PHYS 139/239: Machine Learning in Physics

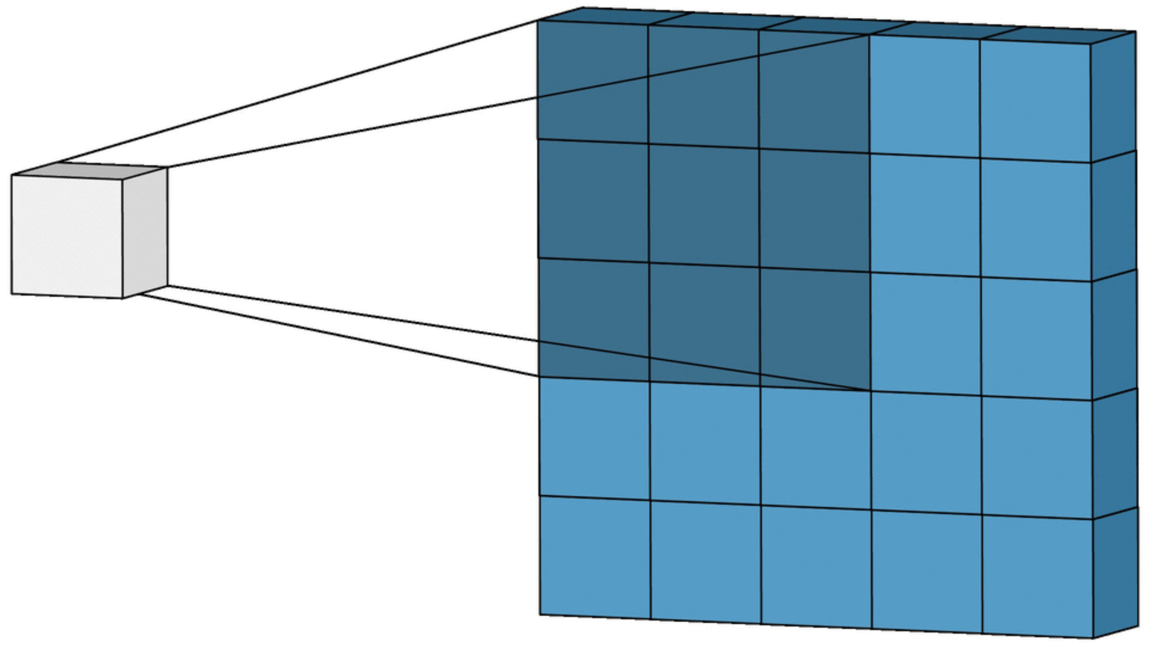
Lecture 11: Graph neural networks

Javier Duarte — February 14, 2023

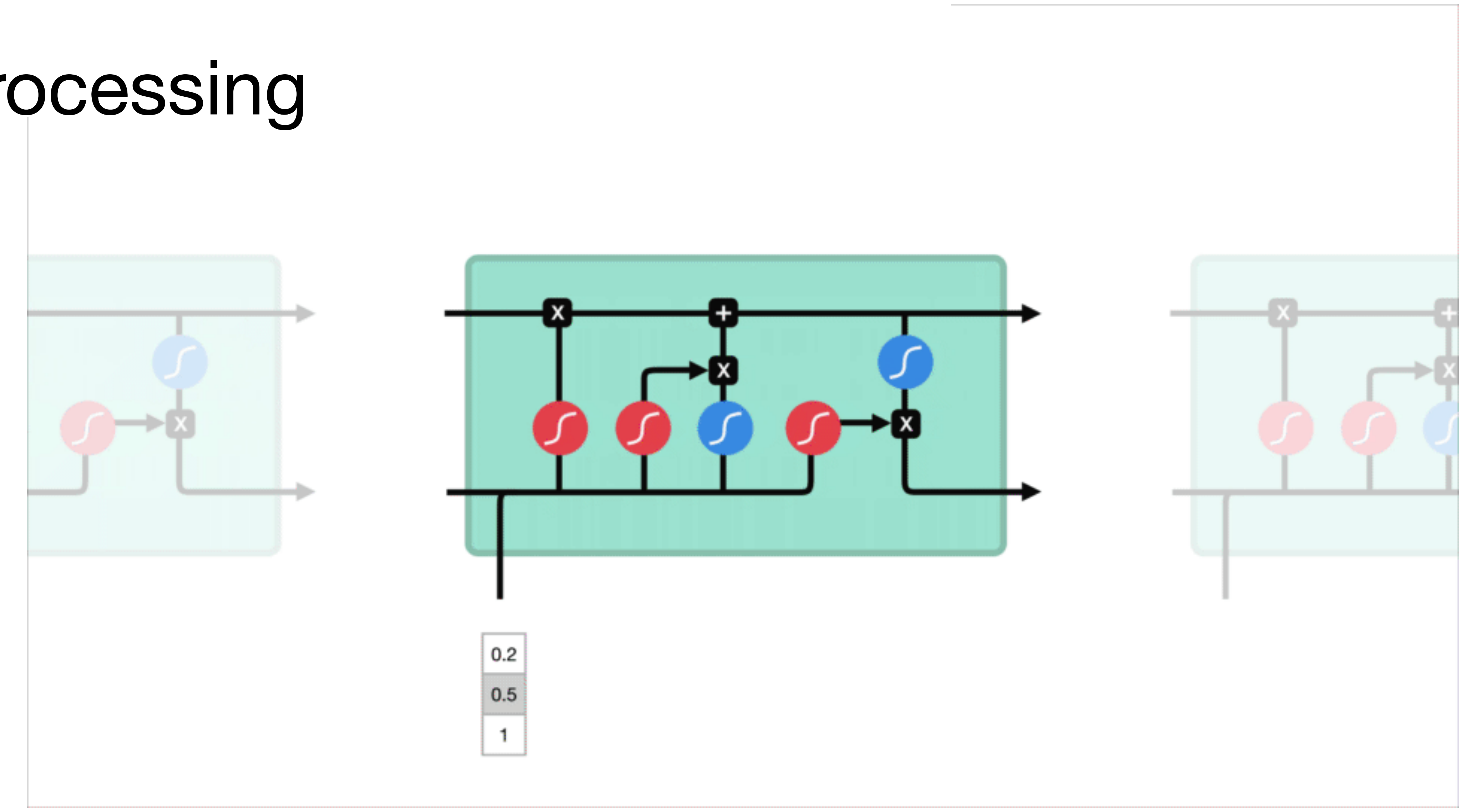
Data representations

arXiv:2007.13681
arXiv:2012.01249

- In deep learning, tailoring algorithms to the structure (and symmetries) of the data has led to groundbreaking performance
    - CNNs for images



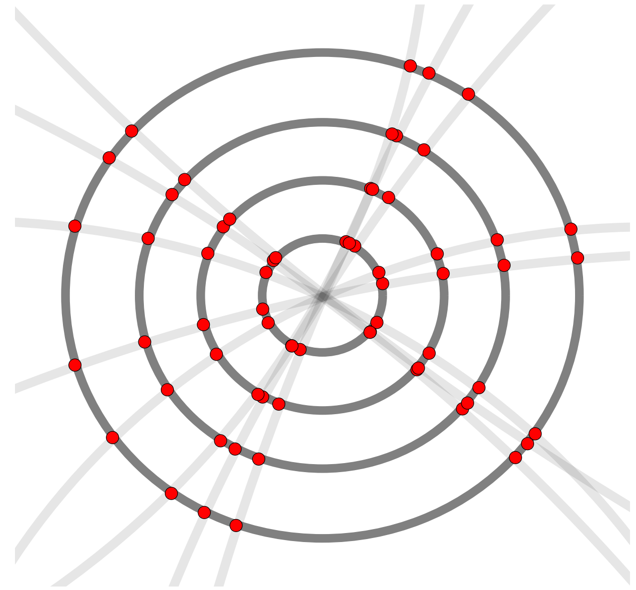
    - RNNs for language processing

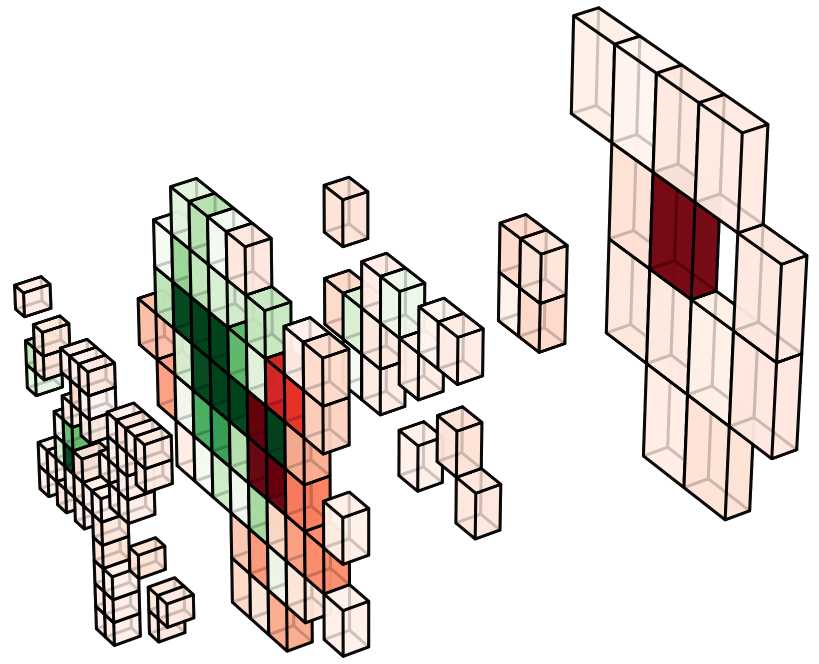


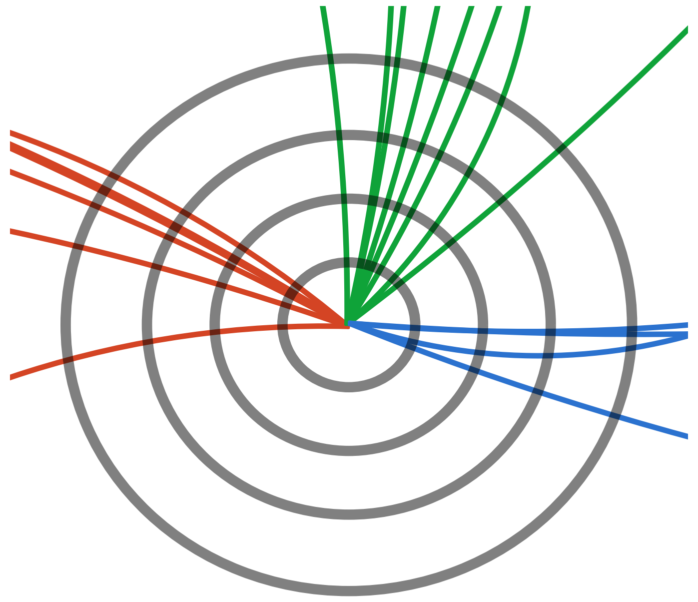
- What about physics data like jets?

