Karlsruhe Institute of Technology

KCDS Summer School 2026

2. Building graph neural networks from first principles*

Jun.-Prof. Dr. Jan Stühmer

*Based on the slides by Petar Veličković, AMMI 2022

Karlsruhe Institute of Technology

Based on the slides by Petar Veličković, AMMI 2022
Available at geometricdeeplearning.com

$\mathfrak{G}$-equivariant map

- *Definition* $\mathfrak{G}$-*equivariant* map: if $\mathfrak{G}$ is acting on both $\mathcal{X}$ and $\mathcal{Y}$, a mapping $B : \mathcal{X} \to \mathcal{Y}$ is $\mathfrak{G}$-*equivariant* if for all $\mathfrak{g} \in \mathfrak{G},\ x \in \mathcal{X}$,

$$\mathrm{B}(\mathfrak{g}.x) = \mathfrak{g}.B(x)$$

- We will see many examples throughout the course (convolutions, graph diffusions, etc.)

$\mathfrak{G}$-invariant map

- *Definition* $\mathfrak{G}$-*invariant* map: if $\mathfrak{G}$ is acting on both $\mathcal{X}$ and $\mathcal{Y}$, a mapping $B: \mathcal{X} \to \mathcal{Y}$ is $\mathfrak{G}$-*invariant* if for all $\mathfrak{g} \in \mathfrak{G},\ x \in \mathcal{X}$,

$$\mathrm{A}(\mathfrak{g}.x) = A(x)$$

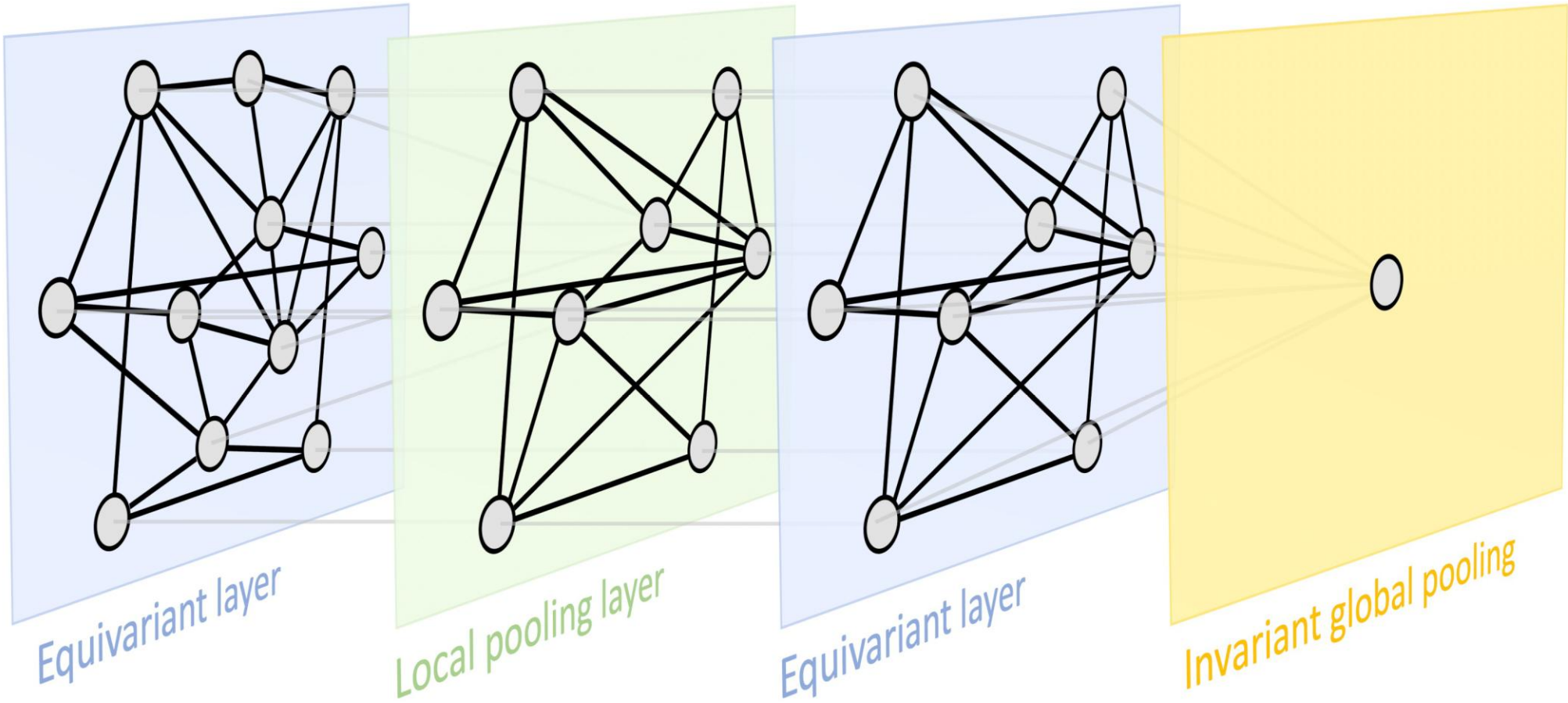
- Usually, a $\mathfrak{G}$-*invariant* map does not preserve as much information as a $\mathfrak{G}$-*equivariant* map

The Geometric Deep Learning Blueprint

- We follow a constructive approach to build rich invariant networks with multiscale structure, with the building blocks:
  - *Linear $\mathfrak{G}$-equivariant layer:* $B : \mathcal{X}(\Omega, \mathcal{C}) \to \mathcal{X}(\Omega', \mathcal{C}')$, satisfying $\mathrm{B}(\mathfrak{g}.x) = \mathfrak{g}.B(x)$ for all $\mathfrak{g} \in \mathfrak{G}$ and $\mathrm{x} \in \mathcal{X}(\Omega, \mathcal{C})$
  - Nonlinearity: $\sigma : \mathcal{C} \to \mathcal{C}'$ applied element-wise as $\big(\boldsymbol{\sigma}(x)\big)(u) = \sigma\big(x(u)\big)$
  - Local pooling (coarsening): $P : \mathcal{X}(\Omega, \mathcal{C}) \to \mathcal{X}(\Omega', \mathcal{C})$, such that $\Omega' \subseteq \Omega$
  - *$\mathfrak{G}$-invariant layer (global pooling):* $A : \mathcal{X}(\Omega, \mathcal{C}) \to \mathcal{Y}$, satisfying $A(\mathfrak{g}.x) = A(x)$ for all $\mathfrak{g} \in \mathfrak{G}$ and $\mathrm{x} \in \mathcal{X}(\Omega, \mathcal{C})$

- These blocks can be defined under very mild conditions on the geometric domain Ω

The Geometric Deep Learning Blueprint



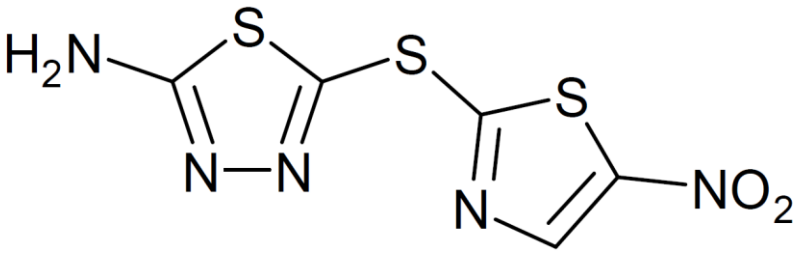
Recap: Popular architectures as instances of GDL blueprint

