graph classification



$$\rho_1(P_{12})v = P_{12}v$$

Vector feature
e.g. one feature per node



$$\rho_2(P_{12})M = P_{12}MP_{12}^T$$

Tensor feature
e.g. one feature per edge or node pair

Equivariant Graph Networks

- Common approach[1,2,3]: interpret adjacency matrix as a 2nd order graph feature, stack with other features along channel dim, and use this as input to the network:

$$\Phi([G, f]) = f'$$

- Network should be equivariant:

$$\Phi(\rho_{\text{in}}(P)[G, f]) = \rho_{\text{out}}(P)\Phi([G, f])$$

- For linear maps:
  - Order 1 -> 0 (Invariant Deep Sets): 1 parameter
  - Order 1 -> 1 (Equivariant Deep Sets): 2 parameters
  - Order 2 -> 2 (PPGN [1]): 15 parameters
- Not a lot of parameters!

[1] Provably Powerful Graph Networks, Haggai Maron*, Heli Ben-Hamu*, Hadar Serviansky*, Yaron Lipman (*equal contribution), *NeurIPS 2019*
[2] Invariant and Equivariant Graph Networks, Haggai Maron, Heli Ben-Hamu, Nadav Shamir and Yaron Lipman, *ICLR 2019*
[3] Incidence Networks for Geometric Deep Learning, Albooyeh, Marjan, Bertolini, Daniele, and *Ravanbakhsh, ICML 2020*

Natural Graph Networks: global version

- Groupoid $Graph$

- Groupoid representation $\rho: Graph \to Vec$, e.g.
  - Node feature $\rho(G) = \mathbb{R}^n$,
  - Permutation $\rho(g): \mathbb{R}^n \to \mathbb{R}^n$

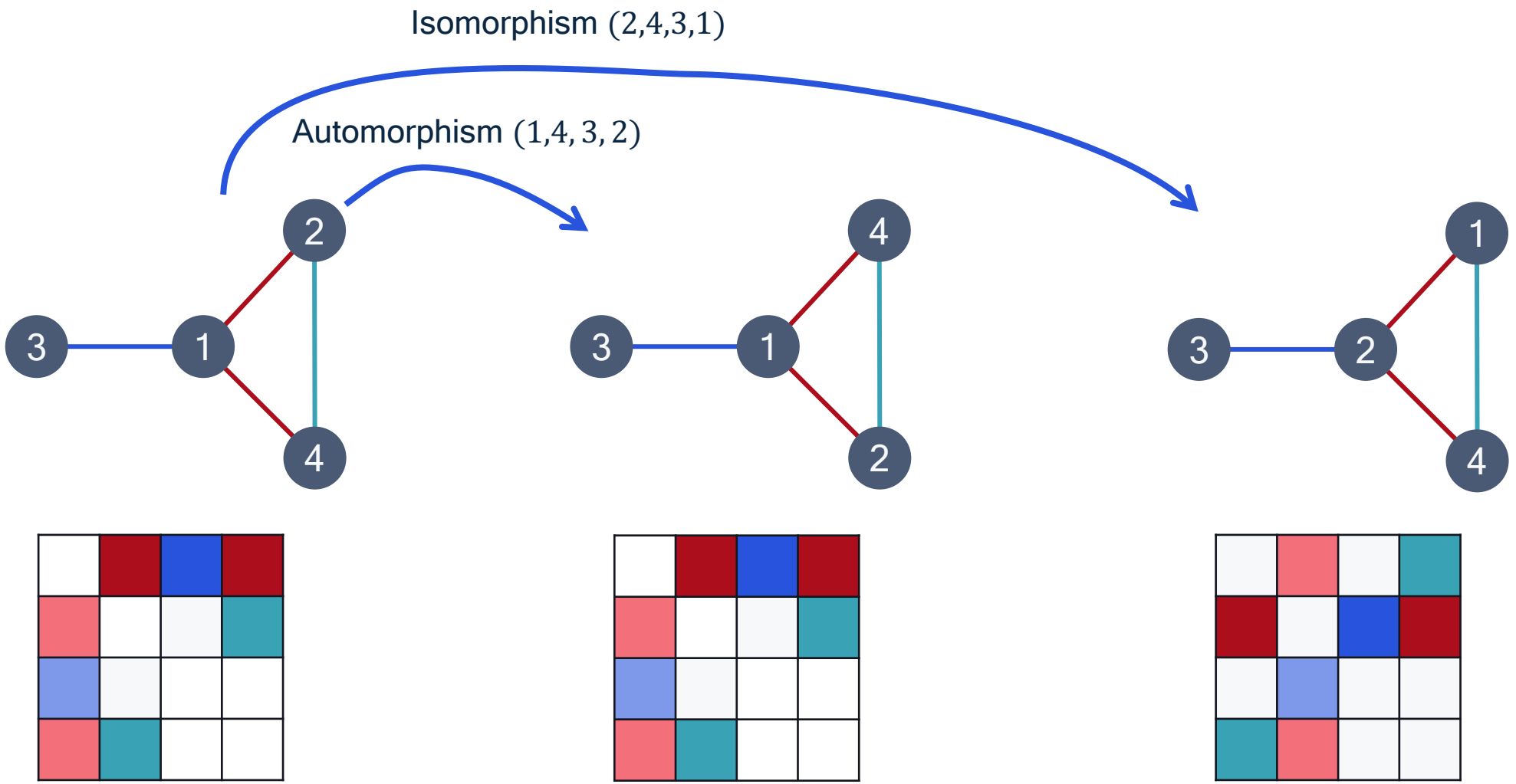
- Natural transformation $\eta: \rho_{\text{in}} \Rightarrow \rho_{\text{out}}$, contains
  - For each graph $G \in Ob(Graph)$, a map $\eta_G: \rho_{\text{in}}(G) \to \rho_{\text{out}}(G)$
  - Such that

- Isomorphism gives weight sharing
  - Non-isomorphic graphs untied weights
- Automorphism gives constraint
  - Non-symmetric graph unconstrained

$$
\begin{array}{ccc}
\rho_{\text{in}}(G) & \xrightarrow{\eta_G} & \rho_{\text{out}}(G) \\
\text{auto}\uparrow & & \uparrow\text{auto} \\
\rho_{\text{in}}(G) & \xrightarrow{\eta_G} & \rho_{\text{out}}(G) \\
\text{iso}\downarrow & & \downarrow\text{iso} \\
\rho_{\text{in}}(G') & \xrightarrow{\eta_{G'}} & \rho_{\text{out}}(G') \\
\\
\rho_{\text{in}}(G'') & \xrightarrow{\eta_{G''}} & \rho_{\text{out}}(G'')
\end{array}
$$

Natural Graph Networks: example

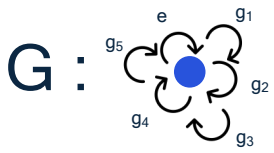
Message passing - 7 parameters



Equivariant vs Natural Networks

Equivariant Graph Networks

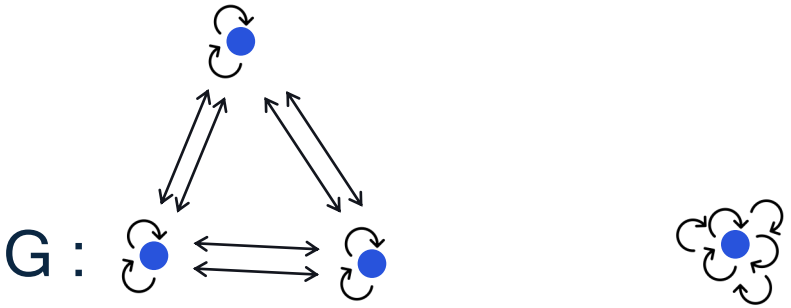
- Symmetry group G = S_n



- Feature space: Functor G → Vect
  - (Group representation)
- Network layer: natural transformation
  - (Equivariant map)
  - One constraint per permutation
  - The same map for every graph

Natural Graph Networks

- Symmetry groupoid G = {adjacency mats} // S_n
  - Called the action groupoid



- Feature space: Functor G → Vect
  - (Groupoid representation)
- Network layer: natural transformation
  - One constraint per *automorphism*
  - Different maps for non-isomorphic graphs

...