Physics data

- Properties
  - Measurements distributed in space (and time) irregularly
  - Sparse (most detector channels are empty), but pockets of density
  - Complex interdependencies between measurements
  - Physics “objects” composed of multiple measurements
  - Inherent symmetries (Lorentz boosts, rotational)

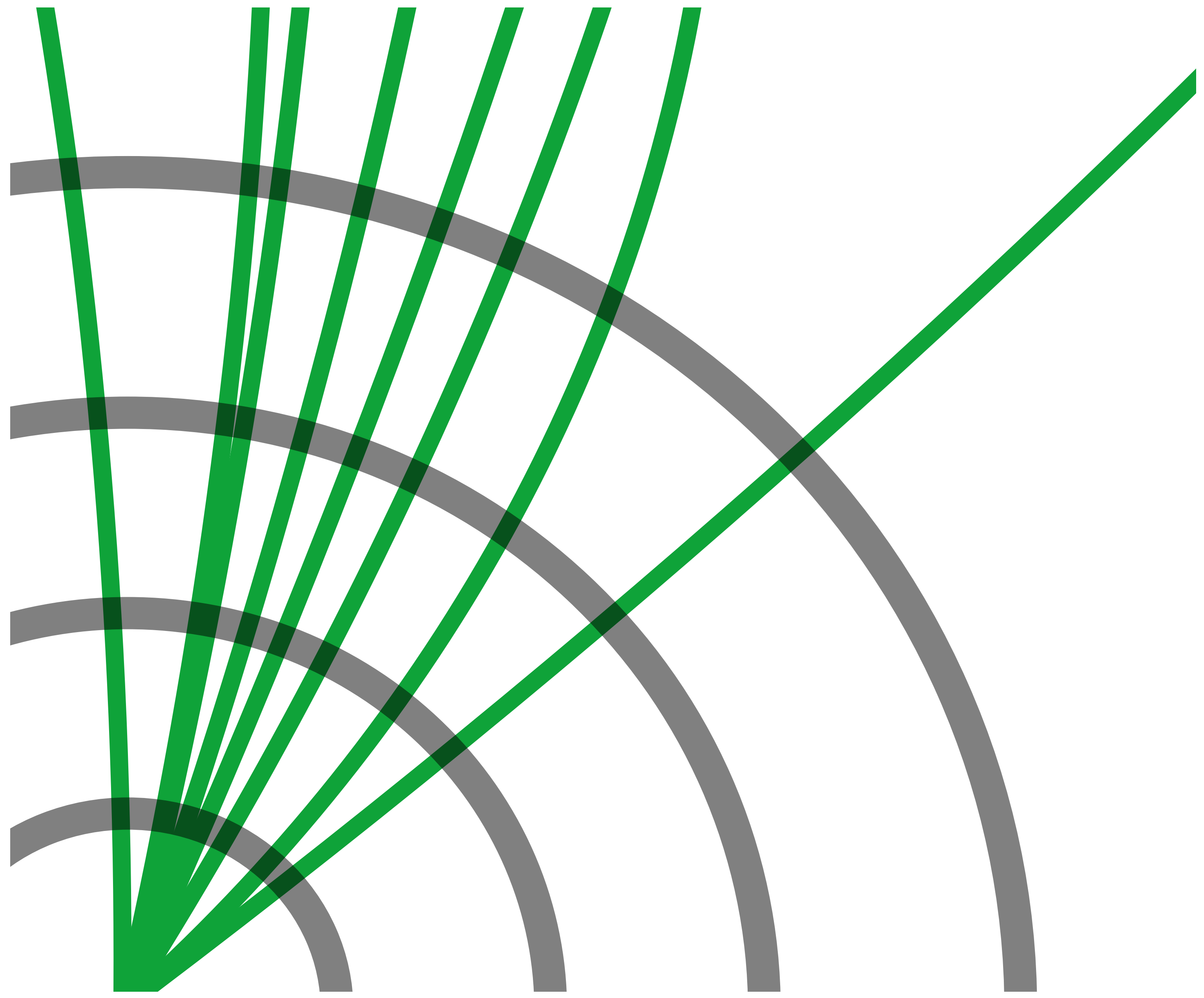






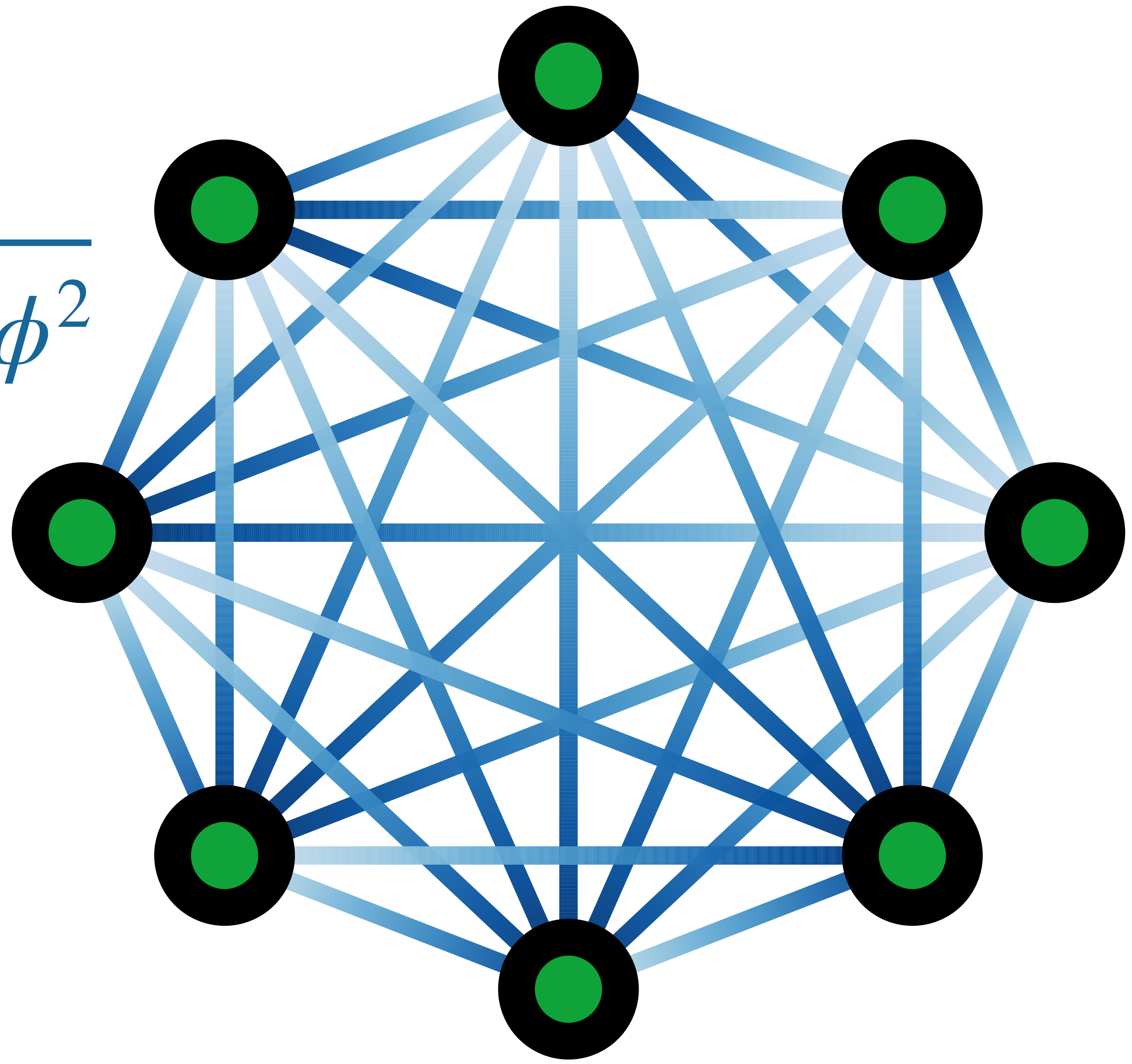
- Graph (or point cloud) embedding of the data can handle these properties

Node, edge, graph features (e.g. jet)



$$p = [E, p_x, p_y, p_z] \equiv [p_\mathrm{T}, \eta, \phi, m]$$

$$\Delta R = \sqrt{\Delta\eta^2 + \Delta\phi^2}$$



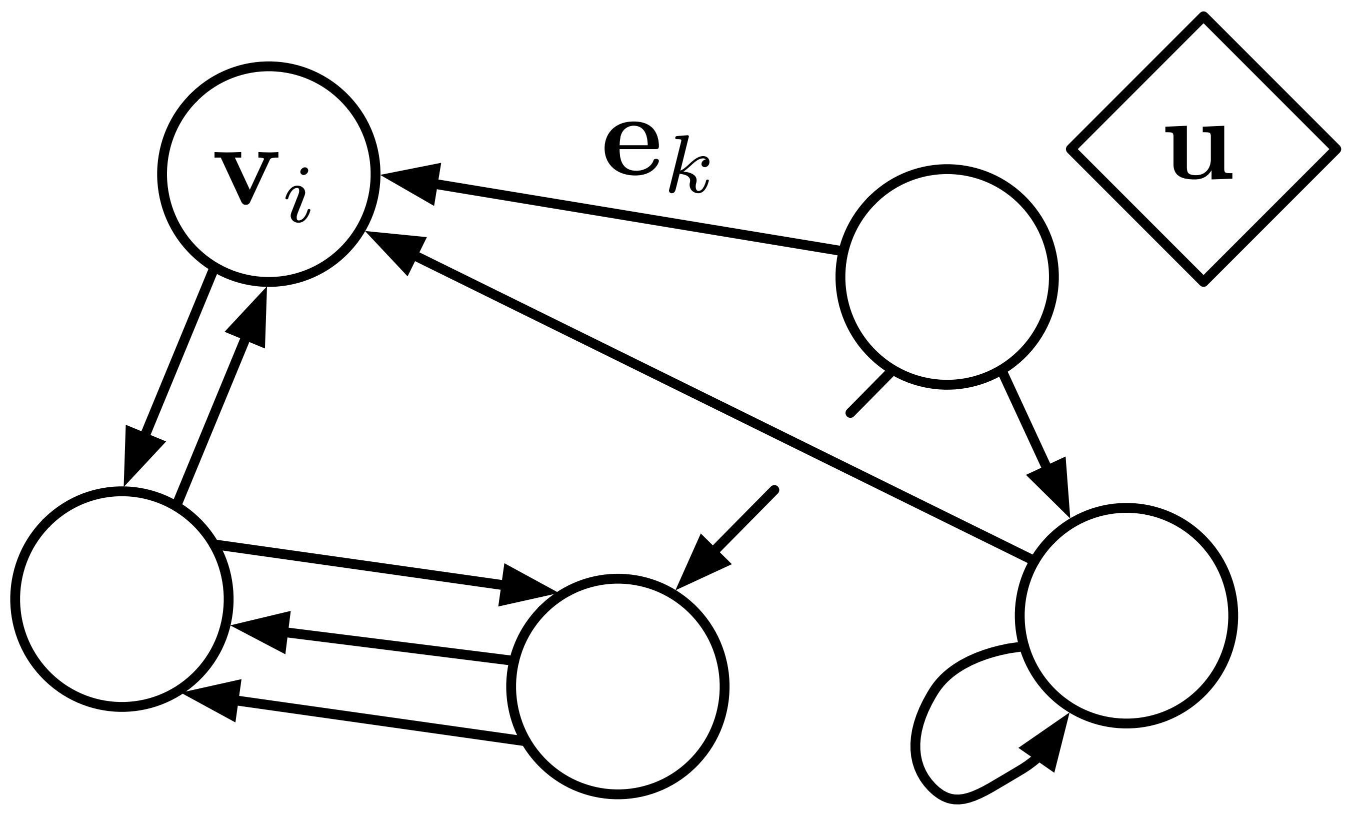
- Node features $\mathbf{v}_i$: particle 4-momentum
- Edge features $\mathbf{e}_k$: pseudoangular distance between particles
- Graph (global) features $\mathbf{u}$: jet mass