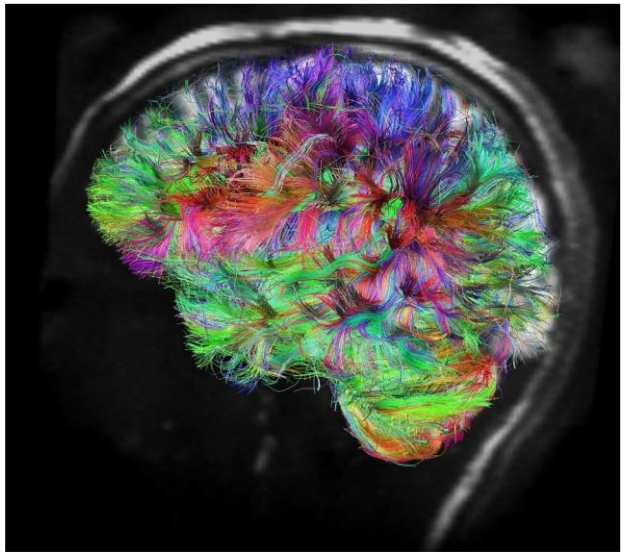
Architecture	Domain	Symmetry Group $\mathfrak{G}$
CNN	Grid	Translation
Spherical CNN	Sphere / $\mathrm{SO}(3)$	Rotation $\mathrm{SO}(3)$
Intrinsic / Mesh CNN	Manifold	Isometry $\mathrm{Iso}(\Omega)$ / Gauge symmetry $\mathrm{SO}(2)$
GNN	Graph	Permutation Σ_n
Deep Sets	Set	Permutation Σ_n
Transformer	Complete Graph	Permutation Σ_n
LSTM	1D Grid	Time warping

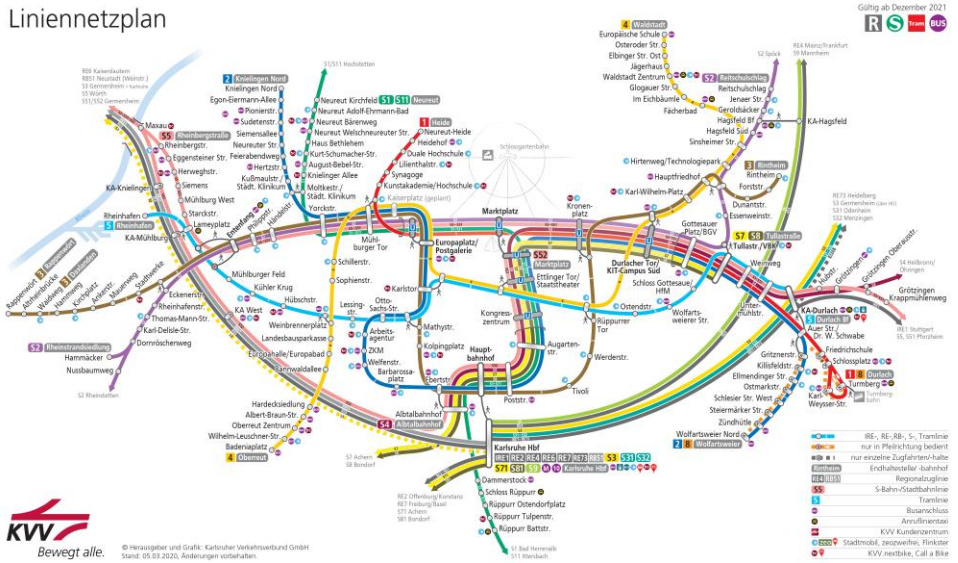
Recap: Popular architectures as instances of GDL blueprint

Architecture	Domain	Symmetry Group $\mathfrak{G}$
CNN	Grid	Translation
Spherical CNN	Sphere / $\mathrm{SO}(3)$	Rotation $\mathrm{SO}(3)$
Intrinsic / Mesh CNN	Manifold	Isometry $\mathrm{Iso}(\Omega)$ / Gauge symmetry $\mathrm{SO}(2)$
GNN	Graph	Permutation Σ_n
Deep Sets	Set	Permutation Σ_n
Transformer	Complete Graph	Permutation Σ_n
LSTM	1D Grid	Time warping

Graphs are everywhere!







Graph neural networks are everywhere!

BBC Sign in News Sport Reel Worklife Travel Future

NEWS

Home Video World UK Business Tech Science Stories Entertainment & Arts

BBC WORKLIFE Our new guide for getting ahead

Scientists discover powerful antibiotic using AI

21 February 2020 Share

VB The Machine GamesBeat Jobs Special Issue Become a Member | S

The Machine
Making sense of AI

DeepMind claims its AI improved Google Maps travel time estimates by up to 50%

Kyle Wiggers @Kyle_L_Wiggers September 3, 2020 7:00 AM

GOOGLE TECH ARTIFICIAL INTELLIGENCE

Google is using AI to design its next generation of AI chips more quickly than humans can

Designs that take humans months can be matched or beaten by AI in six hours

By James Vincent | Jun 10, 2021, 9:13am EDT

Food Discovery with Uber Eats: Using Graph Learning to Power Recommendations

Ankit Jain, Isaac Liu, Ankur Sarda, and Piero Molino December 4, 2019

Graphs as general model of data

In many ways, graphs are the main modality of data we receive from nature.

“The image of the world around us, which we carry in our head, is just a *model*. Nobody in his head imagines all the world, government or country. He has only selected *concepts*, and *relationships* between them, and uses those to represent the real system.”

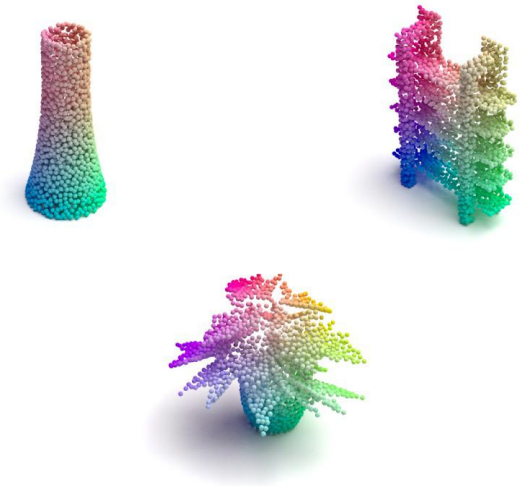
Jay Wright Forrester

1971

Where to begin?

We will first look at *graphs without connections* (**sets**)

- Much simpler to analyze
- Most conclusions will naturally *carry over* to graphs
- Still *very relevant!* (point clouds from cameras and LiDAR)



Du, Wang, Cremers 2020

Learning on sets: Setup

For now, assume our graph **has no edges** (i.e. $\Omega = \mathcal{V}$, the set of nodes)

Let $\mathbf{x}_i \in \mathbb{R}^k$ be the features of node i. (i.e. our feature space is $\mathcal{C} = \mathbb{R}^k$).

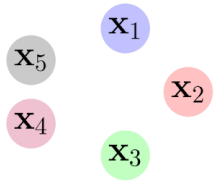
We can stack these features into a *node feature matrix* of shape $|\mathcal{V}| \times k$:

$$\mathbf{X} = (\mathbf{x_1}, \dots, \mathbf{x_n})^T$$

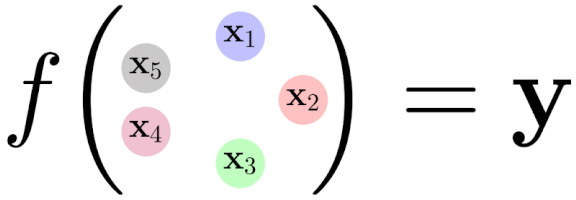
That is, the i -th row of $\mathbf{X}$ corresponds to $\mathbf{x_i}$

Note that, by doing so, we have specified a **node ordering**!
We would like the result of any neural networks to not depend on this.

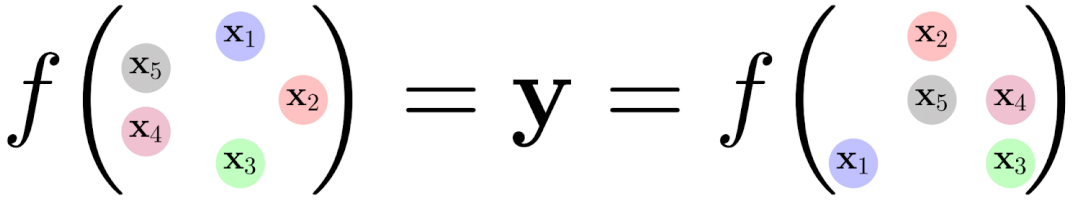
What do we want?



What do we want?



What do we want?