Going local

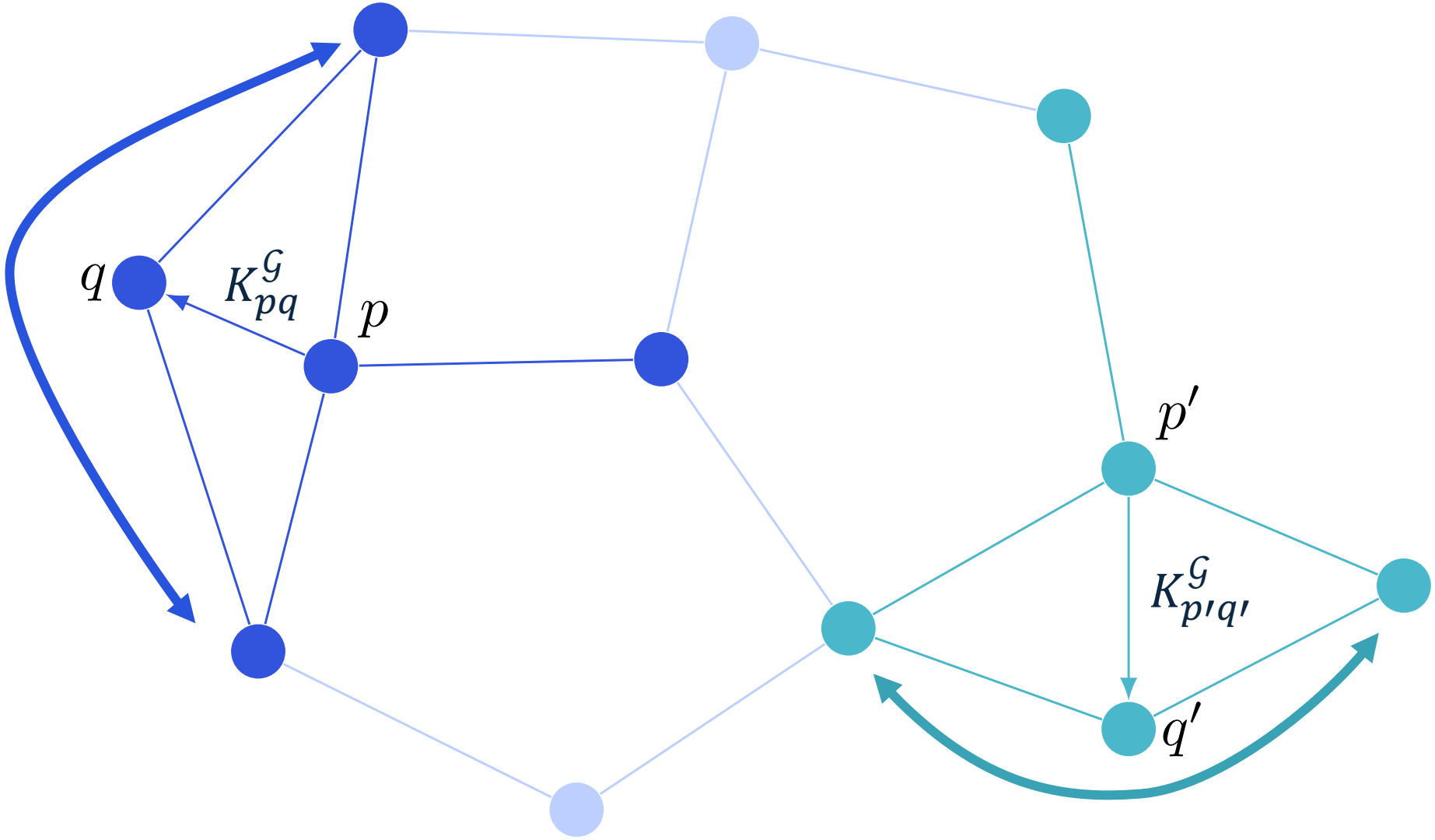
- Global methods are not scalable to large graphs
- Vanilla NGNs do not generalize across non-isomorphic graphs

Natural Graph Networks

Local version

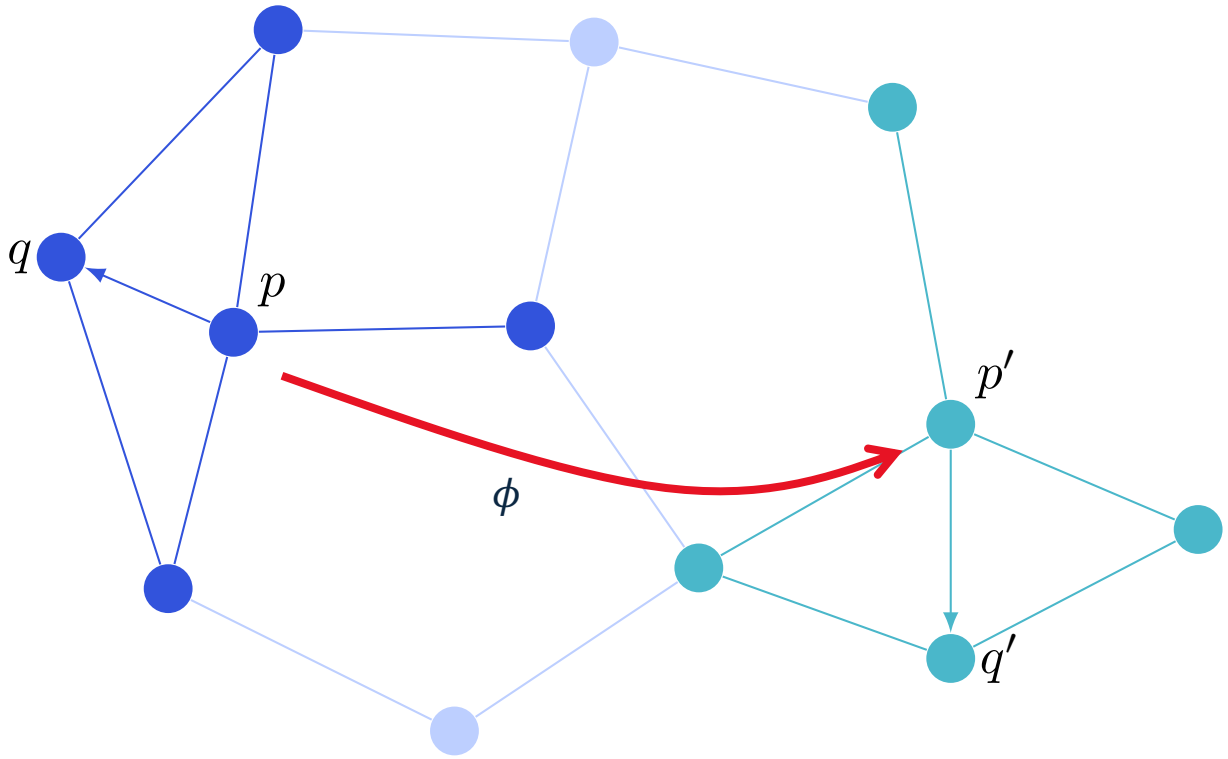
- For each node and edge, define a neighbourhood
  - Edge neigbourhoods should include the node neighbourhoods of the start and end node
  - E.g. one-hop neighbourhood
- Neighbourhood isomorphisms give node and graph isomorphisms
- To each node, attach a graph feature
  - I.e. a representation of S_n where n is the size of the node neighbourhood
  - Typically we use a vector feature (i.e. one number per neigbour)
- Constraints:
  - Weight matrix for an edge is constrained by the automorphisms of the edge neighbourhood
  - Weight matrices on isomorphic edges share weights