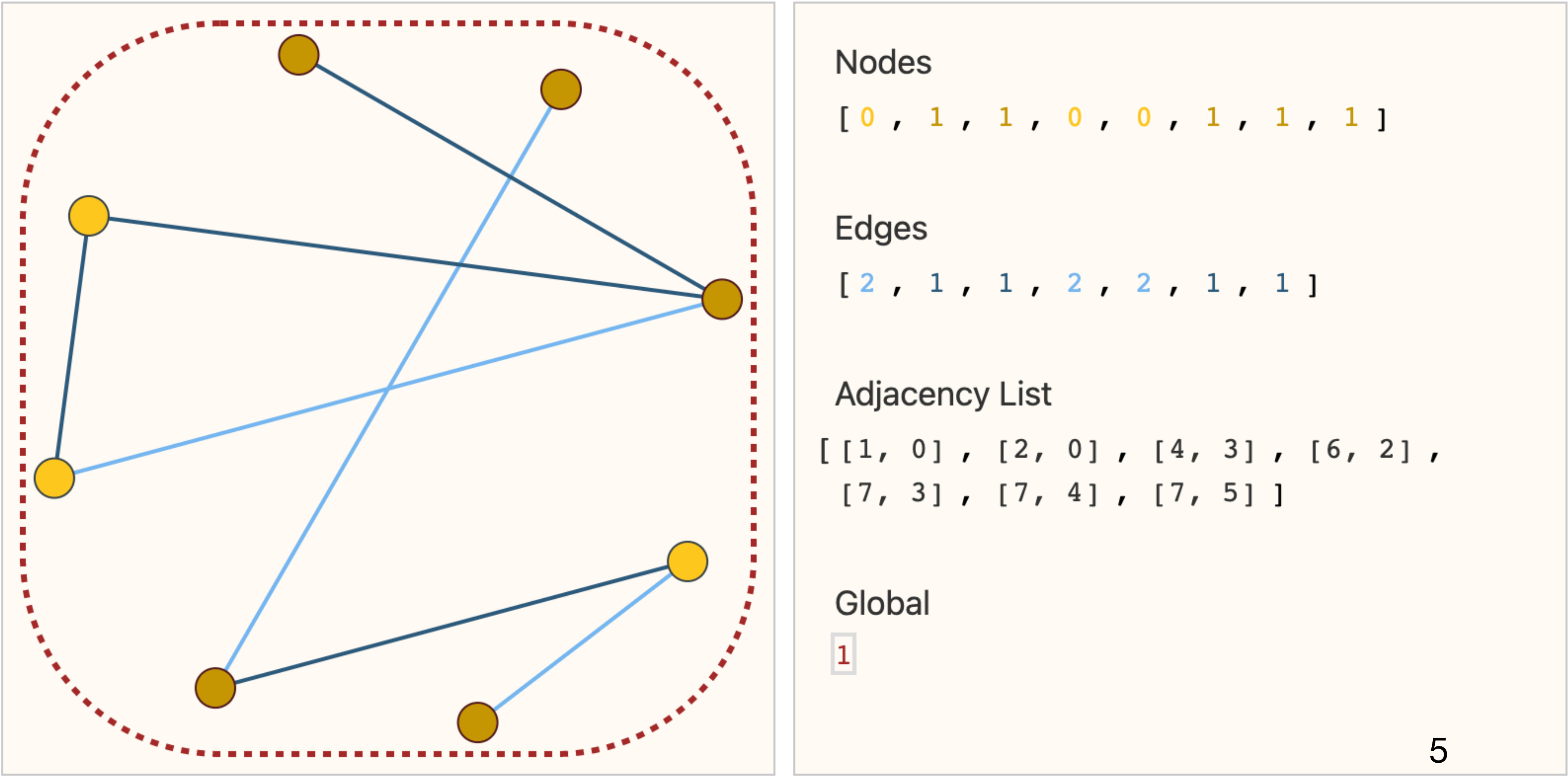
$$m = \sqrt{\sum_{i \in \mathrm{jet}} E_i^2 - p_{x,i}^2 - p_{y,i}^2 - p_{z,i}^2}$$

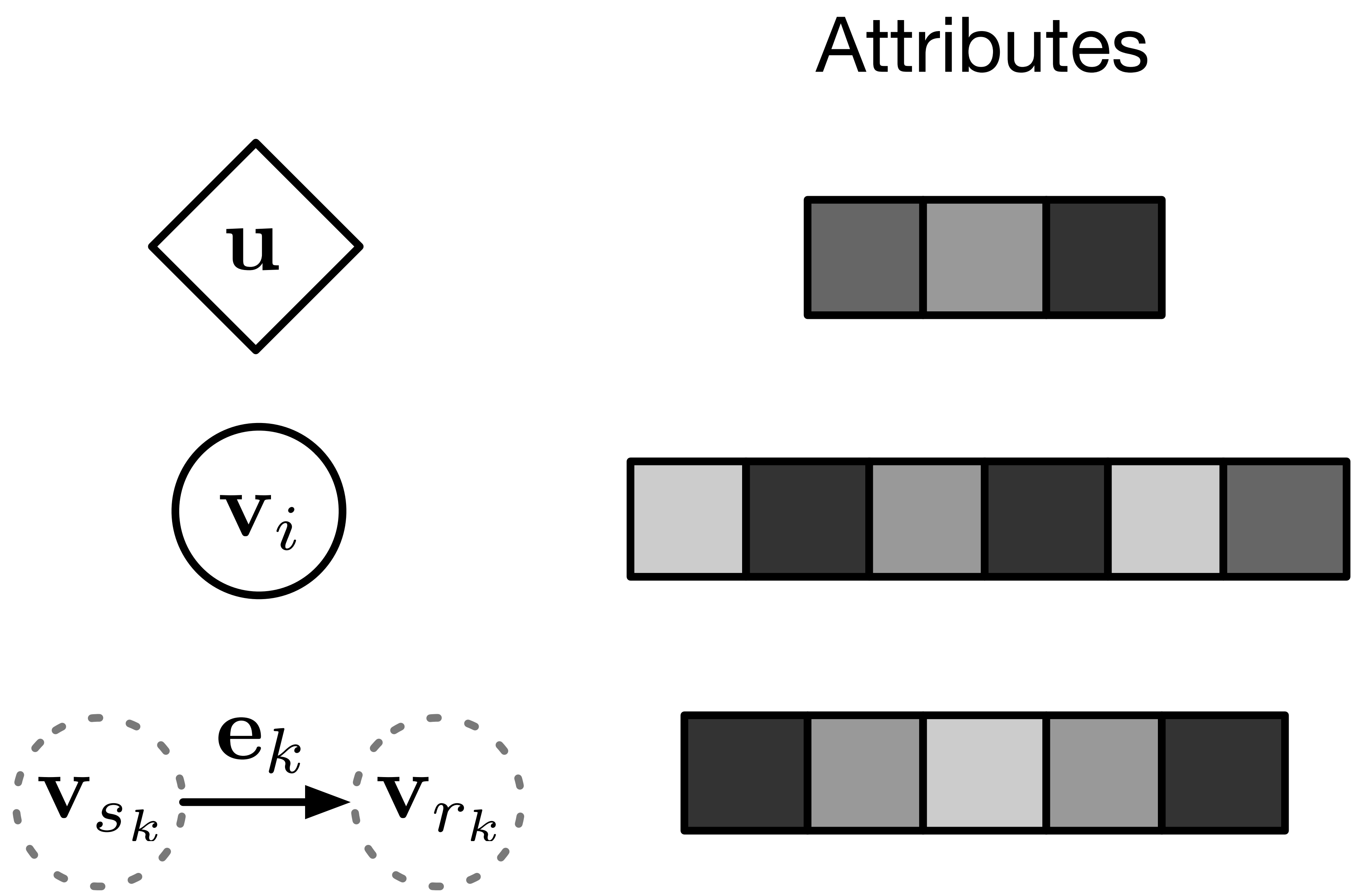
Formalizing a graph

https://distill.pub/2021/gnn-intro/

- Features: triplet of global features, node features, and edge features: $(\mathbf{u}, V, E)$
- Graph connectivity: adjacency matrix $A = \{a_{ij} = 1$ if i is connected to $j\}$
  - Sparse representation:
    "receiver" indices $\boldsymbol{r}$ and "sender" indices $\boldsymbol{s}$
    e.g. k[th] edge connects node s_k to node r_k