Symmetry group $\mathfrak{G}$: n-element Permutation group Σ_n
Group element $\mathfrak{g} \in \mathfrak{G}$: Permutations

Permutations and Permutation Matrices

It will be useful to think about operators that **change** the node order

Such operations are known as *permutations*

e.g. a permutation (2, 4, 1, 3) maps $\mathbf{x_1} \leftarrow \mathbf{x_2}, \mathbf{x_2} \leftarrow \mathbf{x_4}, \mathbf{x_3} \leftarrow \mathbf{x_1}, \mathbf{x_4} \leftarrow \mathbf{x_3}$

Permutations and Permutation Matrices

Within linear algebra, each permutation defines a $|\mathcal{V}| \times |\mathcal{V}|$ **matrix**

- Such matrices are called permutation matrices (group representation $\rho(\mathfrak{g})$)
- They have exactly one 1 in every row and column, zeros elsewhere
- Their effect when left-multiplied is to permute rows of $\mathbf{X}$, like so:

$$\mathbf{P}_{(2,4,1,3)}\mathbf{X} = \begin{bmatrix} \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} \\ \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} \end{bmatrix} \begin{bmatrix} - \mathbf{X}_1 - \\ - \mathbf{X}_2 - \\ - \mathbf{X}_3 - \\ - \mathbf{X}_4 - \end{bmatrix} = \begin{bmatrix} - \mathbf{X}_2 - \\ - \mathbf{X}_4 - \\ - \mathbf{X}_1 - \\ - \mathbf{X}_3 - \end{bmatrix}$$

Permutation Invariance

- Want: Functions $f(\mathbf{X})$ over sets that will not depend on the order
- Equivalently: applying a permutation matrix should not modify the result!
- We arrive at a very useful notion of permutation invariance:
  $f(\mathbf{X})$ is *permutation invariant* if, for *all* permutation matrices $\mathbf{P}$:

$$f(\mathbf{PX}) = f(\mathbf{X})$$

Permutation Invariance

- Want: Functions $f(\mathbf{X})$ over sets that will not depend on the order
- Equivalently: applying a permutation matrix should not modify the result!
- We arrive at a very useful notion of permutation invariance:
  $f(\mathbf{X})$ is *permutation invariant* if, for *all* permutation matrices $\mathbf{P}$:

$$f(\mathbf{PX}) = f(\mathbf{X})$$

- Compare with GDL blueprint:
  $\mathfrak{G}$-invariant layer (global pooling) $A: \mathcal{X}(\Omega, \mathcal{C}) \to \mathcal{Y}$,
  satisfying $A(\mathfrak{g}.x) = A(x)$ for all $\mathfrak{g} \in \mathfrak{G}$ and $x \in \mathcal{X}(\Omega, \mathcal{C})$.

Deep Sets

- A very generic model is the Deep Sets model [Zaheer et al., Neurips '18]:

$$f(\mathbf{X}) = \rho\left(\sum_{i \in \mathcal{V}} \psi(\mathbf{x}_i)\right)$$

  where ρ and ψ are universal function approximators, e.g. MLPs.

- A commutative aggregation, e.g. **sum** is *critical!*
  (other choices possible, e.g. **max**, **min**, **median**, or **avg**)

Deep Sets

- A very generic model is the Deep Sets model [Zaheer et al., Neurips '18]:

$$f(\mathbf{X}) = \rho\left(\bigoplus_{i \in \mathcal{V}} \psi(\mathbf{x}_i)\right)$$

  where ρ and ψ are universal function approximators, e.g. MLPs.

- A commutative aggregation, e.g. **sum** is *critical!*
  (other choices possible, e.g. **max**, **min**, **median**, or **avg**)
- We will use $\bigoplus$ to denote *any* permutation invariant operator.

Permutation equivariance

- Permutation invariant models are good for set-level outputs
- What if we would like answers at the **node** level?
- We want to still be able to **identify** node outputs, which a permutation-invariant aggregator would destroy!
- We may instead seek functions that don't **change** the node order
  i.e. if we permute nodes, it doesn't matter if we do it **before** or **after**!
- Accordingly, we say that $\mathbf{F}: \mathcal{X} \to \mathcal{Y}$, is *permutation equivariant* if, for all permutation matrices $\mathbf{P}$ (and corresponding matrices $\widetilde{\mathbf{P}}$ that act on $\mathcal{Y}$):

$$\mathbf{F}(\mathbf{PX}) = \widetilde{\mathbf{P}}\mathbf{F}(\mathbf{X})$$

Permutation equivariance

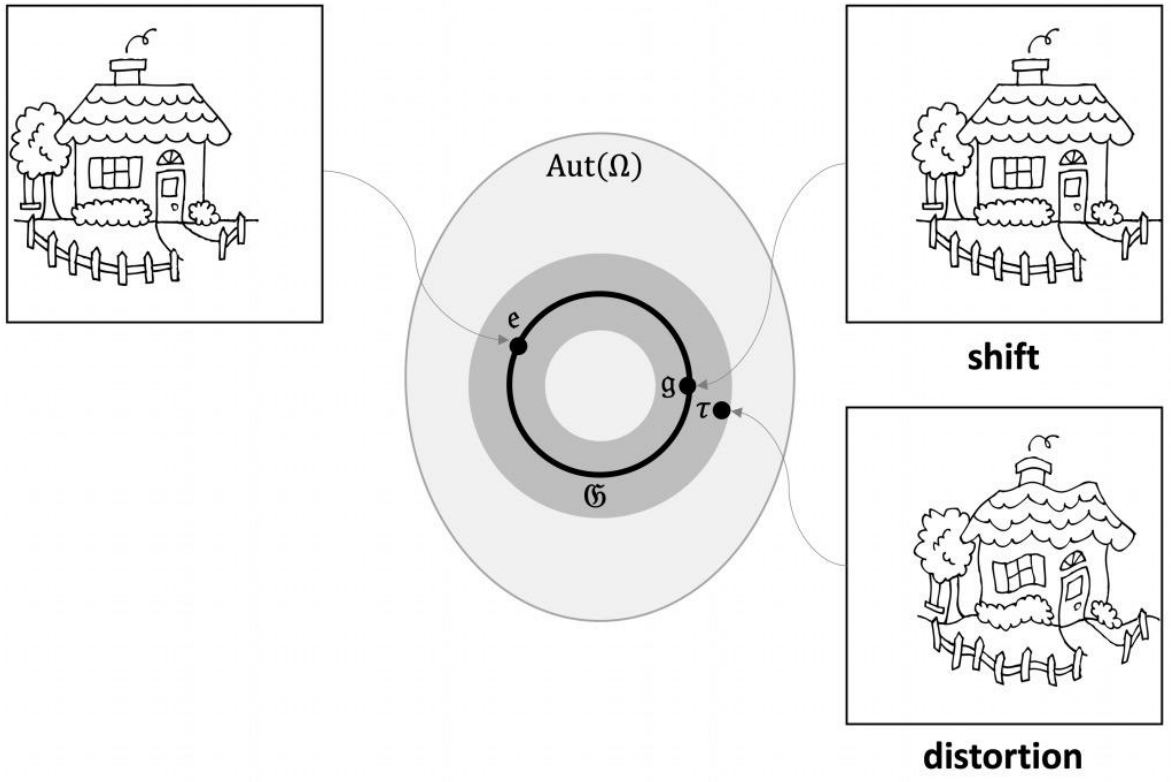
- Accordingly, we say that $\mathbf{F}: \mathcal{X} \to \mathcal{Y}$ is *permutation equivariant* if, for all permutation matrices $\mathbf{P}$ (and corresponding matrices $\widetilde{\mathbf{P}}$ that act on $\mathcal{Y}$):

$$\mathbf{F}(\mathbf{PX}) = \widetilde{\mathbf{P}}\mathbf{F}(\mathbf{X})$$

- Compare with GDL blueprint:
  Linear $\mathfrak{G}$-equivariant layer $B: \mathcal{X}(\Omega, \mathcal{C}) \to \mathcal{X}(\Omega', \mathcal{C}')$,
  satisfying $B(\mathfrak{g}.x) = \mathfrak{g}.B(x)$ for all $\mathfrak{g} \in \mathfrak{G}$ and $x \in \mathcal{X}(\Omega, \mathcal{C})$.