Natural Message Passing - local version



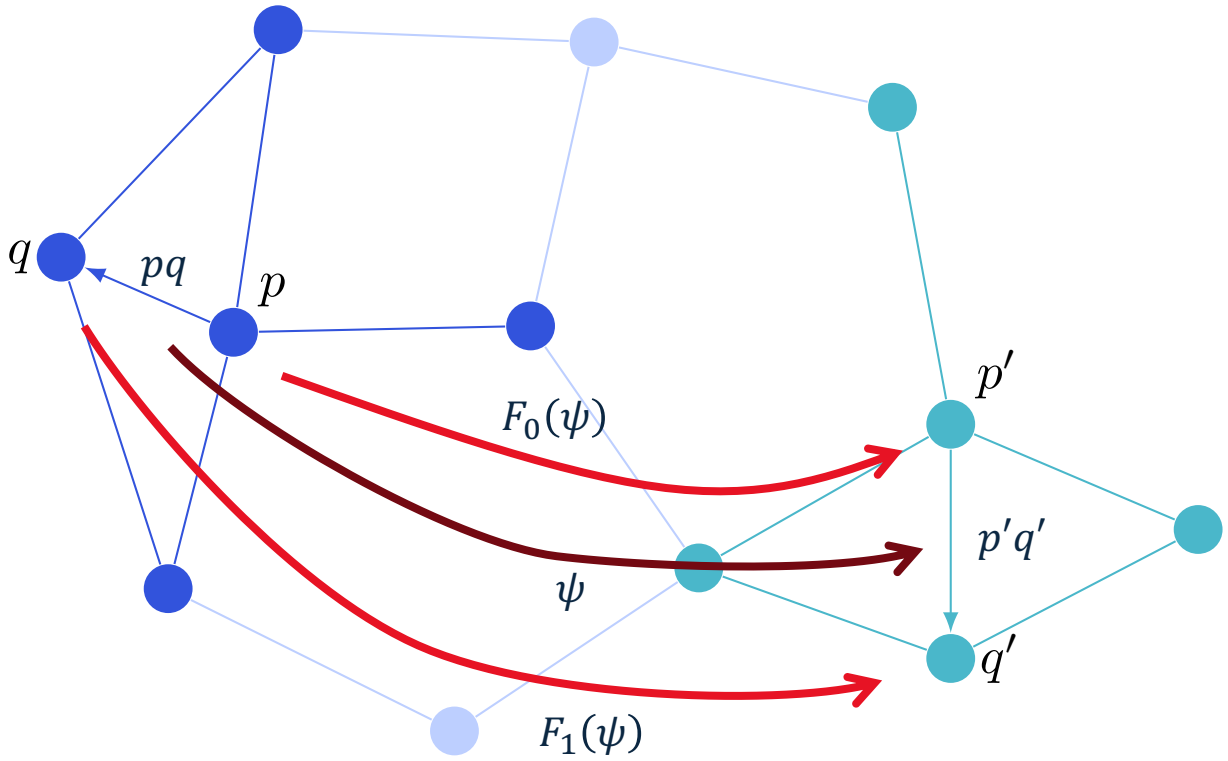
Node groupoid

- For each node p find neighbourhood N_p
- Graph isomorphism ϕ: $N_p \to N_{p'}$
- Node groupoid $\mathcal{N}$
  - Objects are nodes p
  - Morphisms are neighbourhood isomorphisms $\phi: p \to p'$



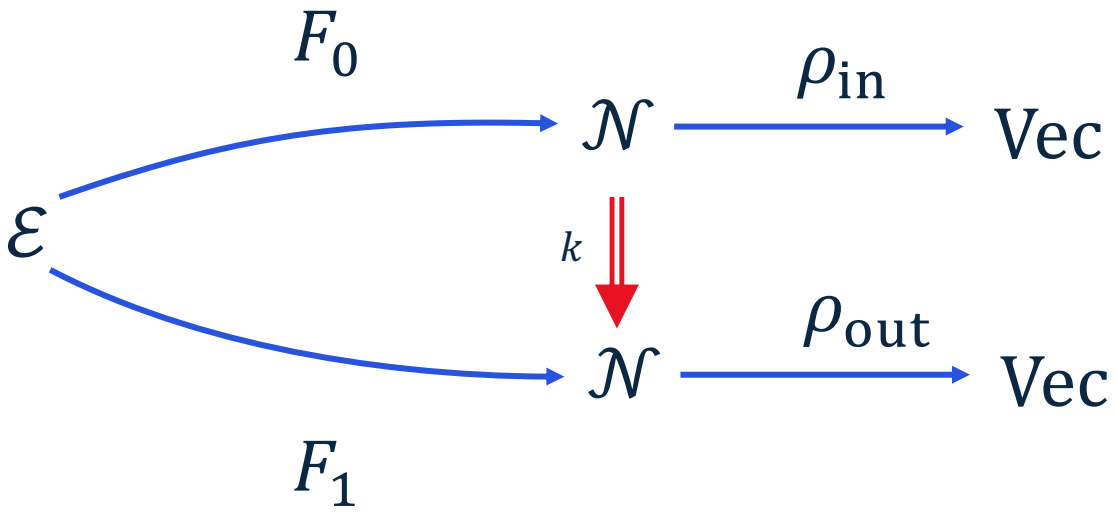
Edge groupoid

- For each edge pq find neighbourhood N_{pq}
- Graph isomorphism ψ: $N_{pq} \to N_{p'q'}$
- Edge groupoid $\mathcal{E}$
  - Objects are edge pq
  - Morphisms are neighbourhood isomorphisms $\psi: pq \to p'q'$
- Edge/node correspondence:
  - Edge pq has start node p
  - Edge isomorphism $\psi: pq \to p'q'$ has start node isomorphism $p \to p'$
- Functor $F_0: \mathcal{E} \to \mathcal{N}$
  - Maps $F_0(pq) = p$
  - Maps $F_0(\psi) = p \to p'$
- Similar for tail node: $F_1: \mathcal{E} \to \mathcal{N}$



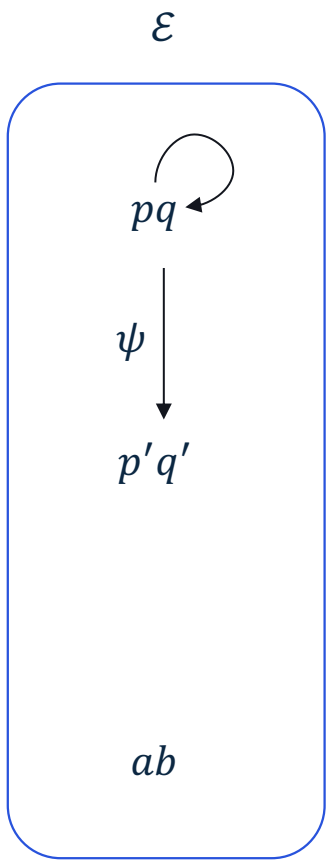
Natural Graph Network - abstractly

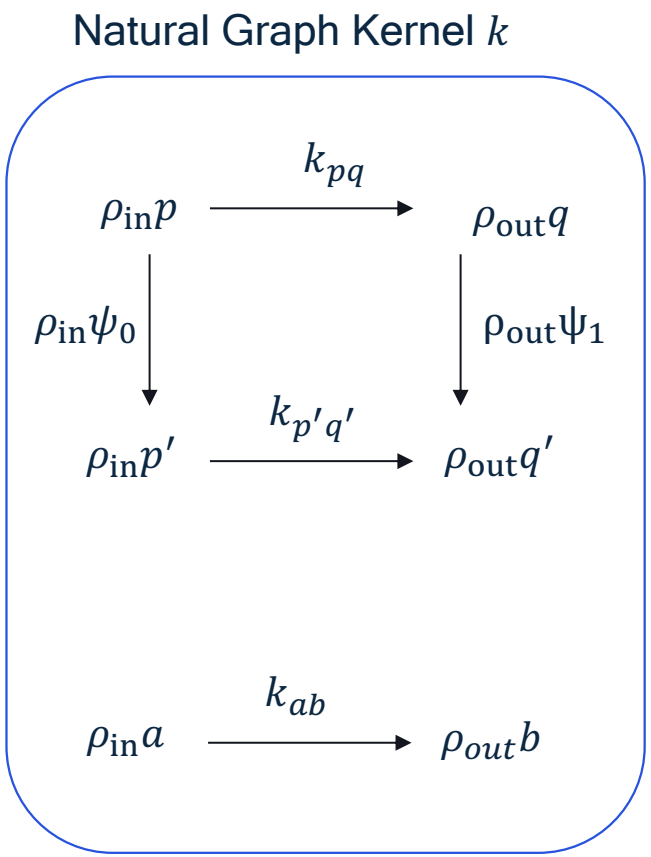
- Node features $\rho: \mathcal{N} \to Vec$
  - $\rho(p)$ is graph feature of neighbourhood N_p, e.g. $\mathbb{R}^{n_p}$
  - Node iso $\phi: p \to p'$ becomes linear transformation $\rho(\phi): \rho(p) \to \rho(p')$, e.g. permutation
- Compose functors



- Message passing kernel is natural transformation $k: \rho_{\text{in}} \circ F_0 \Rightarrow \rho_{\text{out}} \circ F_1$