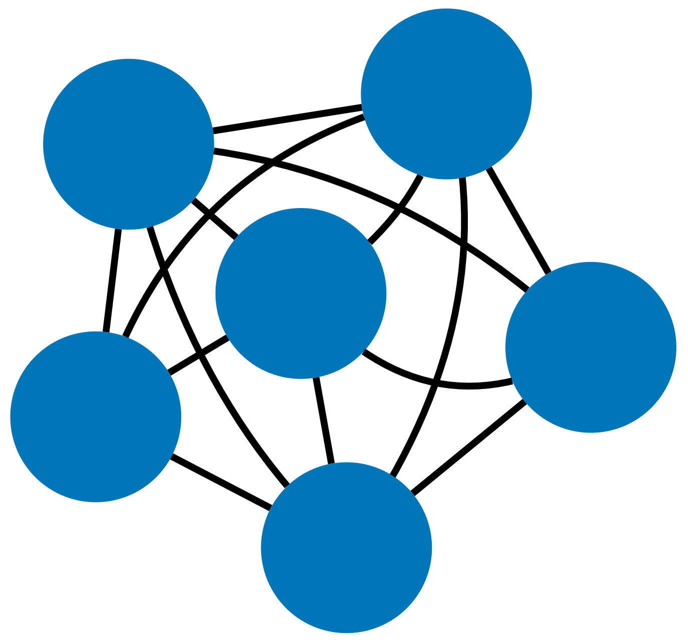




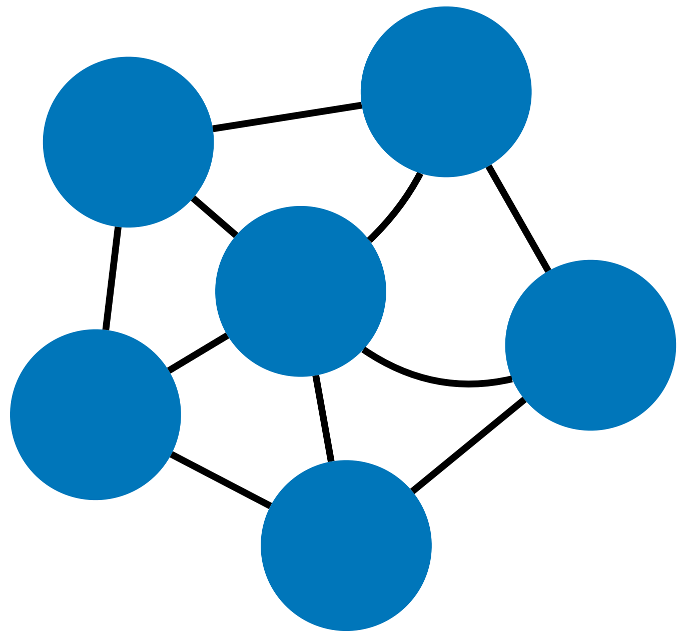
Graph connectivity

arXiv:2012.01249

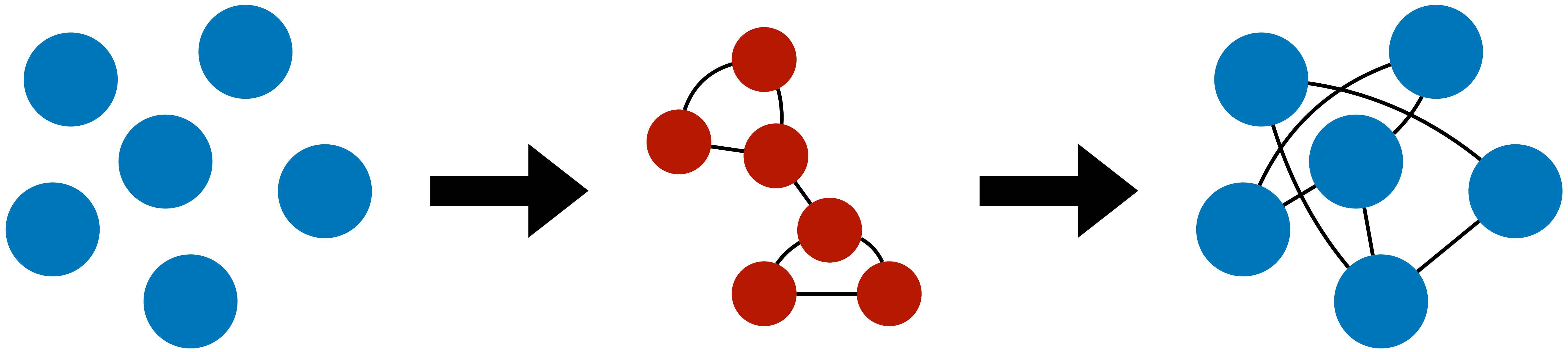
- Different methods for constructing the graph include:
  - connecting all pairs of nodes



  - connecting neighboring nodes in a predefined feature space



  - connecting neighboring nodes in a latent feature space



Inductive biases

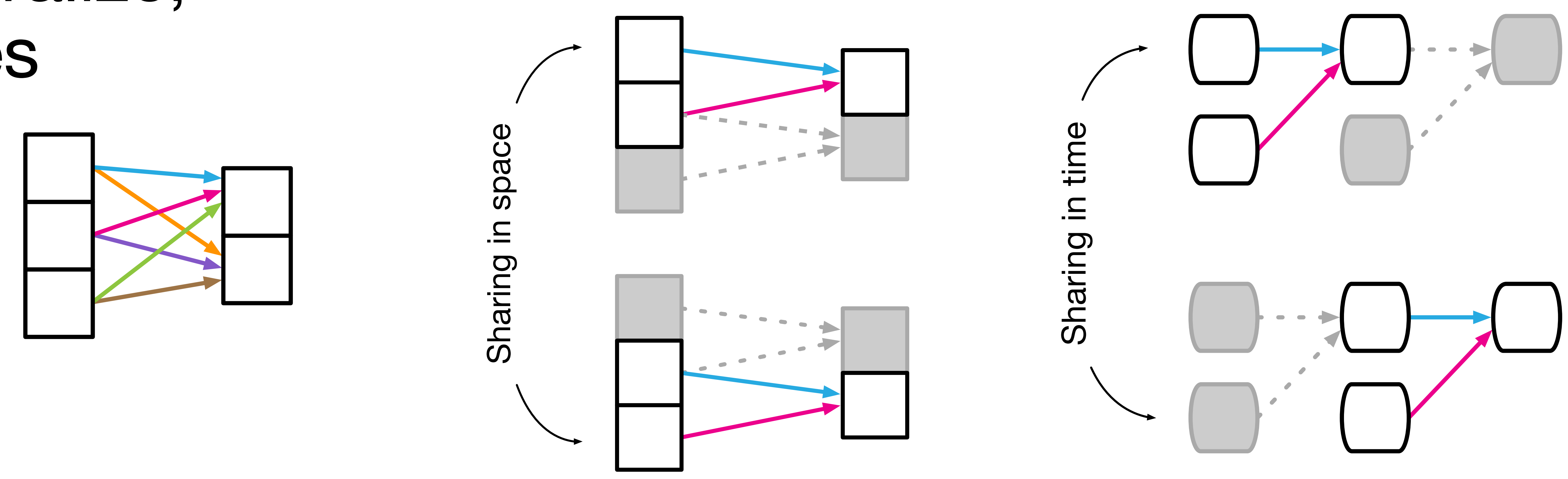
arXiv:1806.01261

Component	Entities	Relations	Rel. inductive bias	Invariance
Fully connected	Units	All-to-all	Weak	-
Convolutional	Grid elements	Local	Locality	Spatial translation
Recurrent	Timesteps	Sequential	Sequentiality	Time translation
Graph network	Nodes	Edges	Arbitrary	Node, edge permutations

- An **inductive bias** expresses assumptions about the data-generating process or the space of solutions, allowing a learning algorithm to prioritize one solution over another

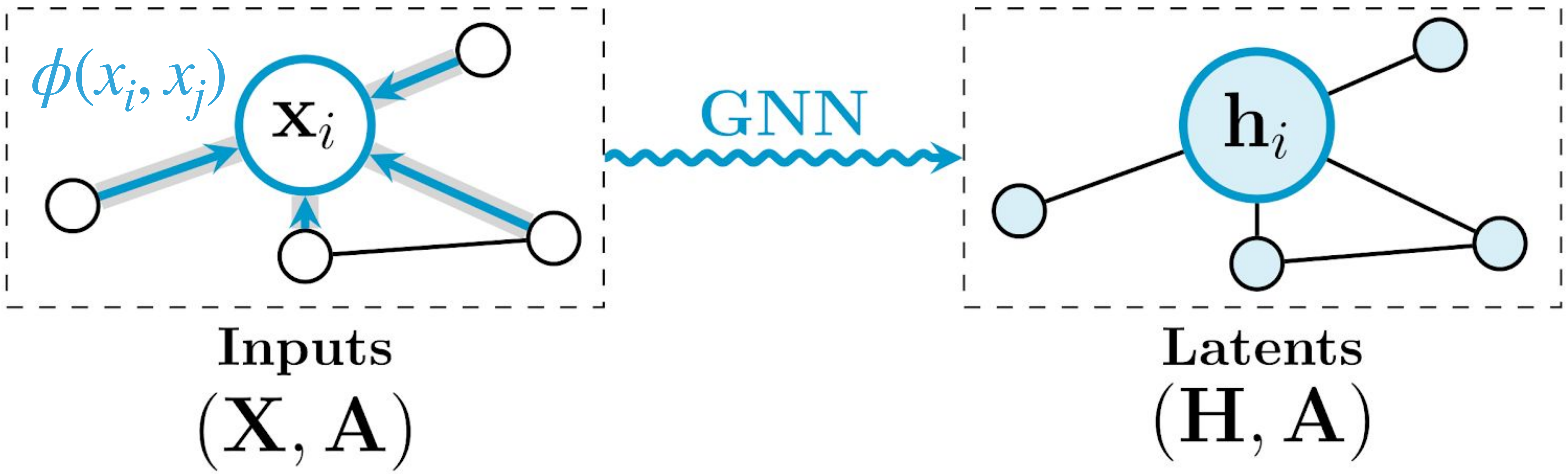
- Inductive biases can improve the search for solutions and find solutions that generalize; however, mismatched inductive biases can also lead to suboptimal performance by introducing constraints that are too strong

- **Relational inductive bias**: explicit representations of entities and relations



(a) Fully connected (b) Convolutional (c) Recurrent

GNN’s main ingredient: message passing

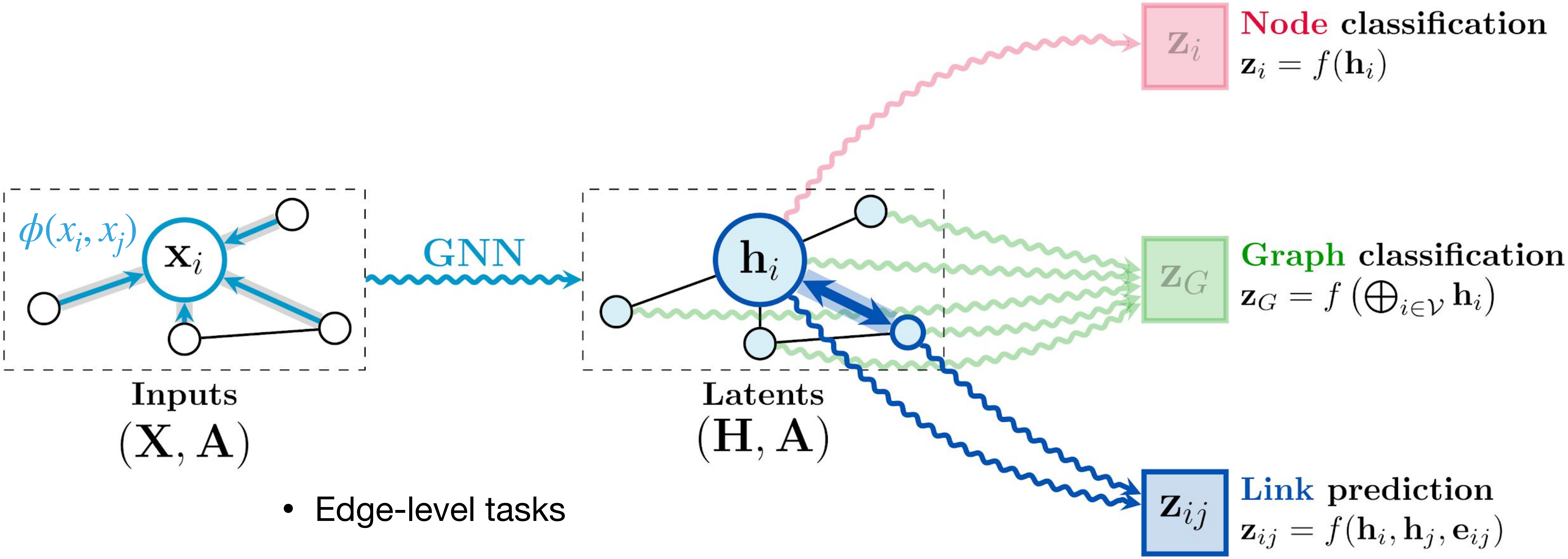


“message passing”

GNN tasks

Source: https://youtu.be/uF53xsT7mjc

- Node-level tasks
  - Identify "pileup" particles
- Graph-level tasks
  - Jet tagging