Important Constraint: Locality

- We want a signal to be **stable** under slight *deformations* of the domain

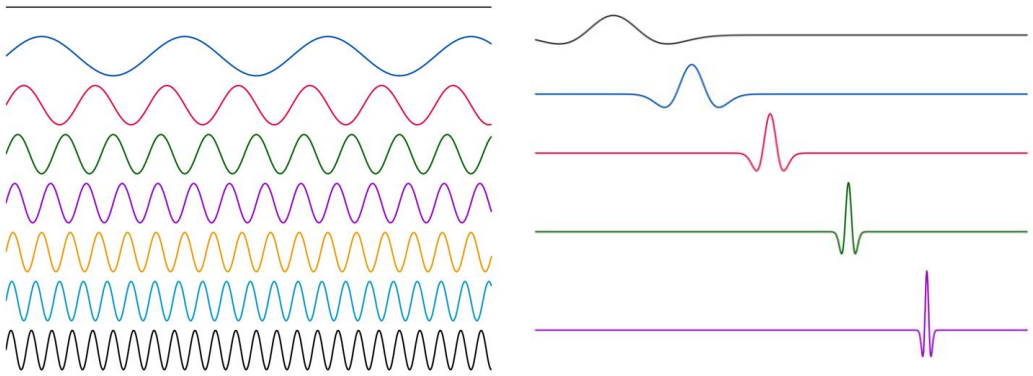


[Bronstein, Bruna, Cohen, Veličković, 2021]

Important Constraint: Locality

- Want signal to be **stable** under slight *deformations* of the domain

- It is highly beneficial to compose local operations to model larger-scale ones, as local ops won't globally propagate errors
  e.g. CNNs with 3 x 3 kernels, but very deep
- Accordingly, we would like to support locality in our layers!
- cf. Fourier Transform vs. Wavelets



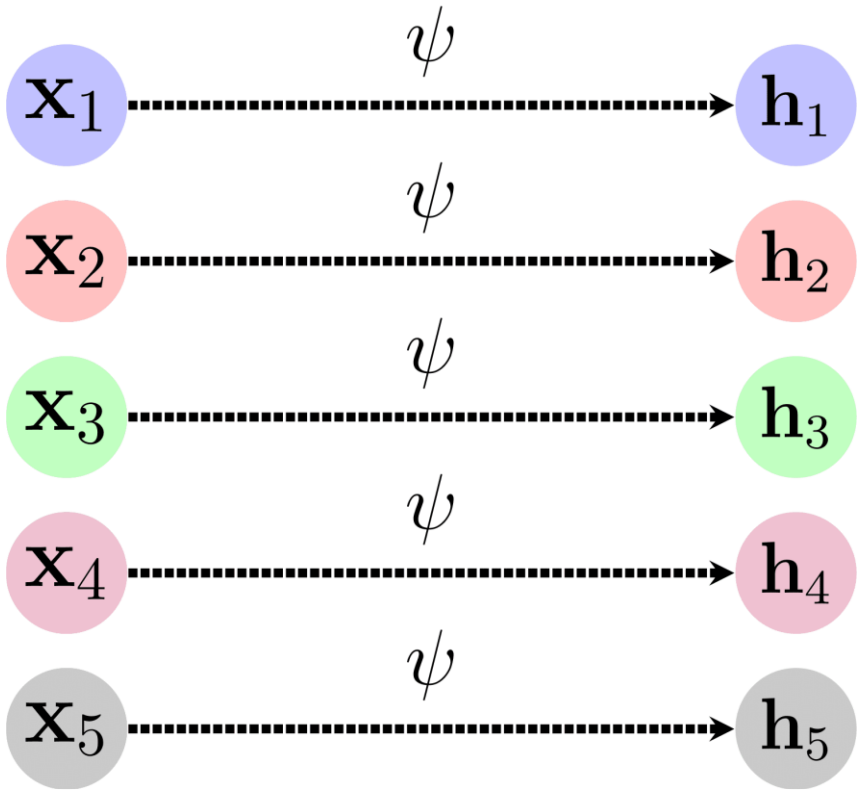
What does this mean for sets?

General blueprint for learning on sets

- One way to enforce locality in equivariant set functions is through a shared function ψ applied to every node in isolation:

$$\mathbf{h}_i = \psi(\mathbf{x}_i)$$

- Stacking $\mathbf{h}_i$ into a matrix yields $\mathbf{H} = \mathbf{F}(\mathbf{X})$



General blueprint for learning on sets

- One way to enforce locality in equivariant set functions is through a shared function ψ applied to every node in isolation:

$$\mathbf{h}_i = \psi(\mathbf{x}_i)$$

- Deep Sets as a model that follows the geometric deep learning blueprint: invariance by aggregating equivariant functions

$$f(\mathbf{X}) = \rho\left(\bigoplus_{i \in \mathcal{V}} \psi(\mathbf{x}_i)\right)$$

Is this really as far as we can get?

- What we showed so far is:
  - Useful functions on sets are permutation equivariant / invariant
  - Deep Sets offer one way of achieving permutation invariance
  - Is this necessarily the only way?

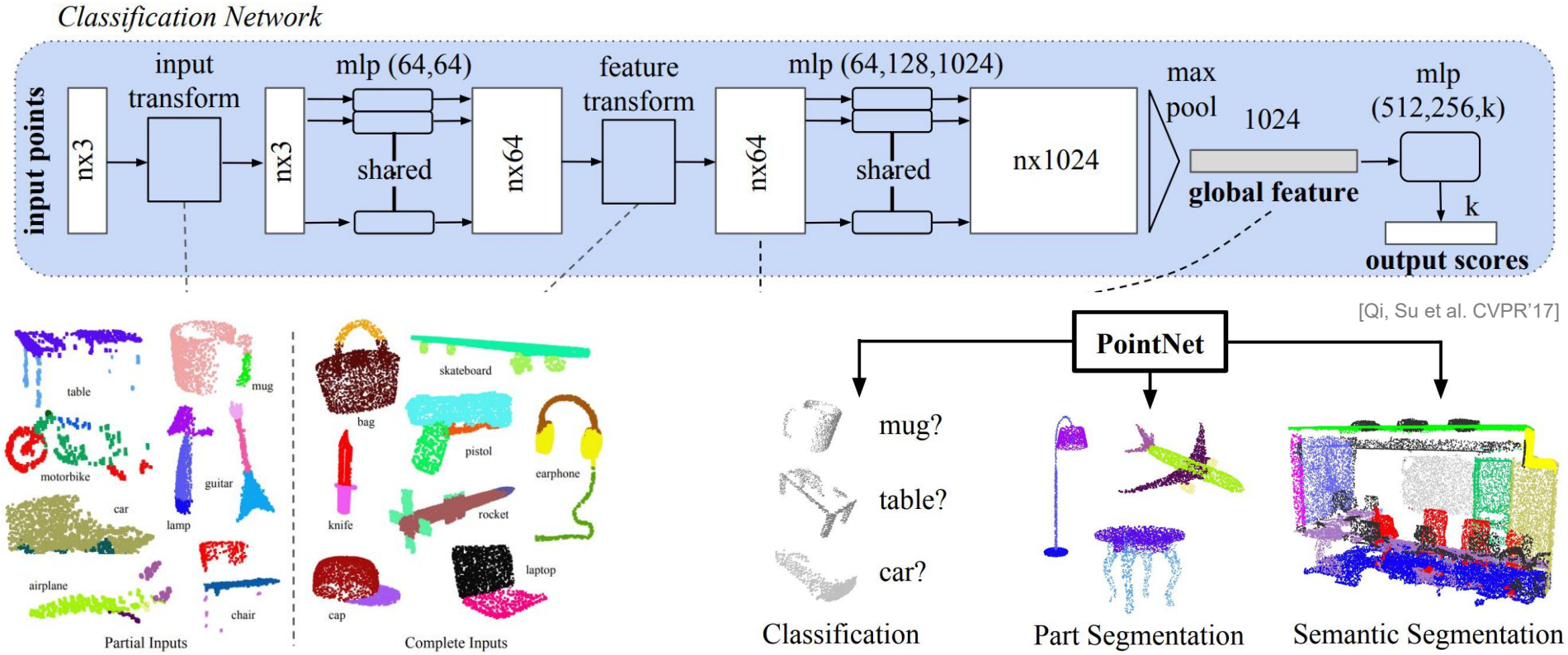
- For many classes of input sets, the answer is a decisive yes!