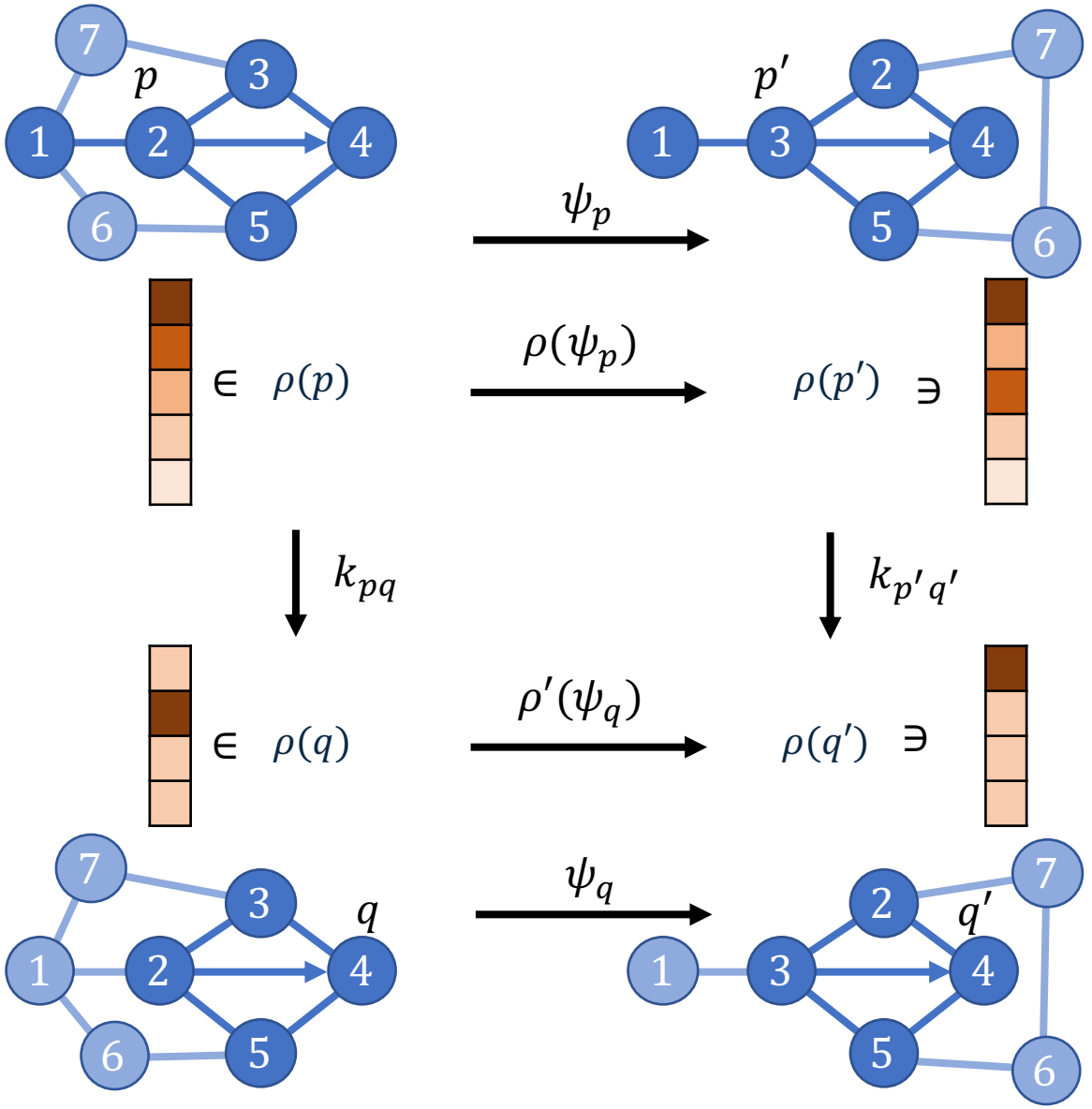
Natural Graph Networks





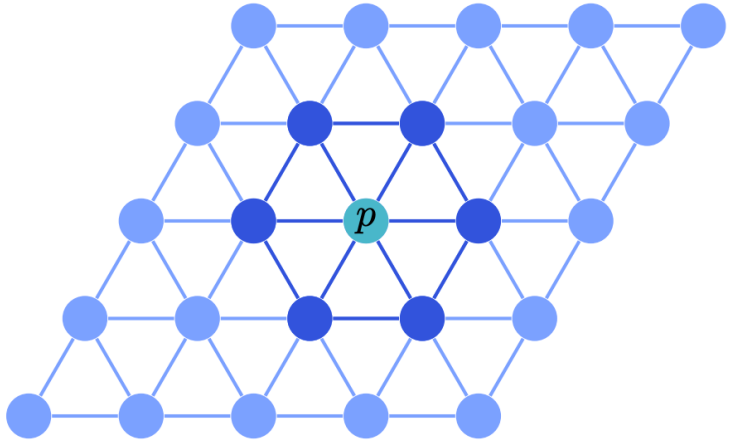
- Isomorphism = weight sharing
- Automorphism = constraints
- Linear kernel: solve linear constraints
- Layer: $v'_q = \sum_{e=(p,q)} k_{pq}(v_p)$

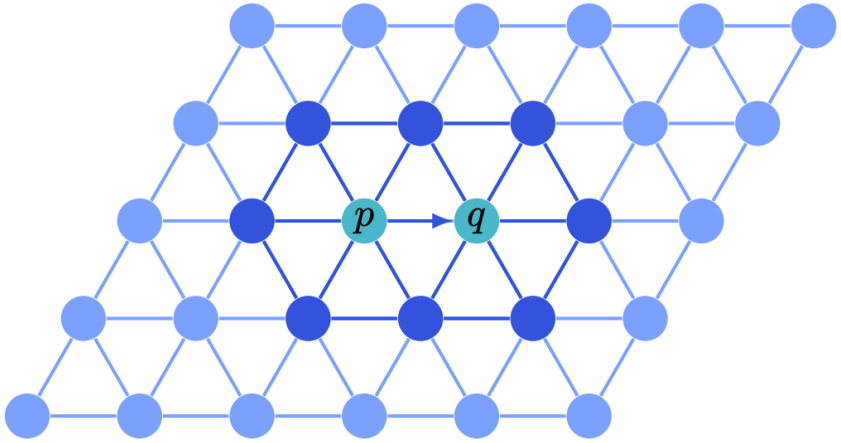
Local Natural Graph Network



Reduction to Group / Manifold Equivariance

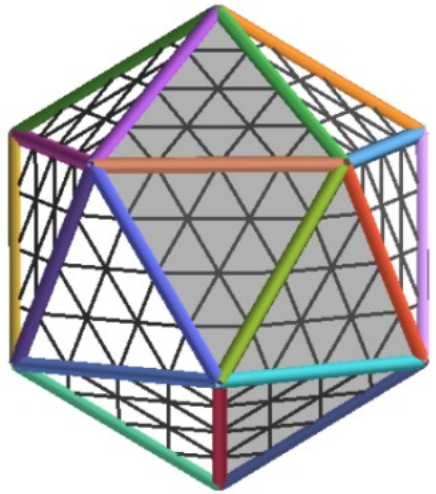
- If graph is a grid
- Node category $\mathcal{N}$
  - Objects are elements of grid
  - Automorphisms are D_4
    - Rotations
    - Mirrors
- Representation $\rho: \mathcal{N} \to Vec$, equivalent to D_4 representation
  - E.g. regular representation $\mathbb{R}^4$
- Edge automorphisms: mirror
- Equivalent to D_4-CNN
  - Cohen & Welling (2016): Group Equivariant CNNs





Equivalent to coordinate free CNN

- Graph = grid on manifold
- Equivalent to Icosahedral CNN
  - Cohen, Weiler, Kicanaoglu, Welling et al (2019): Gauge Equivariant CNNs
- Except at corners: these should have different weights



Basic Algorithm

- Precompute:

1. Define node and edge neighbourhoods
2. Classify edge neighbourhood isomorphism classes
3. Compute edge automorphisms
4. Solve kernel constraint, initialise params

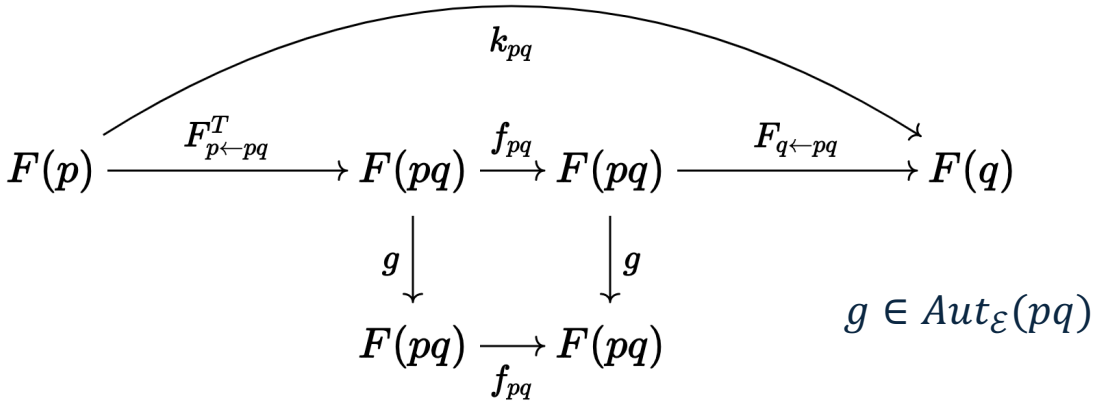
- During training:

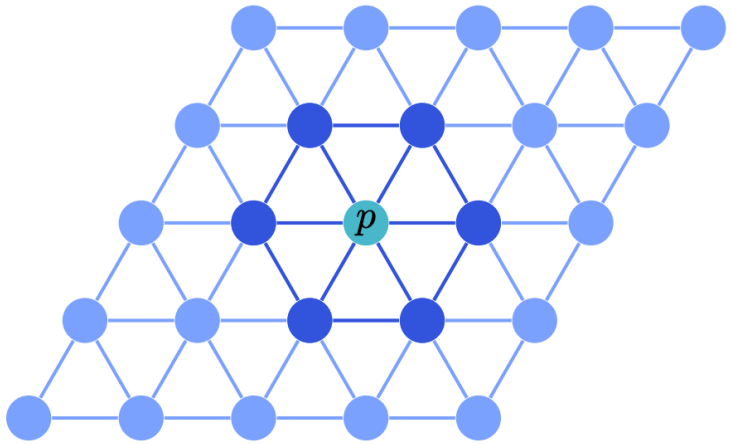
1. Linearly combine kernel solutions using parameters
2. Transport kernels by isomorphisms
3. Compute convolution

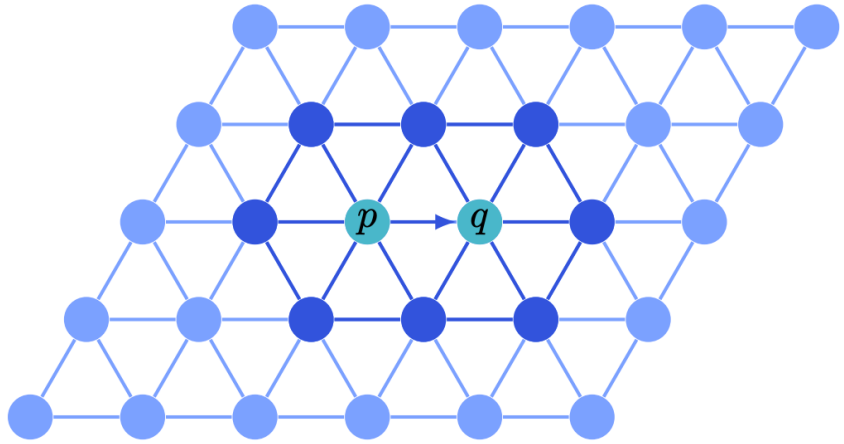
- Cost linear in number of edges, expensive in neighbourhood size

Connection to sheafs?

- Sheaf F
  - Node p, $F(p) =$ graph feature for N_p e.g. $\mathbb{R}^{n_p}$
    - $F(p) = \rho(p)$
  - Edge pq, $F(pq) =$ graph feature for N_{pq} e.g. $\mathbb{R}^{n_{pq}}$
  - $F_{p \leftarrow pq} : F(pq) \to F(p)$ projection
- $f_{pq} : F(pq) \to F(pq)$ graph network







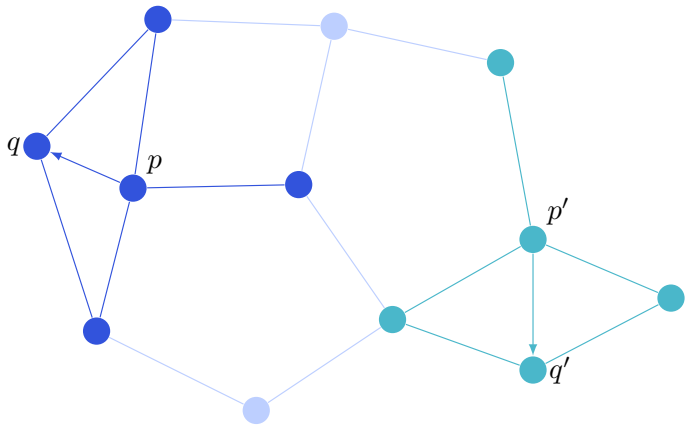
- f_{pq} “global” natural graph network ⇒ k_{pq} natural message passing

Challenges

- 1000s of edge iso classes
- How to find common parametrization for f_{pq}?
- Only treat common edges as non-trivial
- Canonization
  - Unique order for each graph – up to auto
  - One common $f: \mathbb{R}^n \to \mathbb{R}^n$
  - Symmetrize with automorphism group
- Make $f_{pq}(v)$ a graph neural net $\Phi(N_{pq}, v)$ - hypernetwork
  - E.g. graph CNN
- Only consider subgraphs with nice automorphism groups
  - Chains, cycles
  - Thiede et al, “Autobahn: Automorphism-based Graph Neural Nets” (2021)

Natural Graph Networks: Summary

- Graph networks must respect graph symmetries, & treat isomorphic graphs equivalently
- Graph symmetries = autmomorphisms ≠ permutation of nodes
- Network layer = natural transformation between functors
- Global NGN = natural transformation between graph symmetries
- Local NGN = natural transformation between edge symmetries
  - Induces Global NGN
  - More expressive message passing
- Exploiting local symmetries yields efficient and powerful graph networks