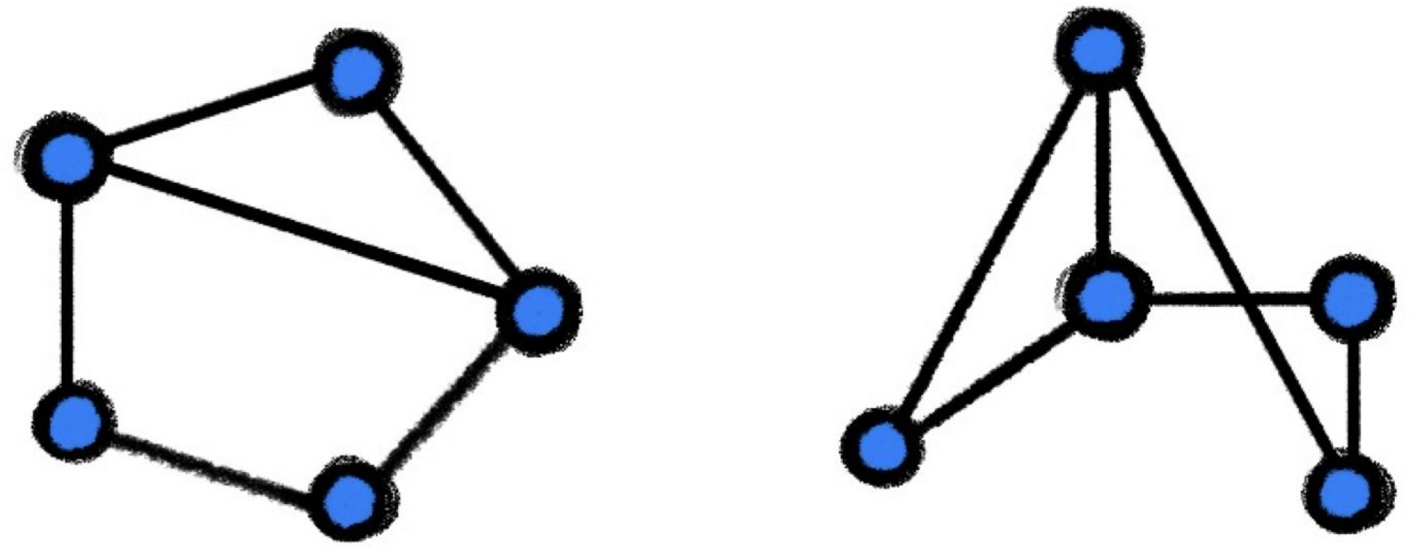
- Edge-level tasks
  - Identify good track candidates

Properties of GNNs

- Must have:
  - Graph-to-graph mappings valid for variable-size graphs $(\mathbf{u}, V, E) \rightarrow (\mathbf{u}', V', E')$
  - Graph-level outputs should be *invariant* under permutations of the nodes (node ordering should not matter) $f(V) = f(PV)$
  - Node- or edge-level outputs should be *equivariant* under permutations of the nodes (outputs should be permuted if inputs are permuted) $f(PV, PAP^T) = Pf(V, A)$
  - Learning uses "locality" (outputs of nodes in the same neighborhood should be more similar) $g(\mathbf{v}_i) \sim g(\mathbf{v}_j) \text{ if } \mathcal{N}_i \sim \mathcal{N}_j$
- Nice to have:
  - Account for nodes of different types ("heterogeneous graphs")
  - Generalizes to smaller/larger graphs than those seen in training
  - Computationally efficient

Recipe for GNNs: permutation invariance

- The nodes of a graph are not assumed to be in any order, i.e. we want the same result for two isomorphic graphs

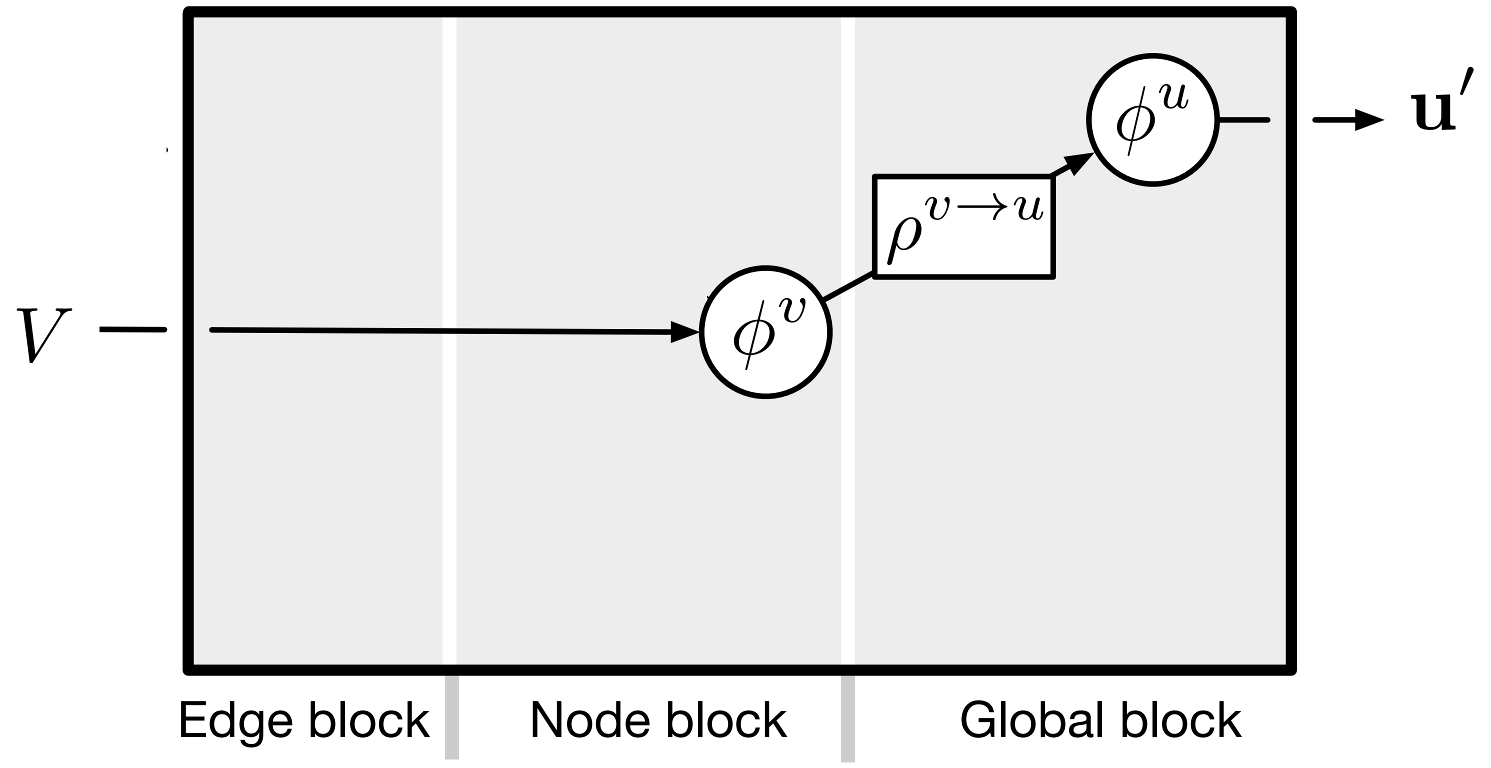


- First property we want is *permutation invariance*

$$f(V) = f(PV)$$

- One generic form is deep sets [arXiv:1703.06114] (a.k.a. particle/energy flow networks in HEP [arXiv:1810.05165]), where ϕ^v and ϕ^u are MLPs
  - Aggregation function $\rho^{v\to u}$ (other possible choices: e.g. min, max, or avg)
  - Limitation: ϕ^v function considers each node in isolation

$$f(V) = \phi^u\left(\sum_{i\in V} \phi^v(\mathbf{v}_i)\right)$$



Recipe for GNNs: permutation equivariance

- What if the algorithm output is not "global" but node-wise (e.g. we want to classify each particle in a collision)

$$f\left(\begin{bmatrix} p_{T1} & \eta_1 & \phi_1 \\ p_{T2} & \eta_2 & \phi_2 \\ p_{T3} & \eta_3 & \phi_3 \end{bmatrix}\right) = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

- Instead, we want functions that don't change the node order (i.e. if we permute the nodes, the outputs are also permuted)
  - Then we need *permutation equivariance*

$$f(PV) = Pf(V)$$

- But a graph is not just a set… The local connectivity is encoded in the adjacency matrix A,

$$a_{ij} = \begin{cases} 1 & \text{if } i \text{ is connected to } j \\ 0 & \text{otherwise} \end{cases}$$

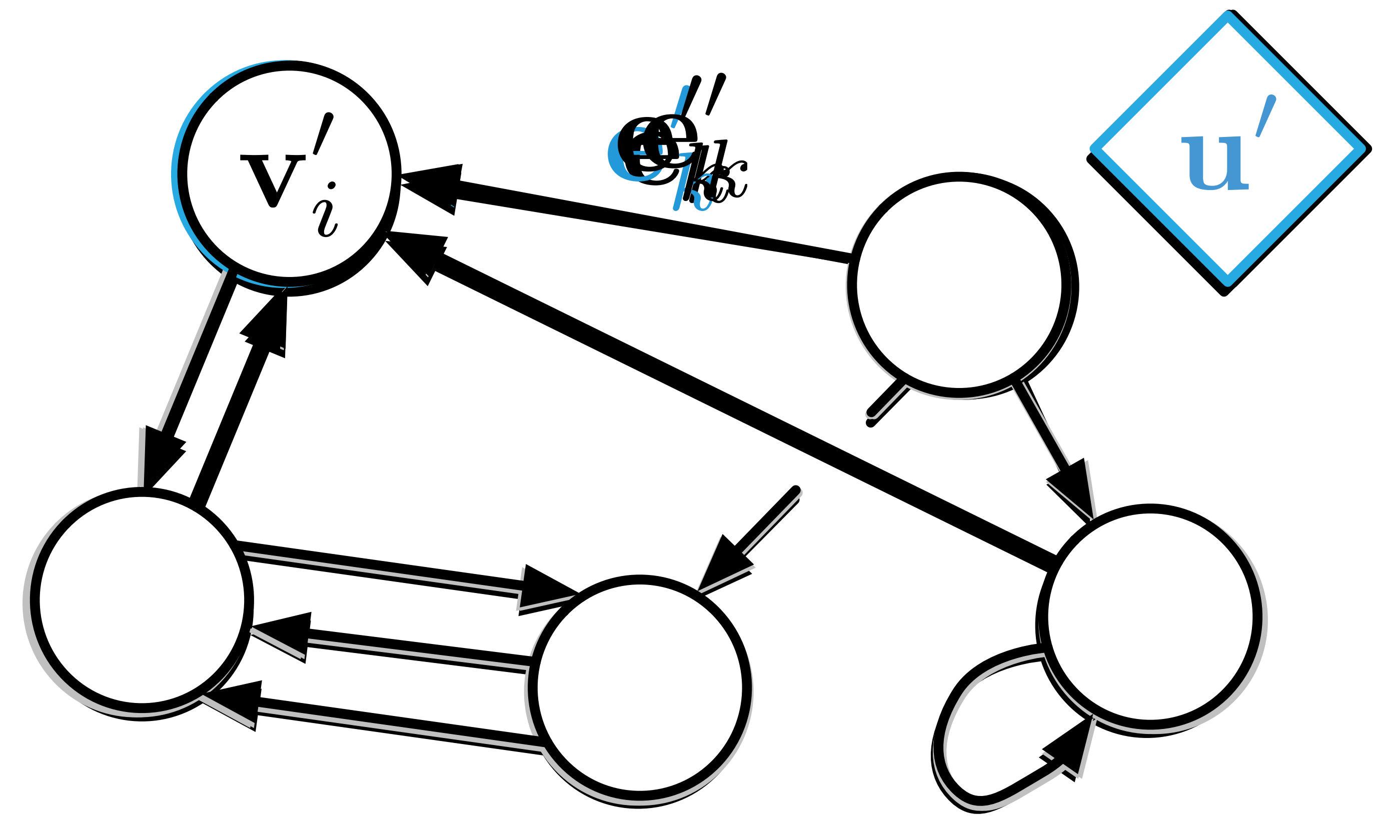
  - So full permutation equivariance condition is

$$f(PV, PAP^T) = Pf(V, A)$$

Full graph neural networks*

*One framework for GNNs: arXiv:1806.01261

- GNNs are graph-to-graph mapping (in this case holding structure fixed)
- Inference divided into three parts: edge block, node block, global block