- That is, no matter what other permutation invariant model you propose, it will necessarily be expressible as a Deep Sets

- We do not have the time to cover the relevant proofs; see:
  - Zaheer et al. (NeurIPS'17)
  - Wagstaff, Fuchs, Engelcke et al. (ICML'19)

Applications of Deep Sets: PointNet

- The PointNet model of [Qi, Su *et al.* CVPR'17] empirically demonstrates that Deep Sets-style ideas work



Applications of Deep Sets: PointNet

- The PointNet model of [Qi, Su *et al.* CVPR'17] empirically demonstrates that Deep Sets-style ideas work
- It applies point-level MLPs and (max)-pooling, just like Deep Sets
- Uses a very similar proof strategy to show their network is universal
- PointNets also feature a coordinate transformation which is mindful of the points' geometry in 3D space (so the set is not "purely unordered")
- We won't study such transformations more deeply at this time
  Keep it in mind – it will be useful for geometry-aware GNNs

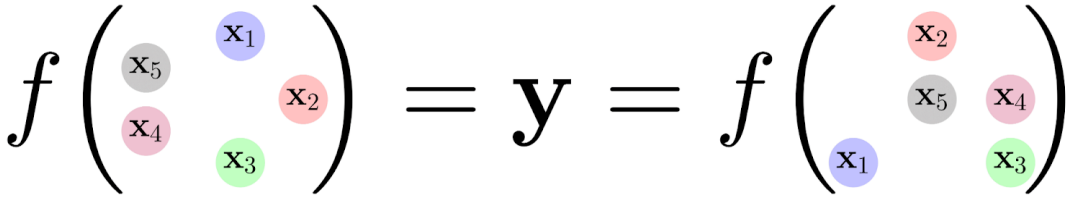
Learning on Graphs

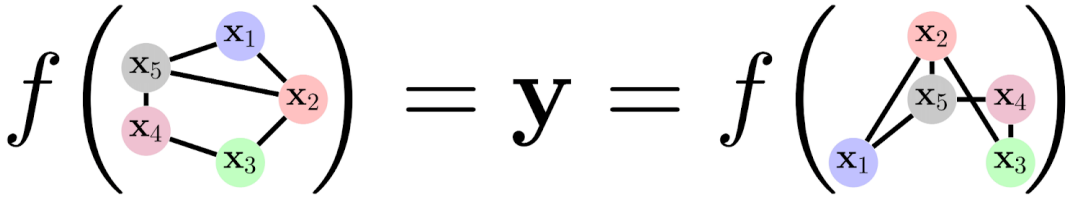
- Now we can augment the set of nodes with edges between them
  That is, we consider graphs $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$
- We can represent these edges in an *adjacency matrix* $\mathbf{A}$, such that:

$$a_{ij} = \begin{cases} 1, & (i,j) \in \mathcal{E} \\ 0, & (i,j) \notin \mathcal{E} \end{cases}$$

- Note that the edges are now *part of the domain!*
- Further additions, e.g. edge features, are possible but **ignored** for now
- Our main desiderata *(permutation {in,equi}variance)* still hold!

What has changed?





Permutation invariance and equivariance on graphs

- Main difference: permutations now also accordingly act on the **edges**
- We need to appropriately permute both **rows** and **columns** of $\mathbf{A}$
  When applying a permutation matrix $\mathbf{P}$, this amounts to $\mathbf{PAP}^{\mathrm{T}}$
- We arrive at updated definitions of suitable functions over graphs:

Invariance: $f\left(\mathbf{PX}, \mathbf{PAP}^{\mathrm{T}}\right) = f(\mathbf{X}, \mathbf{A})$

Equivariance: $\mathbf{F}\left(\mathbf{PX}, \mathbf{PAP}^{\mathrm{T}}\right) = \widetilde{\mathbf{P}}\mathbf{F}(\mathbf{X}, \mathbf{A})$

Locality on graphs: Neighbourhoods

- On sets, we enforced *locality* by transforming every node **in isolation**
- *Graphs* give us a broader context: a node's *neighborhood*
  For a node i, its (1-hop) neighborhood, $\mathcal{N}_i$ is commonly defined as:

$$\mathcal{N}_i = \{j : (i,j) \in \mathcal{E} \ \vee (j,i) \in \mathcal{E}\}$$

- Accordingly, we can extract neighborhood features, the multiset* $\mathbf{X}_{\mathcal{N}_i}$

$$\mathbf{X}_{\mathcal{N}_i} = \left\{\!\left\{ x_j : j \in \mathcal{N}_i \right\}\!\right\}$$

  and define a *local* function, $\phi\left(\mathbf{x}_i, \mathbf{X}_{\mathcal{N}_i}\right)$, operating over them.

*We define $\mathbf{X}_{\mathcal{N}_i}$ as a multiset because feature vectors x_j are not necessarily unique.

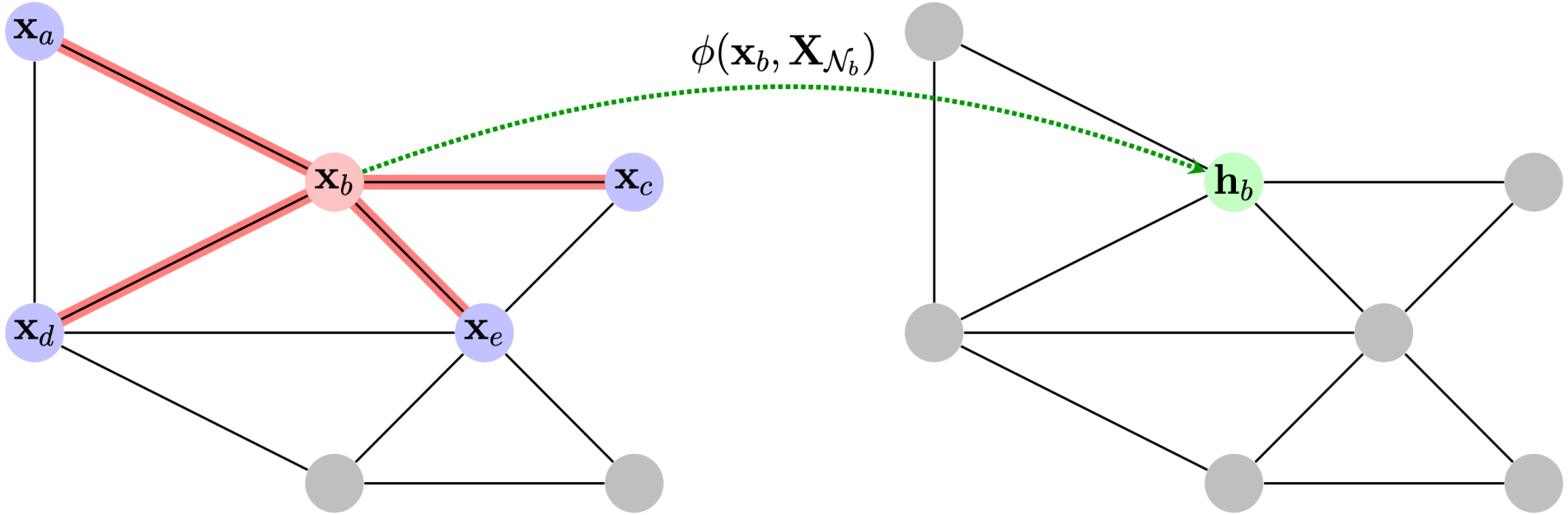
Recipe for graph neural networks

- Now we can construct permutation equivariant functions, $\mathbf{F}(\mathbf{X}, \mathbf{A})$, by appropriately applying the local function, ϕ, over all neighborhoods:

$$\mathbf{F}(\mathbf{X}, \mathbf{A}) = \begin{bmatrix} - & \phi\left(\mathbf{x}_1, \mathbf{X}_{\mathcal{N}_1}\right) & - \\ - & \phi\left(\mathbf{x}_2, \mathbf{X}_{\mathcal{N}_2}\right) & - \\ - & \phi\left(\mathbf{x}_3, \mathbf{X}_{\mathcal{N}_3}\right) & - \\ - & \phi\left(\mathbf{x}_4, \mathbf{X}_{\mathcal{N}_4}\right) & - \end{bmatrix}$$

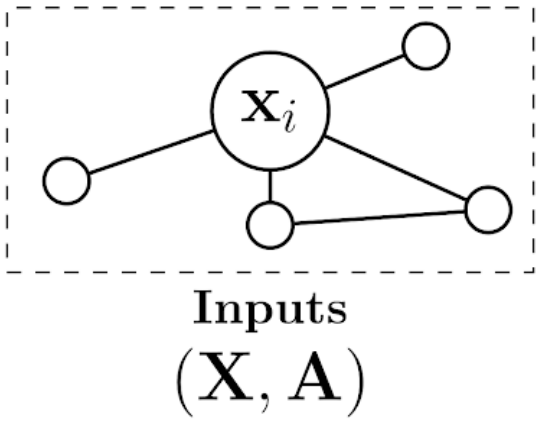
- To ensure equivariance, it is sufficient if ϕ does not depend on the **order** of the nodes in $\mathbf{X}_{\mathcal{N}_i}$ (i.e. if it is *permutation invariant*).