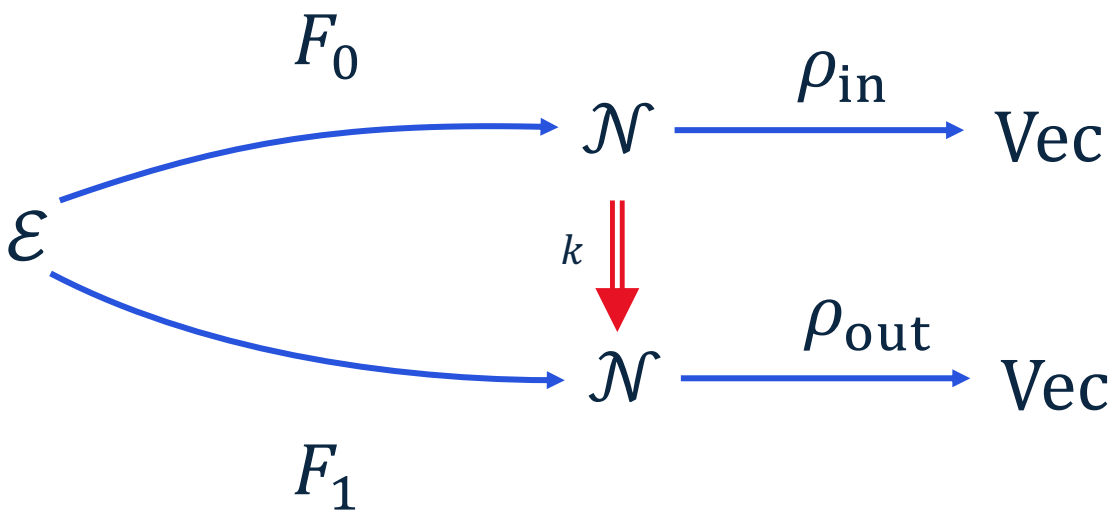


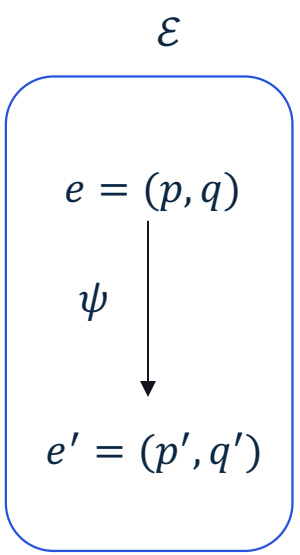
Euclidean Natural Message Passing

Applications to point clouds and meshes

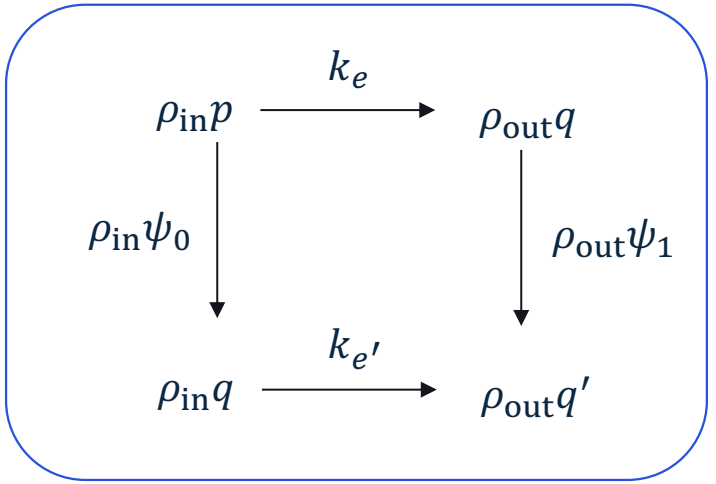
Work in progress

General Natural Message Passing



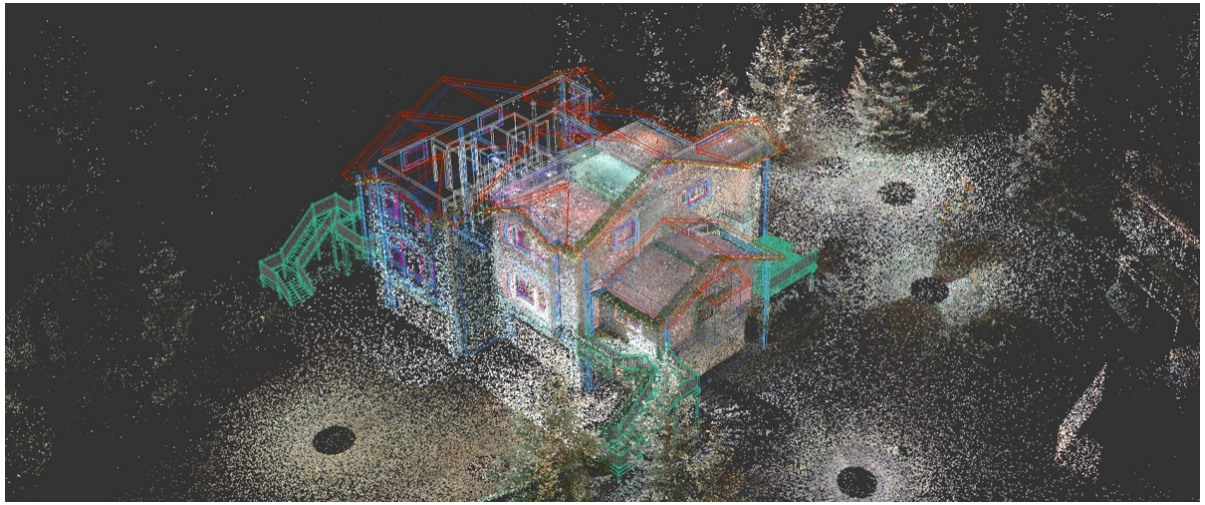


Natural Message Passing Kernel k



Point Clouds

- Set of n points in $\mathbb{R}^3$, data of shape $\mathbb{R}^{n \times 3}$
- Neural network $f: \mathbb{R}^{n \times 3} \to \mathbb{R}^{n \times d}$
- If cloud oriented arbitrarily, equivariant to
  - Translations
  - Permutations
  - Rotations
  - For all $g \in SE(3) \times S_n$, $f(gv) = gf(v)$
- Examples:
  - Molecule
  - Car pose
- Translation equivariance via differences
- Permutation equivariance via message passing
- Rotation equivariance: $R \in SO(3), f(Rv) = Rf(v)$



f

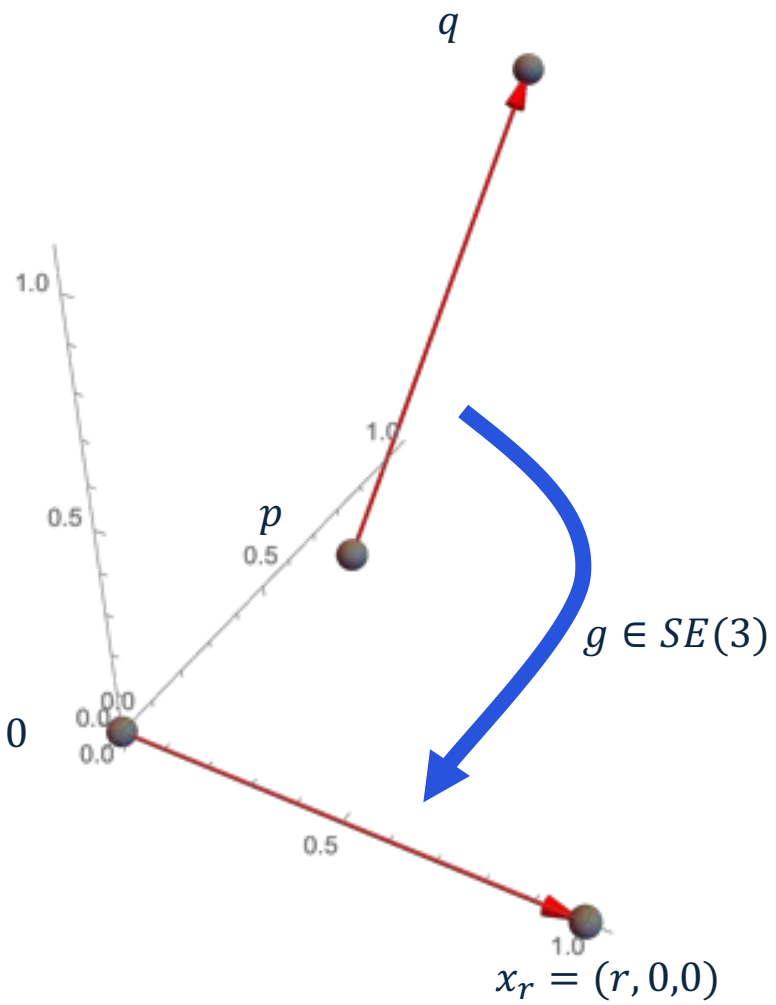
$\mathbb{R}^{n \times d}$

$SO(3)$ equivariant message passing

- Node features SO(3) representations $(\rho_{\text{in}}, \mathbb{R}^d)$, $(\rho_{\text{out}}, \mathbb{R}^{d'})$
  - Examples: $\mathbb{R}^1$ invariant; $\mathbb{R}^3$ 3-vector; $\mathbb{R}^9$ 3x3 matrix
- Message Network: $K: \mathbb{R}^3 \times \mathbb{R}^d \to \mathbb{R}^{d'}$
- Layer: $v'_q = \sum_{pq} K(x_q - x_p, v_p)$
- Equivariance constraint: $K(R(x_q - x_p), Rv_p) = RK(x_q - x_p, v_p)$
- Difficulties:
  - $SO(3)$ representation theory involves varying dimensions and Clebsch-Gordan coefficients. Difficult to implement cleanly and efficiently
  - The network is conditional on the direction, making it different for all edges. Additional computational cost

Natural Message Passing

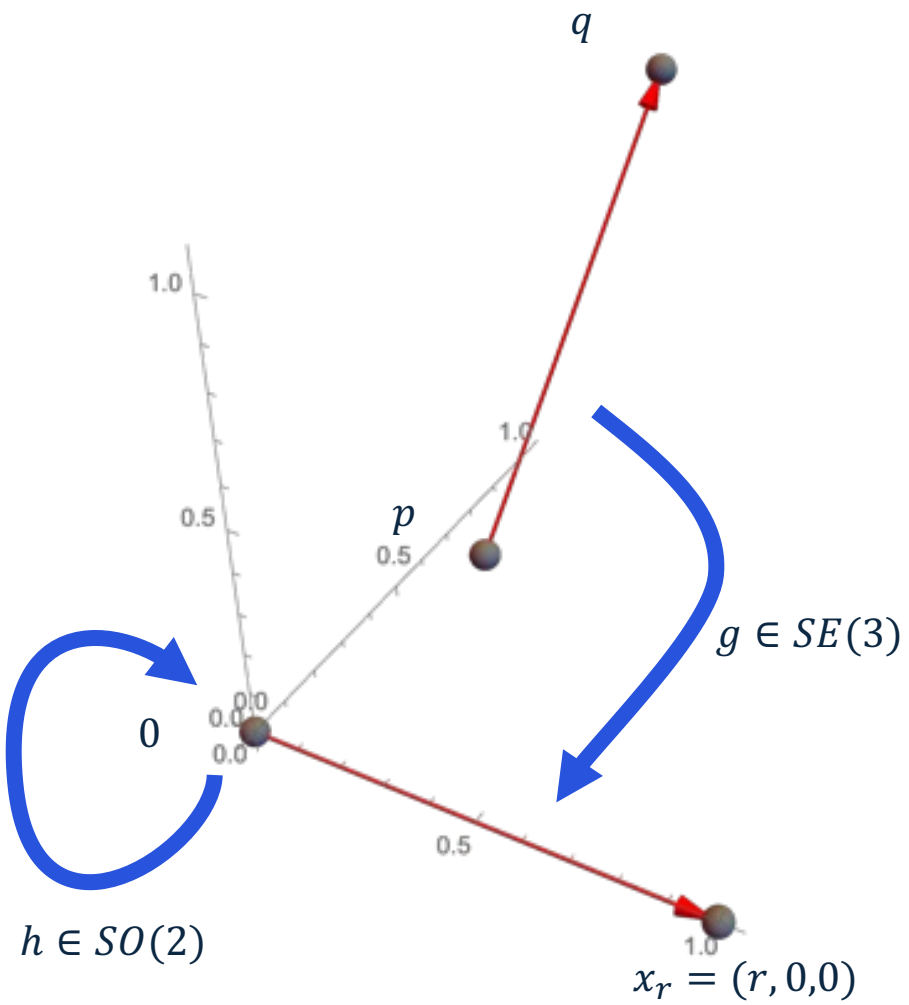
reduction of 3D symmetry to 2D symmetry of edges