$\mathbf{e}'_k$: message computed for edge k connecting nodes r_k, s_k

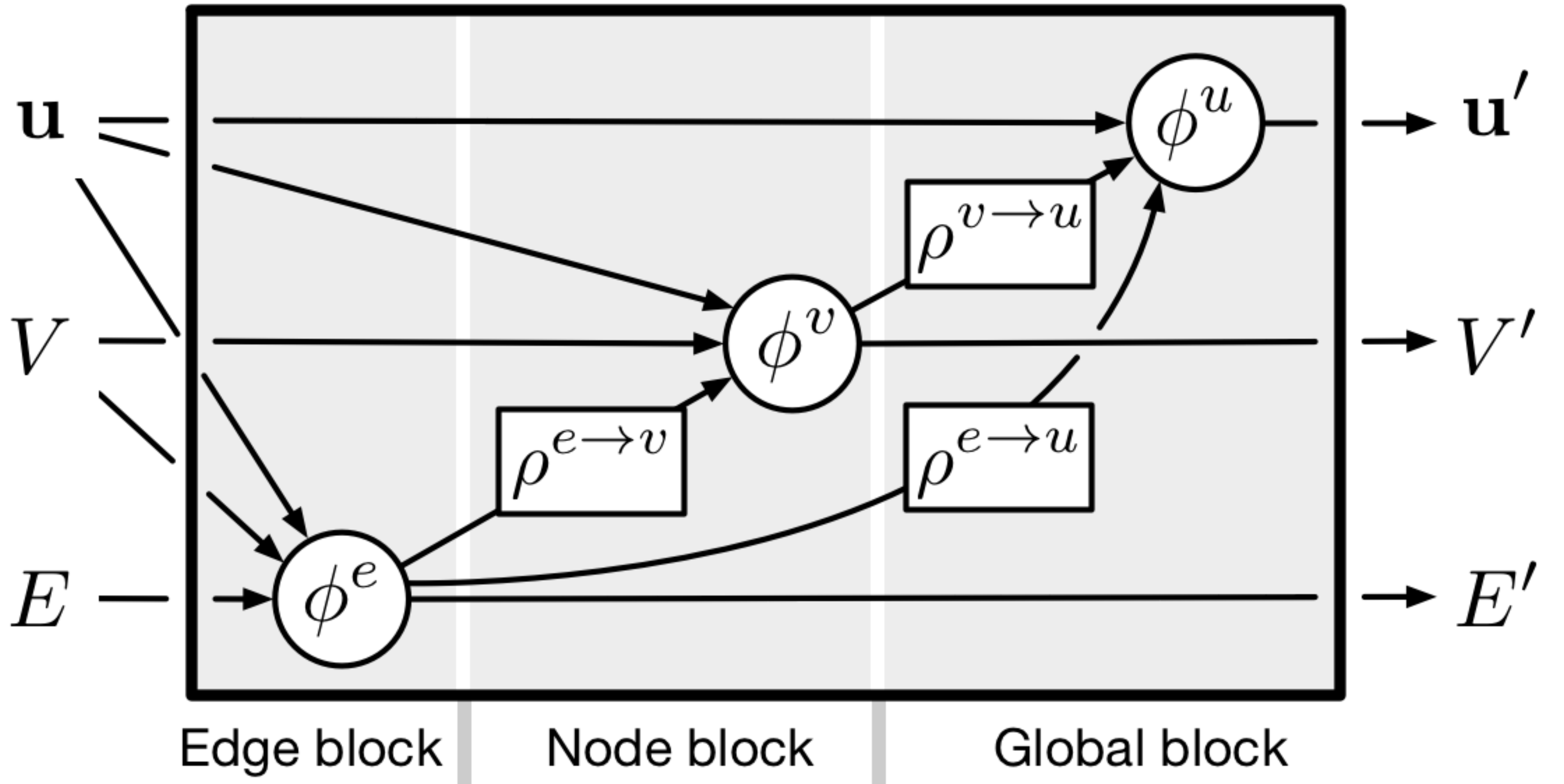
$\mathbf{v}'_i$: node feature update based on aggregated messages and previous features

$\mathbf{u}'$: global feature update based on aggregated, updated node and edge features

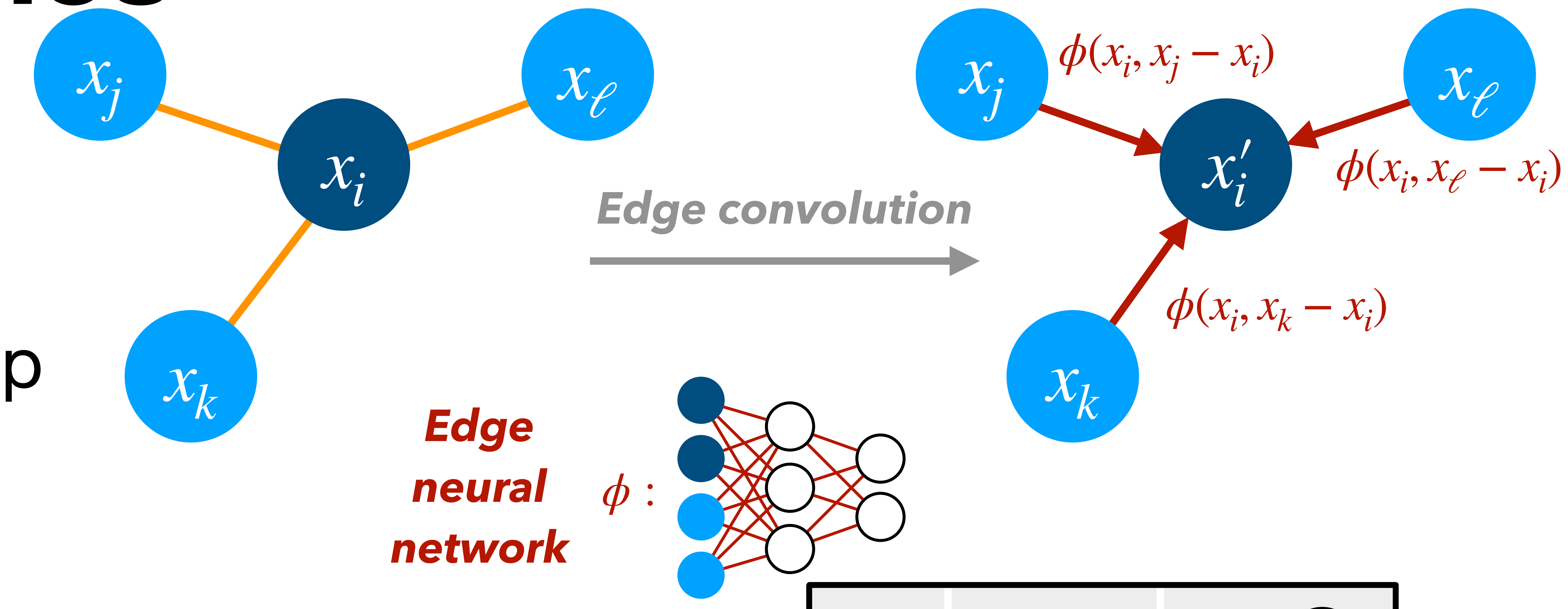
$$\mathbf{e}'_k = \phi^e(\mathbf{e}_k, \mathbf{v}_{r_k}, \mathbf{v}_{s_k}, \mathbf{u}) \qquad \bar{\mathbf{e}}'_i = \rho^{e\to v}(E'_i)$$

$$\mathbf{v}'_i = \phi^v\left(\bar{\mathbf{e}}'_i, \mathbf{v}_i, \mathbf{u}\right) \qquad \bar{\mathbf{e}}' = \rho^{e\to u}(E')$$

$$\mathbf{u}' = \phi^u(\bar{\mathbf{e}}', \bar{\mathbf{v}}', \mathbf{u}) \qquad \bar{\mathbf{v}}' = \rho^{v\to u}(V')$$