Recipe for graph neural networks, visualized

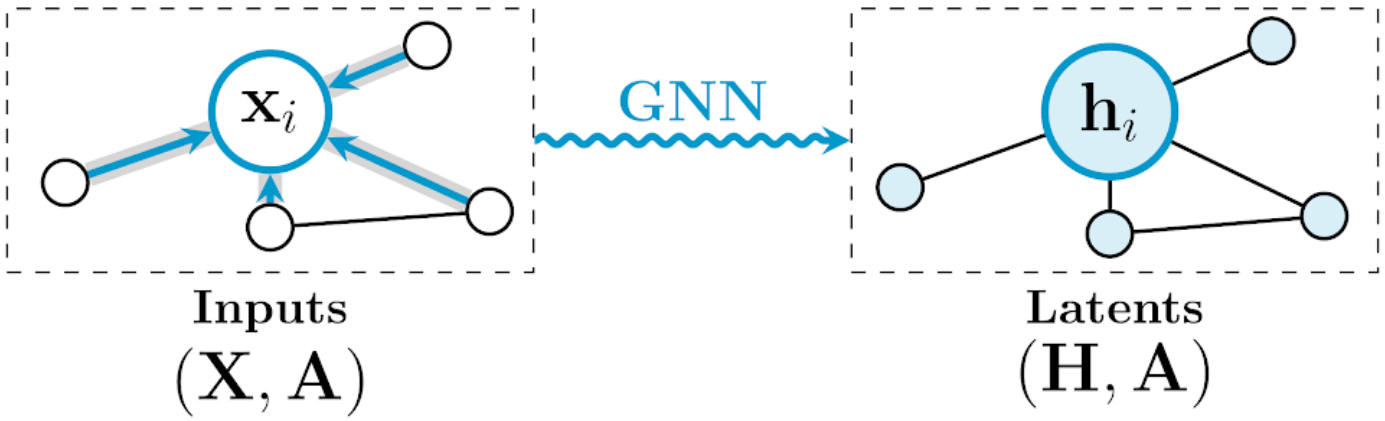


$$\mathbf{X}_{\mathcal{N}_b} = \{x_a, x_b, x_c, x_d, x_e\}$$

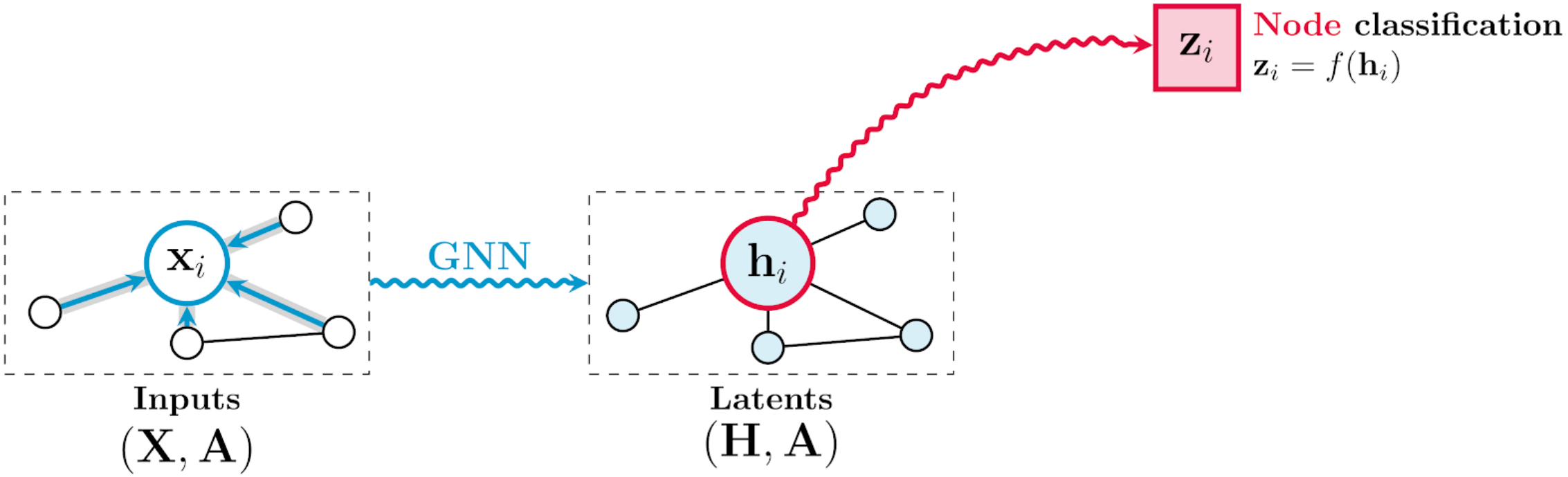
General blueprint for learning on graphs



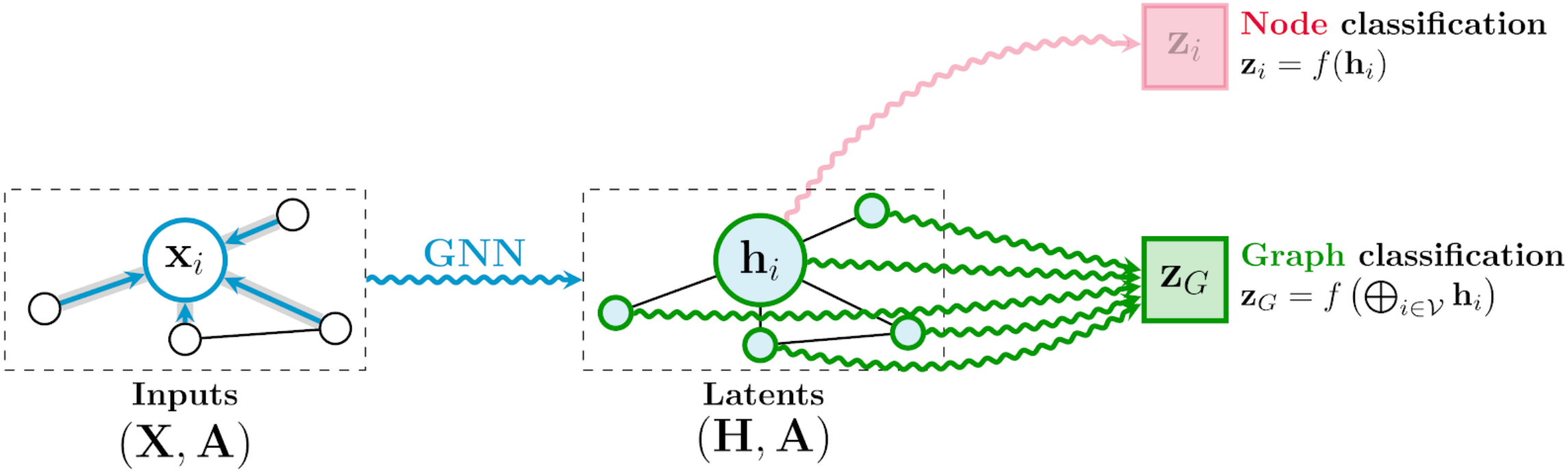
General blueprint for learning on graphs