- Node groupoid $\mathcal{N} = \mathbb{R}^3 \,//SE(3)$
  - Objects are points $p \in \mathbb{R}^3$
  - Isomorphisms $g_p: p \to p',\ g \in SE(3), g(p) = p'$
  - All points isomorphic
- Edge groupoid $\mathcal{E} = (\mathbb{R}^3 \times \mathbb{R}^3)//SE(3)$
  - Objects are pairs of points $p, q \in \mathbb{R}^3$
  - Isomorphisms $g_{pq}: pq \to p'q',\ g \in SE(3), g(p) = p', g(q) = q'$
  - Edges of same length isomorphic
- Node features $\rho: \mathcal{N} \to Vec$ is $SO(3)$ representation
  - $\rho(p) = \mathbb{R}^d$
  - $\rho(g_p) = \rho(g)$
- Natural Message Passing $k: \rho_{\text{in}} \circ F_0 \Rightarrow \rho_{\text{out}} \circ F_1$

Natural Message Passing

reduction of 3D symmetry to 2D symmetry of edges



- Automorphisms: $SO(2)$ aroud x axis

Natural Graph Kernel k

$$\begin{array}{ccc} \mathbb{R}^d & \xrightarrow{k_{pq}} & \mathbb{R}^{d'} \\ \downarrow \rho_{\text{in}} g & & \downarrow \rho_{\text{out}} g \\ \mathbb{R}^d & \xrightarrow{k_{0x_r}} & \mathbb{R}^{d'} \\ \downarrow \rho_{\text{in}} h & & \downarrow \rho_{\text{out}} h \\ \mathbb{R}^d & \xrightarrow{k_{0x_r}} & \rho_{out} b \end{array}$$

Algorithm

Precompute:

- For each edge (p, q) precompute $g \in SE(3)$ to move edge to $(0, (r, 0,0))$
- Construct $SO(2)$ radius-conditional network f_r: $\mathbb{R}^d \to \mathbb{R}^{d'}$

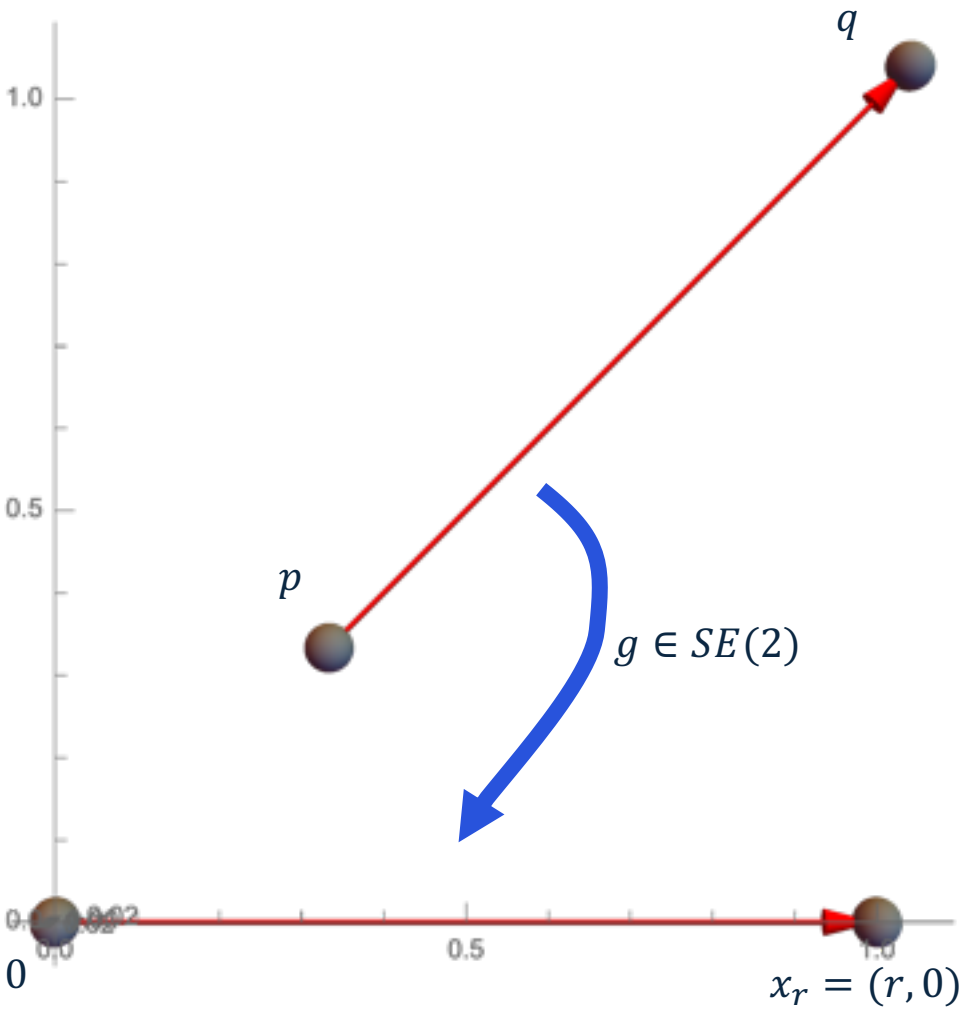
In network:

1. Move feature v from p to 0 with $SO(3)$ action g
2. Message pass from 0 to x_r with f_r
3. Move output feature from x_r to q with $SO(3)$ action g^{-1}
4. Sum over incomming messages

Used $SO(2)$ equivariance to make $SO(3)$ equivariant network!

- Simpler to construct
- More available non-linearities

Planar case: trivial automorphism



- Symmetry group $SE(2)$
  - Features are representations of $SO(2)$
- Edges have no automorphism constraints
- Message passing with unconstrained

$$f_r : \mathbb{R}^d \to \mathbb{R}^{d'}$$

  - ReLU non-linearities
- Also usable for Gauge Equivariant Mesh CNN

Conclusion

- Generalize groups to groupoids
- Generalize equivariance to natural transformations
- Message passing on graphs
  - Weight sharing if edges have isomorphic neighbourhoods
  - Local symmetries give constraints
  - More expressive
- Pointcloud equivariance via $SO(2)$ symmetry
  - Easier to implement
  - Better non-linearities
- Planar equivariance via unconstrained network
  - ReLU non-linearities
- Many more groups, groupoids, categories and representations to explore!
- Open question: link local – global naturality more formally