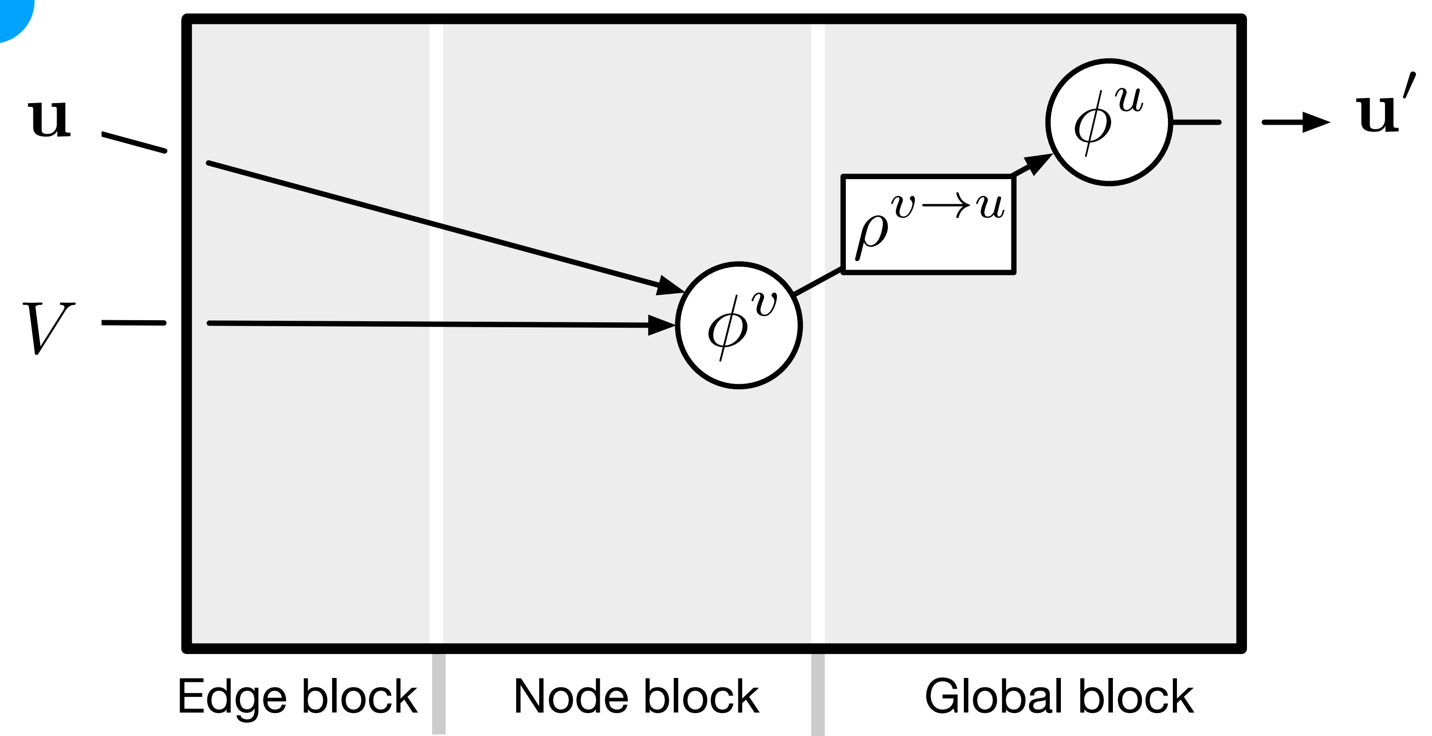
Other GNN examples

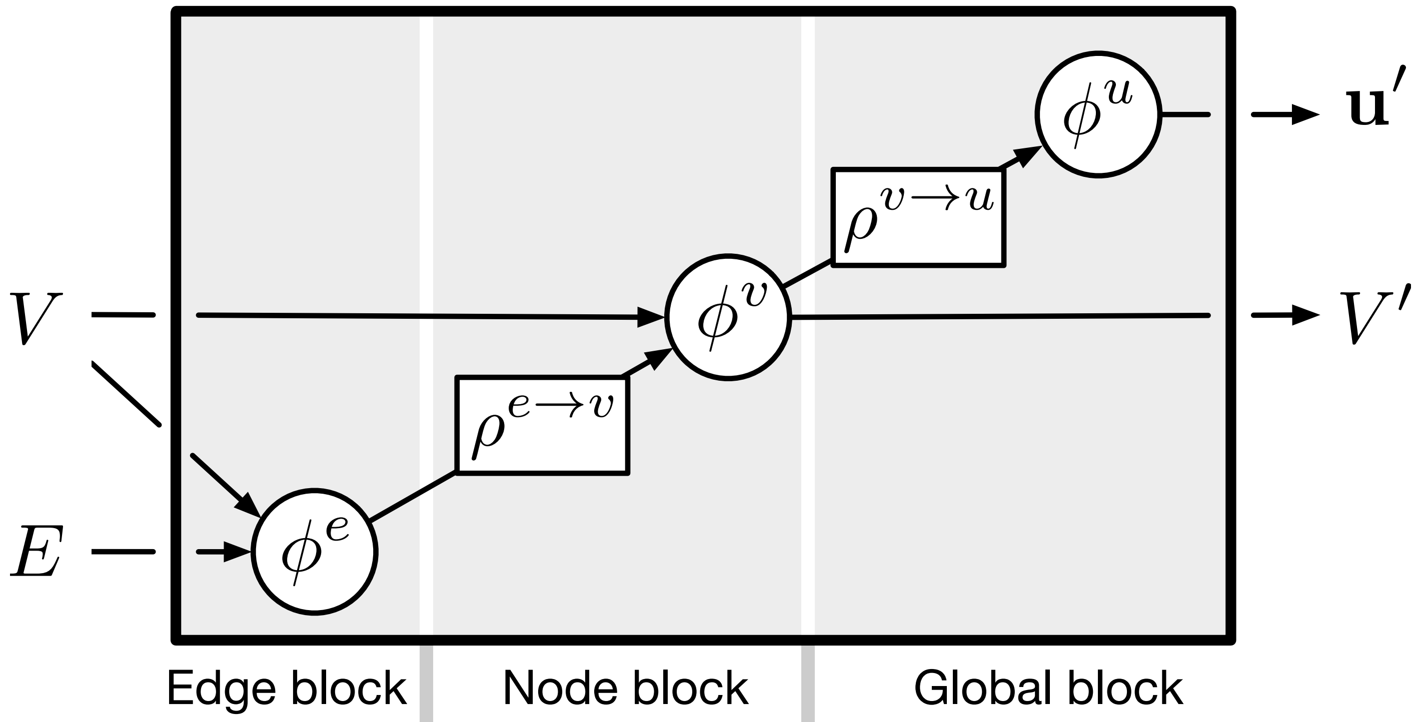


- Edge convolution from DGCNN [arXiv:1801.07829] is a variant of the edge block step (basis of ParticleNet [arXiv:1902.08570])

- Deep sets [arXiv:1703.06114] does not consider edge features (basis of energy flow network [arXiv:1810.05165])

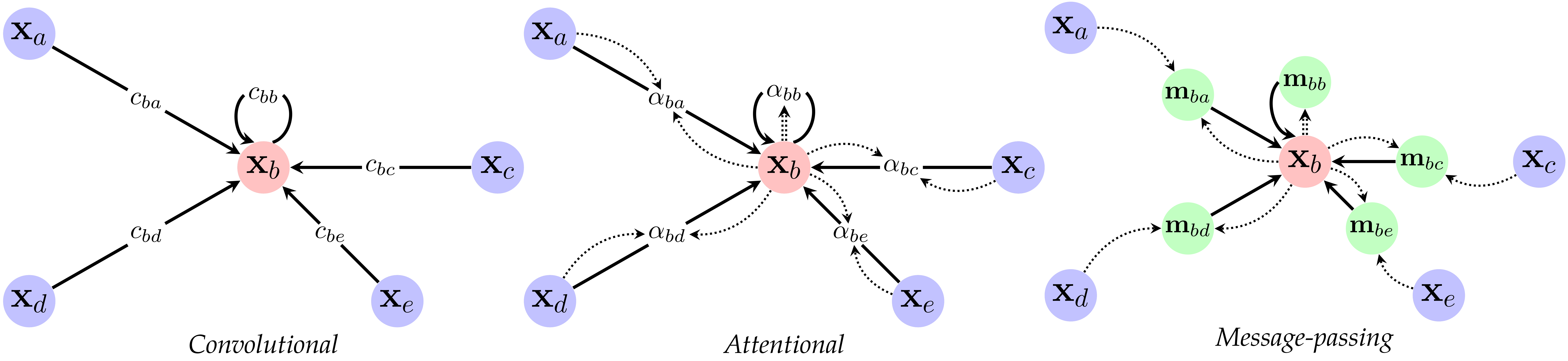
- Interaction network [arXiv:1612.00222] ignores global features (basis of jet taggers [arXiv:1908.05318, arXiv:1909.12285], and edge-classifying GNNs for tracking [arXiv:2003.11603])





GNN taxonomy

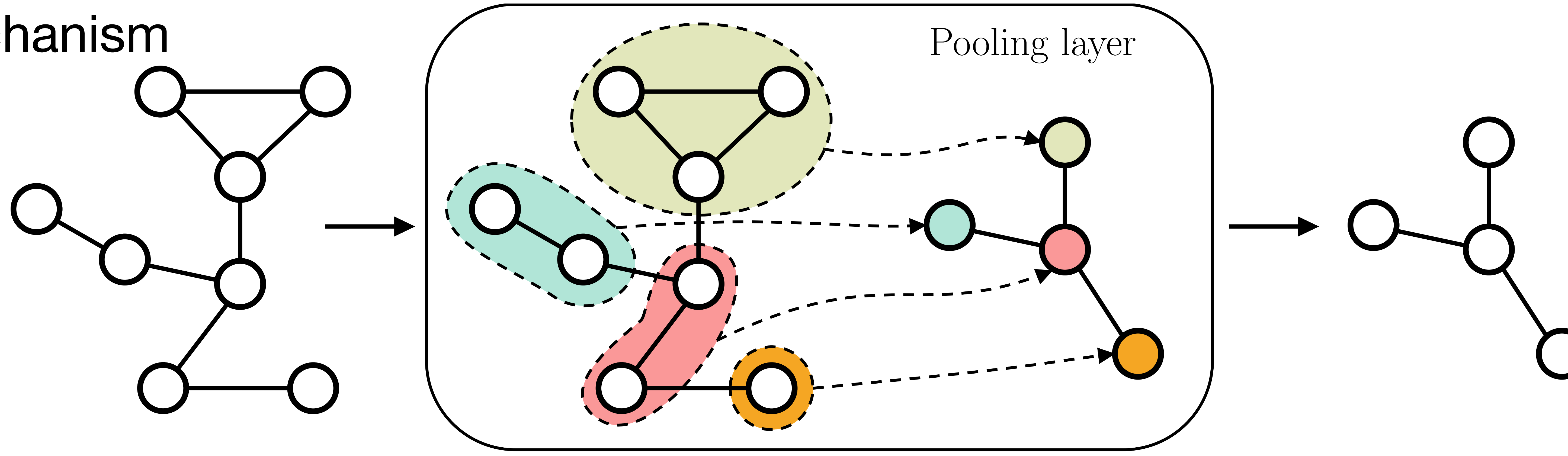
arXiv:104.13478



- Convolutional: sender node features are multiplied with a constant
- Attentional: multiplier is implicitly computed via an attention mechanism of the receiver over the sender
- Message-passing: vector-based messages are computed based on both the sender and receiver

Graph pooling

- Graph pooling layers “downsample” graphs to
  - discover important communities in the graph
  - imbue this knowledge in the learned representations
  - reduce the computational costs of message passing in large scale structures
- Two broad classes: adaptive and topological
- Adaptive: parametric, trainable pooling mechanism
  - Differentiable pooling
  - Top-k pooling
  - Self-attention graph (SAG)
  - Edge pooling