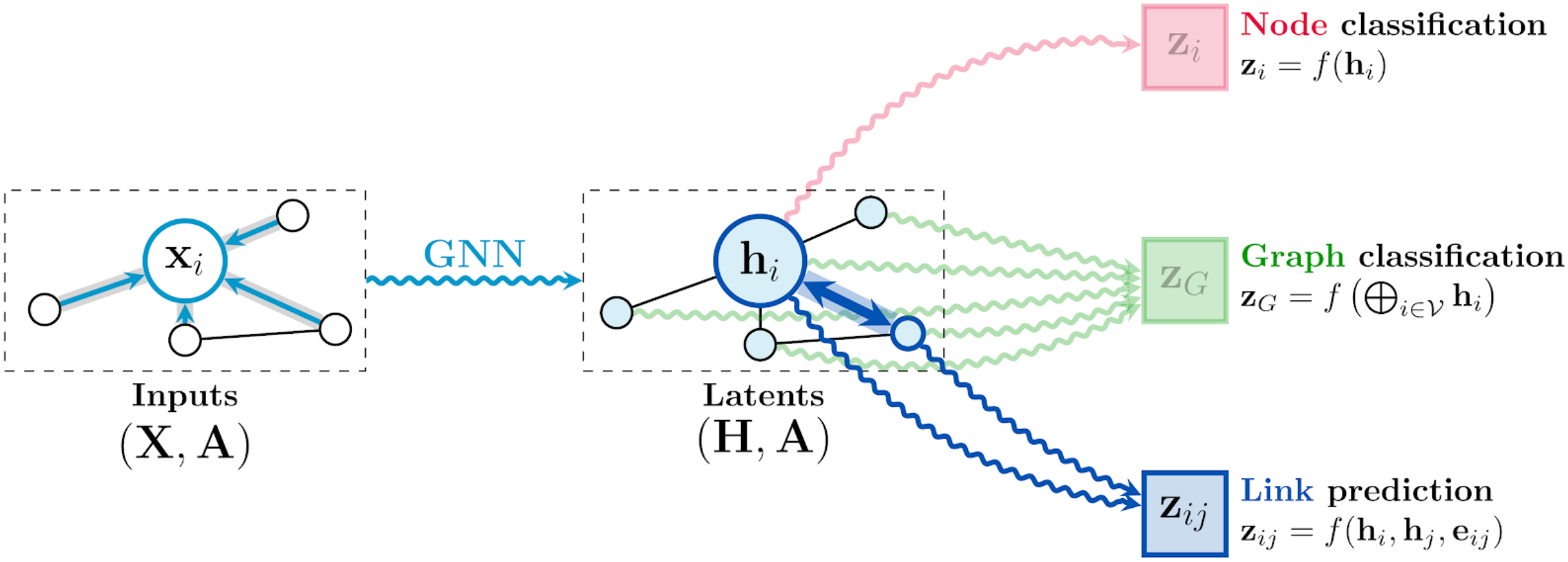
General blueprint for learning on graphs



General blueprint for learning on graphs



General blueprint for learning on graphs



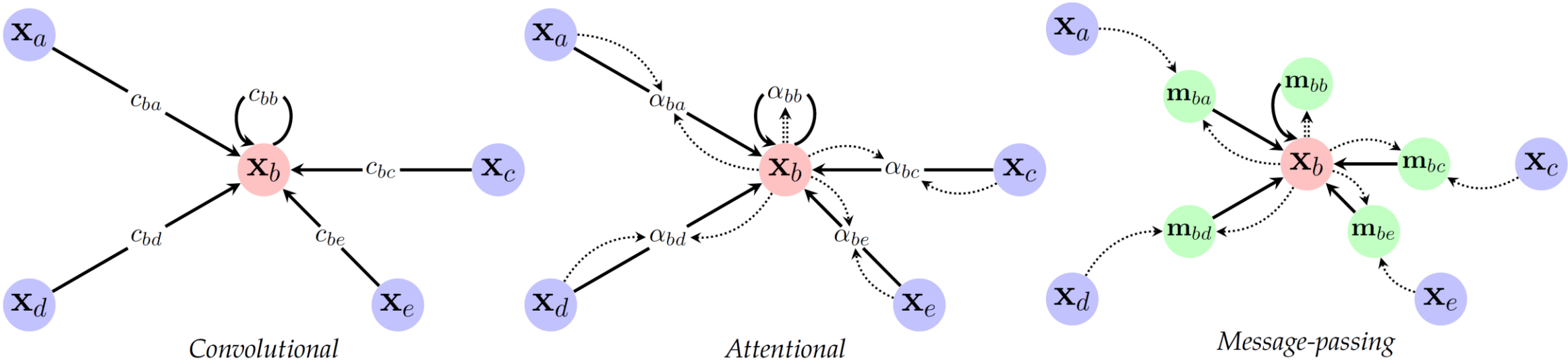
What is in a GNN layer?

- We build permutation equivariant functions $\mathbf{F}(\mathbf{X}, \mathbf{A})$ on graphs by shared application of a local permutation-invariant $\phi\left(\mathbf{x}_i, \mathbf{X}_{\mathcal{N}_i}\right)$

- Common terminology:
  - $\mathbf{F}$ is a “GNN layer”
  - ϕ is “diffusion” / “propagation” / “message passing”

- But **how** do we implement ϕ?
- **Very intense** area of research!
- Fortunately, *almost all* of them can be classified across three *“flavours”*

The three “falvours” of GNN layers



[Bronstein, Bruna, Cohen, Veličković, 2021]

$$\mathbf{h}_i = \phi\left(\mathbf{x}_i, \bigoplus_{j \in \mathcal{N}_i} c_{ij}\, \psi(\mathbf{x}_j)\right) \qquad \mathbf{h}_i = \phi\left(\mathbf{x}_i, \bigoplus_{j \in \mathcal{N}_i} a(\mathbf{x}_i, \mathbf{x}_j)\, \psi(\mathbf{x}_j)\right) \qquad \mathbf{h}_i = \phi\left(\mathbf{x}_i, \bigoplus_{j \in \mathcal{N}_i} \psi(\mathbf{x}_i, \mathbf{x}_j)\right)$$

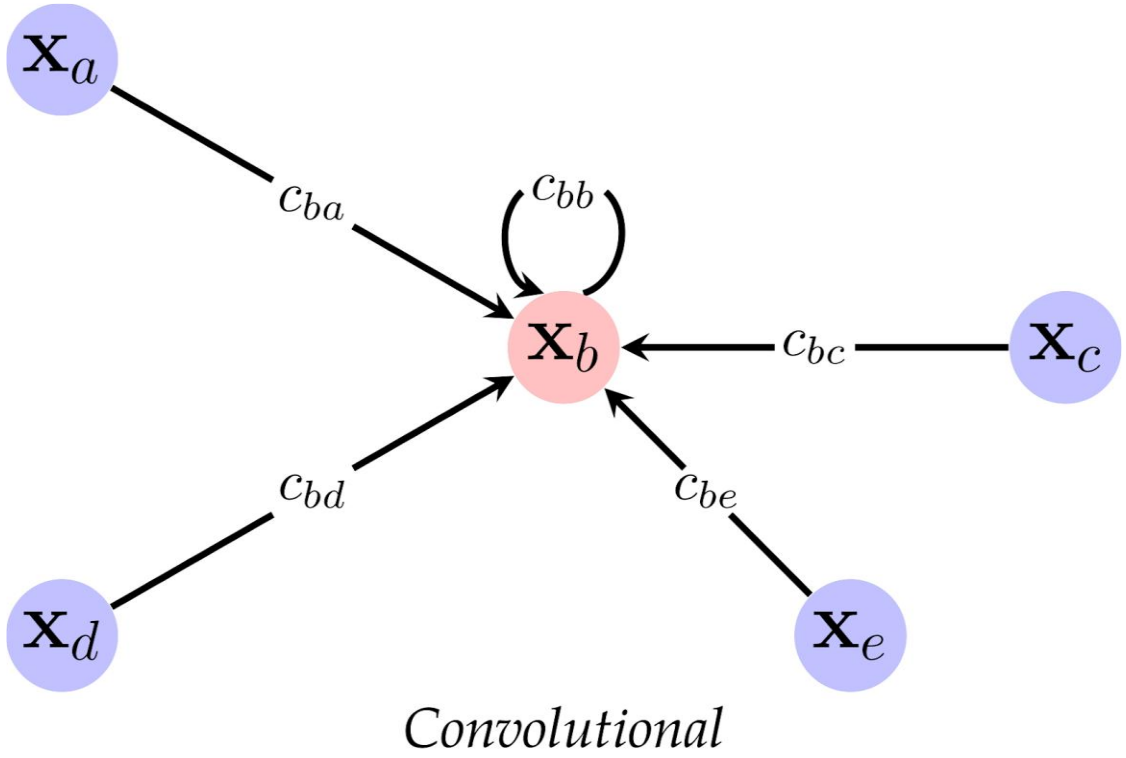
Convolutional GNN

- Features of neighbors aggregated with fixed weights, c_{ij}

$$\mathbf{h}_i = \phi\left(\mathbf{x}_i, \bigoplus_{j \in \mathcal{N}_i} c_{ij}\, \psi(\mathbf{x}_j)\right)$$