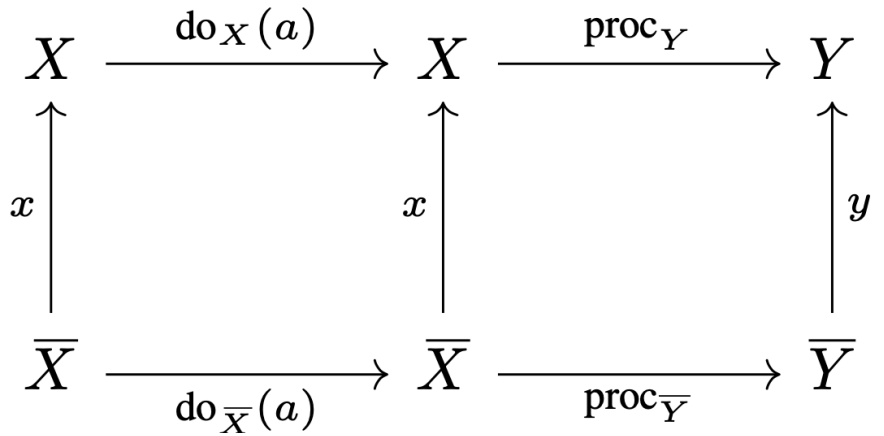
Categories & Causality

Postscriptum

Grounded Causal Models

Cohen 2022: Towards a Grounded Theory of Causation for Embodied AI

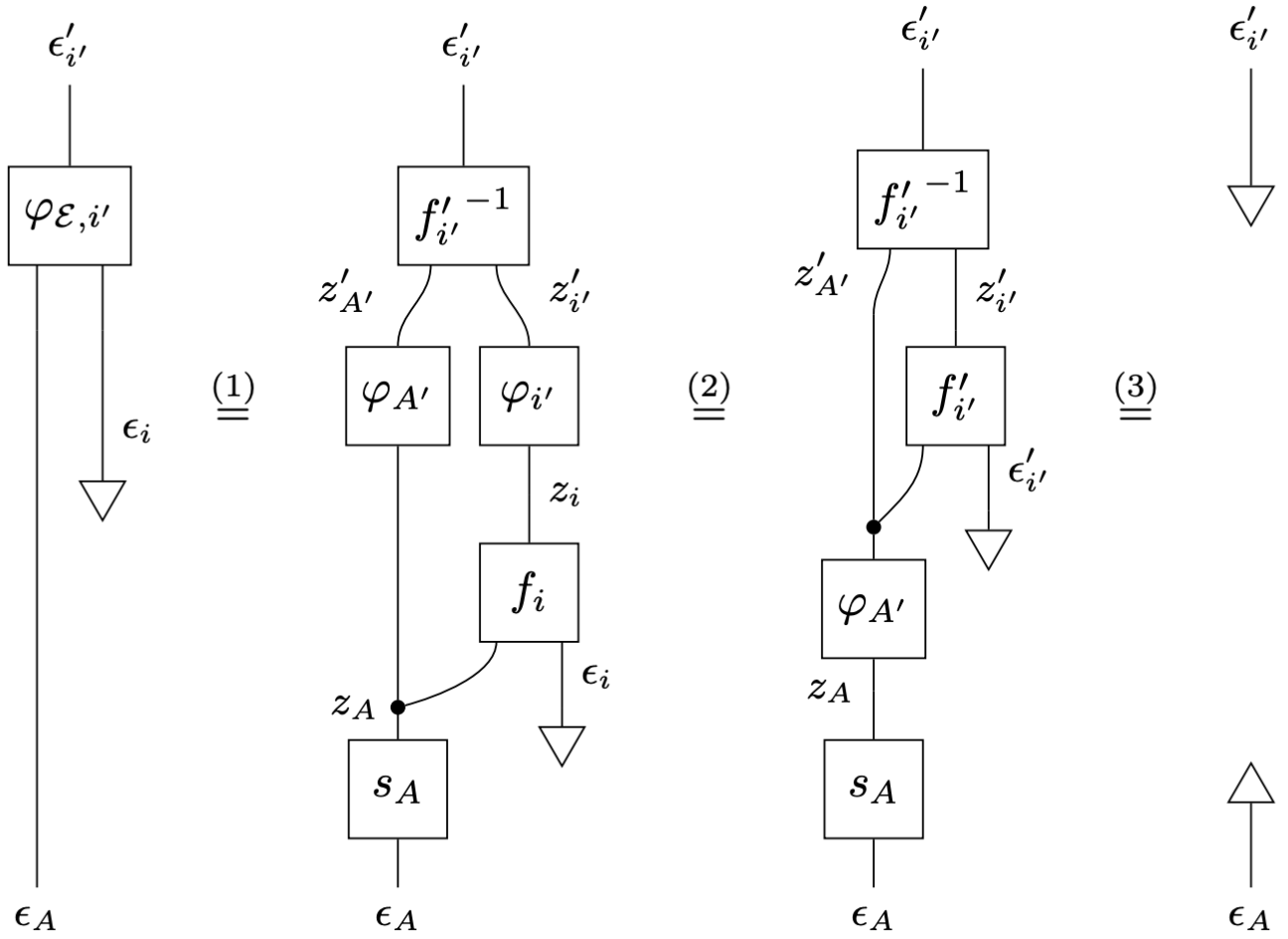
- Set of actions / interventions A
- Dynamics $\mathrm{do}(a)\colon X \to X$
- Effects $\mathrm{proc}\colon X \to Y$



A model is *grounded* if there is a natural transformation like this

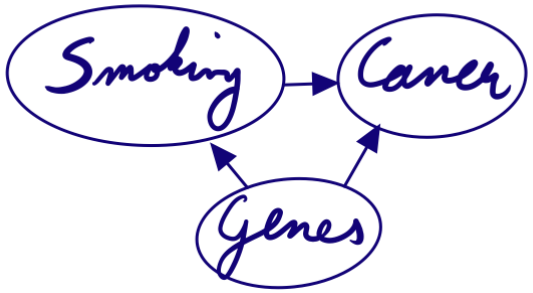
String diagrammatic proofs

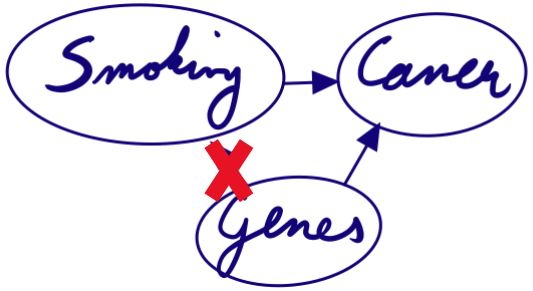
Brehmer, De Haan, Lippe, Cohen: Weakly supervised causal representation learning (2022)

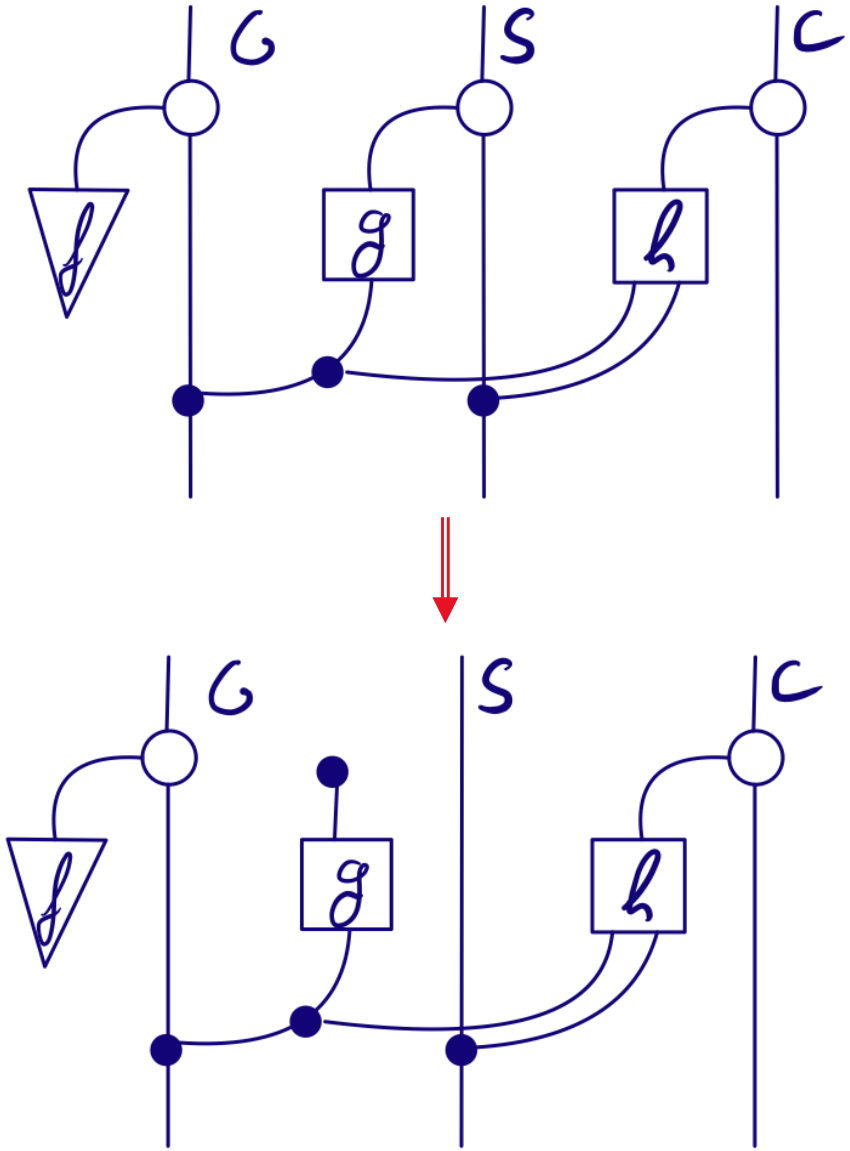


Causal Model – interventions as 2-cell

Bhat, Van Belle, De Haan, Lopez, Román (in progress)







Further reading

- Books:
  - Fong & Spivak, *Seven Sketches in Compositionality: an invitation to applied category theory*
  - Perrone, *Introduction to category theory*
  - Riehl, *Category Theory in Context*
  - Lawvere & Schanuel, *Conceptual Mathematics: a first introduction to categories*
- Papers:
  - De Haan, Cohen, Welling, Natural Graph Networks
  - Dudzik & Velickovic, Graph Neural Networks are Dynamic Programmers
  - Gavranovic, https://github.com/bgavran/Category_Theory_Machine_Learning

Thank you

Snapdragon

Follow us on:

For more information, visit us at:

snapdragon.com & snapdragoninsiders.com