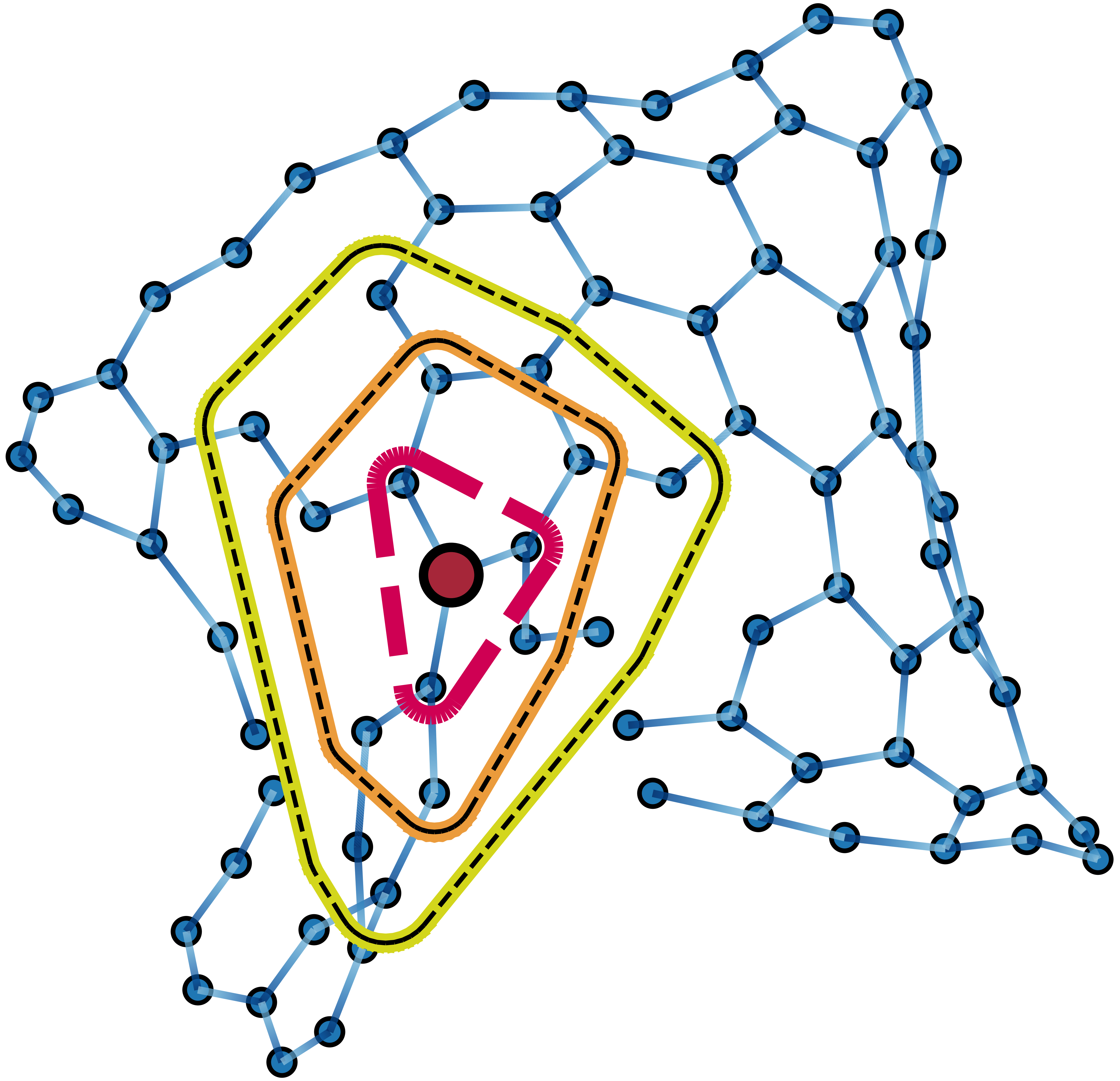


- Topological: not required to be differentiable, leverage the structure of the graph itself
  - GRACLUS
  - Nonnegative matrix factorization pooling

Receptive field in GNNs

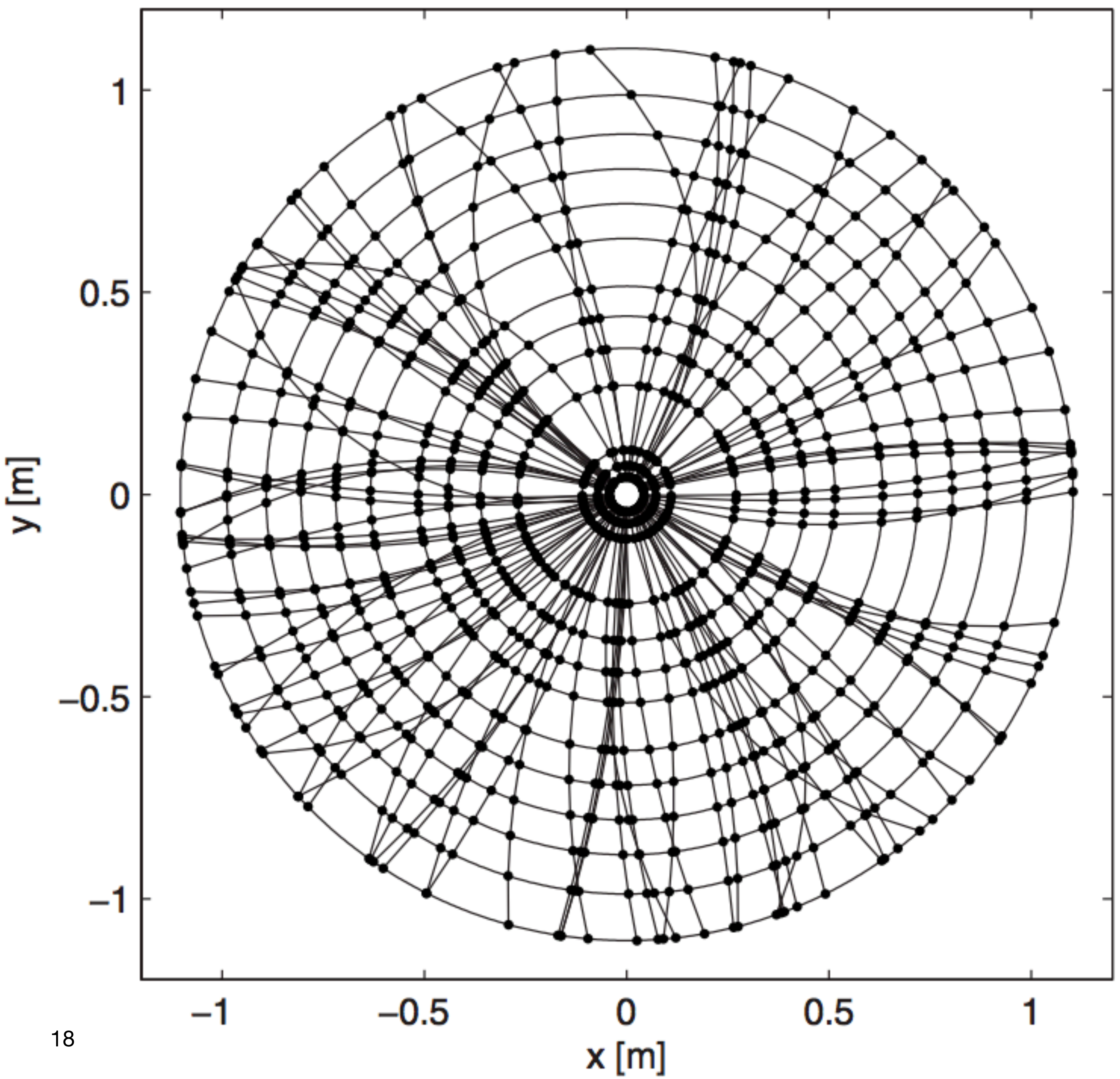
- Red, orange, and yellow boundaries represent the enlarging neighborhood of nodes that may communicate with the red node after one, two, and three iterations of message passing, respectively

- Nodes outside of the yellow boundary do not influence the red node after three iterations



Particle tracking (connecting the dots)

- Particle tracking is a classic reconstruction task
- From a set of hits sampled sparsely in 3D, reconstruct the helical trajectories of particles
- Traditional algorithms scale badly with the number of hits
- GNNs may be able to do better [arXiv:1810.06111, arXiv:2003.11603, arXiv:2007.00149]

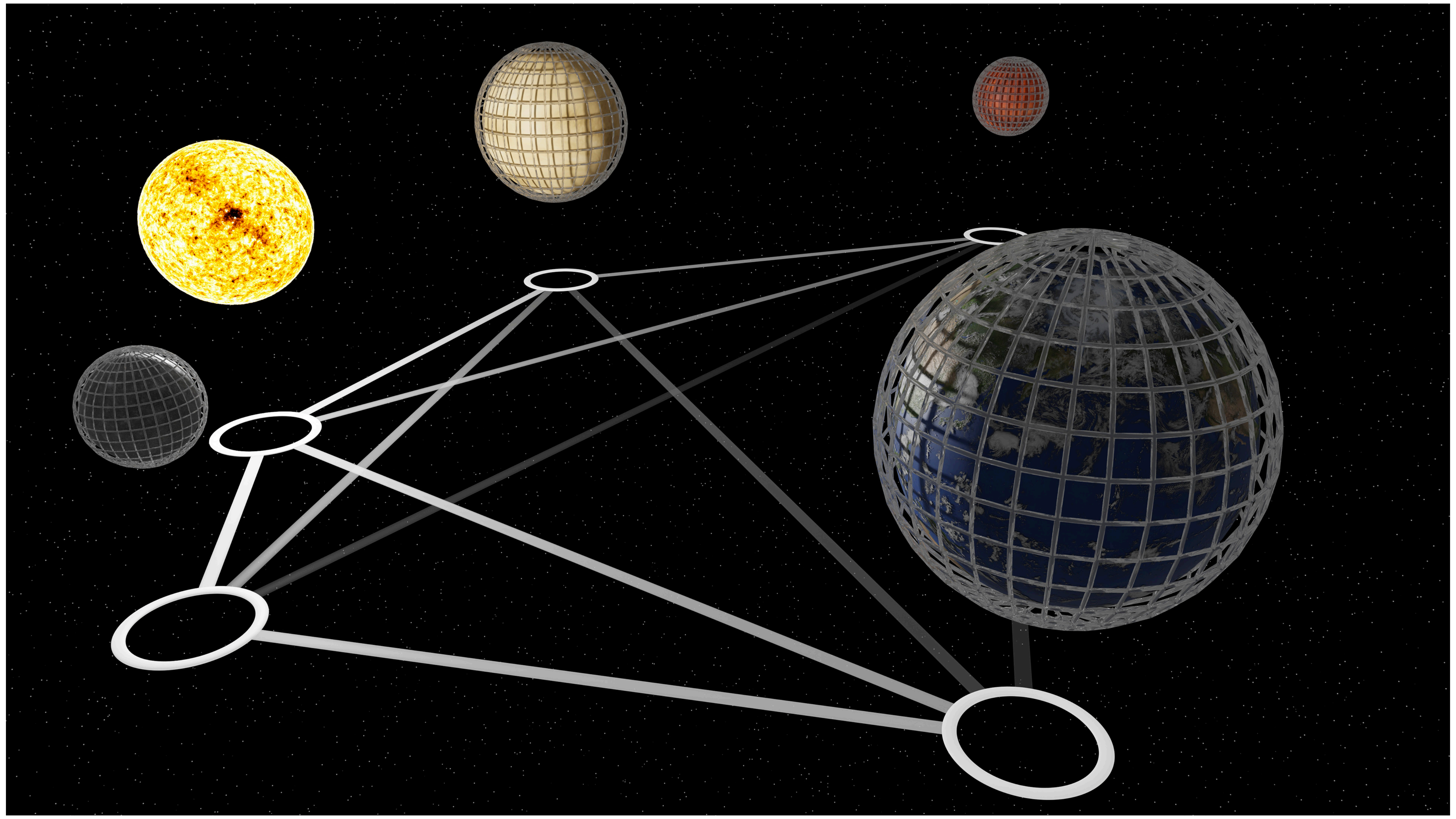


Predicting orbital mechanics

- Interaction network model can simultaneously learn the rules governing interactions and free parameters (i.e. masses)

$$\mathbf{F} = -\frac{GMm}{r^2}\hat{r}$$

https://astroautomata.com/paper/rediscovering-gravity/
https://drive.google.com/file/d/1PqMQKsHYeDgEQhP9darVqs5UXiaoyPzx/view



Next time

- More on GNNs
- If time, transformers
- Hands-on: GNNs