- Usually weights depend directly on $\mathbf{A}$
  - ChebyNet [Defferand *et al.*, Neurips'16]
  - GCN [Kipf & Welling, ICLR'17]
  - SGC [Wu *et al.*, ICML'19]

- Useful for **homophilous** graphs (when edges encode *label similarity*)
- Highly **scalable**



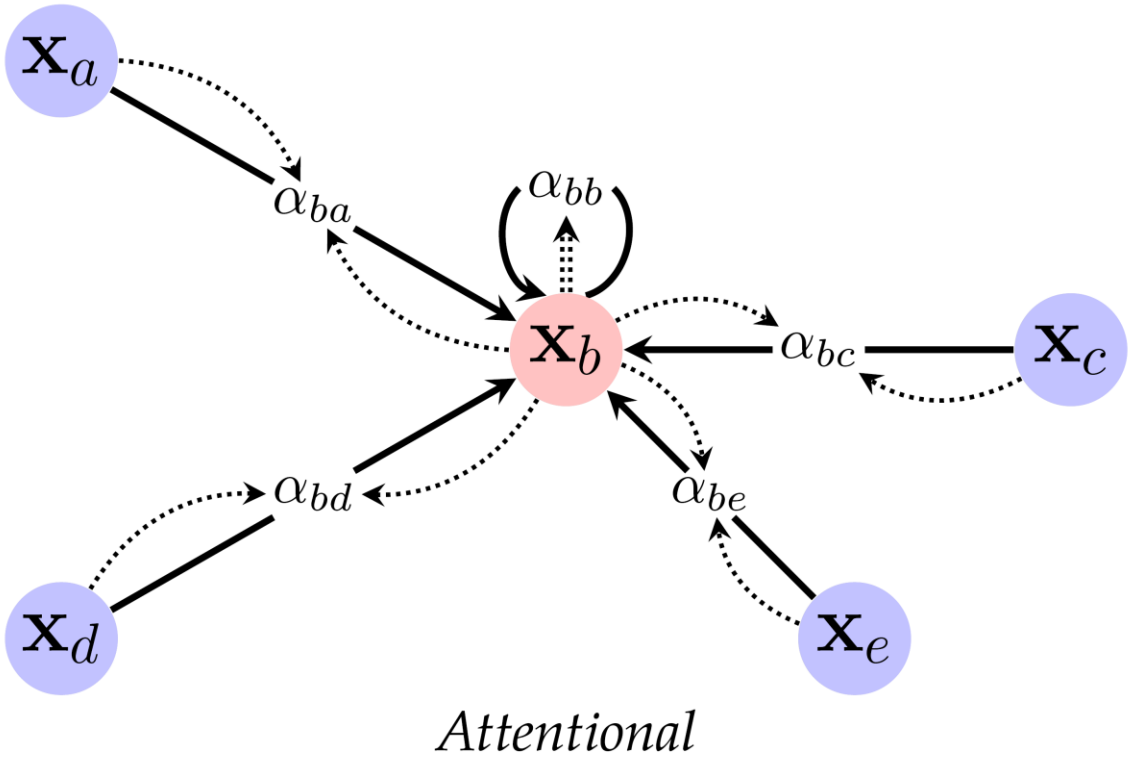
Attentional GNN

- Features of neighbors aggregated with **implicit** weights *(attention)*

$$\mathbf{h}_i = \phi\left(\mathbf{x}_i, \bigoplus_{j \in \mathcal{N}_i} a(\mathbf{x}_i, \mathbf{x}_j)\, \psi(\mathbf{x}_j)\right)$$

- Attention weights computed as $\alpha_{ij} = a(\mathbf{x}_i, \mathbf{x}_j)$
  - MoNet [Monti *et al.*, CVPR'17]
  - GAT [Veličković *et al.*, ICLR'18]
  - GATv2 [Brody et al., ICLR'22]

- Useful as "middle ground" w.r.t. capacity, scale, interpretability
- Edges need not encode homophily
- But still computing only a scalar per edge



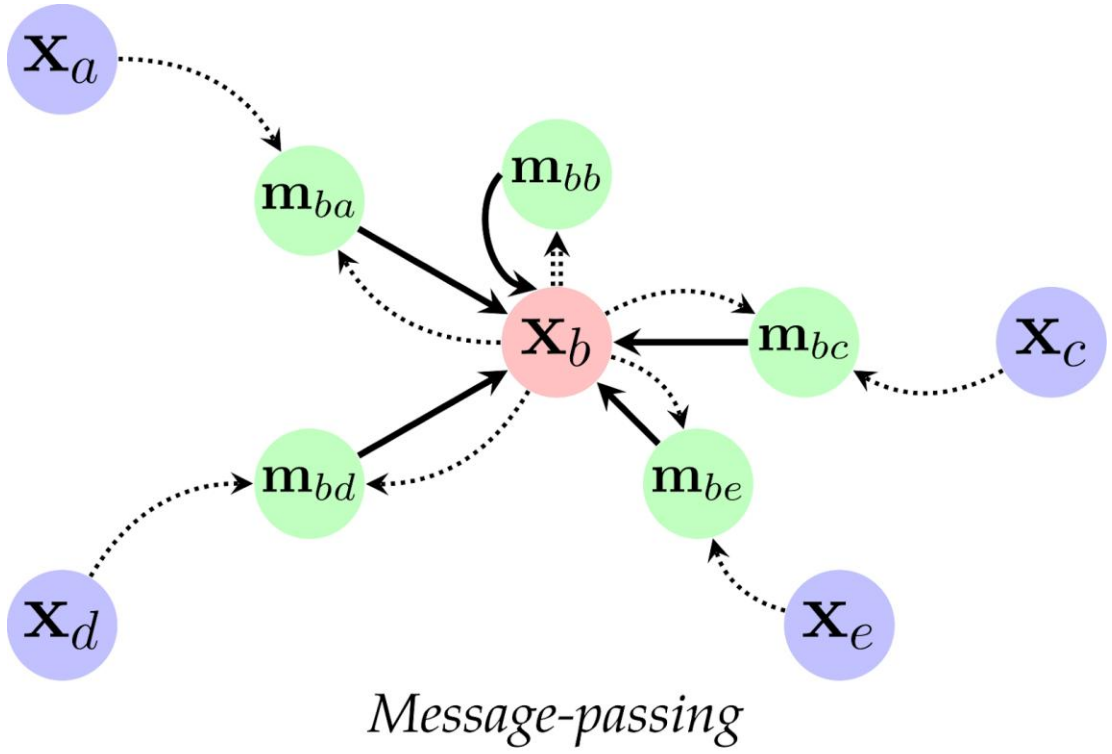
Message-passing GNN

- Compute **arbitrary vectors** *(messages)* to be sent across edges

$$\mathbf{h}_i = \phi\left(\mathbf{x}_i, \bigoplus_{j \in \mathcal{N}_i} \psi(\mathbf{x}_i, \mathbf{x}_j)\right)$$

- Messages computed as $\mathbf{m}_{ij} = \psi(\mathbf{x}_i, \mathbf{x}_j)$
  - Interaction Nets [Battaglia *et al.*, Neurips'16]
  - MPNN [Gilmer *et al.*, ICML'17]
  - GraphNets [Battaglia *et al.*, 2018]

- Most **generic** GNN layer
- May have *scalability* or *learnability* issues
- Ideal for *computational chemistry*, *reasoning* and *simulation* tasks



What have we covered?

- Instantiations of the geometric DL blueprint on **sets** and **graphs**
- Key notions of permutation *invariance* and *equivariance*
- Deep Sets as a “universal” blueprint for set neural networks
- Permutation equivariant GNN layers and their uses
- The three *”flavours”* of GNN layers:
  convolutional, attentional, message-passing

What's next?

- Demonstrate an exact **implementation** of a generic GNN: *Graph Nets*
- Connections between *GNNs*, *Deep Sets* and *Transformers*
- How **powerful** are GNNs? *Graph isomorphism testing*
- Insight into the *algorithmic foundations* of GNNs
- (In the Geometric Deep Learning lecture: spectral GNNs, geometric GNNs, and the graph Fourier transform)