Intro to Graph Neural Networks from a HEP perspective

Joosep Pata (joosep.pata@cern.ch)
NICPB/KBFI, Tallinn, Estonia

Lecture at
QU Data Science Basics (Hamburg)
May 25, 2021

Eesti Teadusagentuur
Estonian Research Council

This work is supported by the Mobilitas Pluss Returning Researcher Grant MOBTP187 of the Estonian Research Council.

Message Passing Networks

Operations on a graph

Jet tagging with a simple neural network

Interaction Networks

Graph Attention Networks

Graph Convolutional Networks

Dynamic Graphs: ParticleNet

heterogeneous graphs

Graph Structure Learning

normalization, regularization

Sparse graph models: GravNet and detector clustering

generative models

graph sampling

Scalable graph models: particle flow

transductive learning

very large graphs (>100000 nodes)

production use

Jet tagging

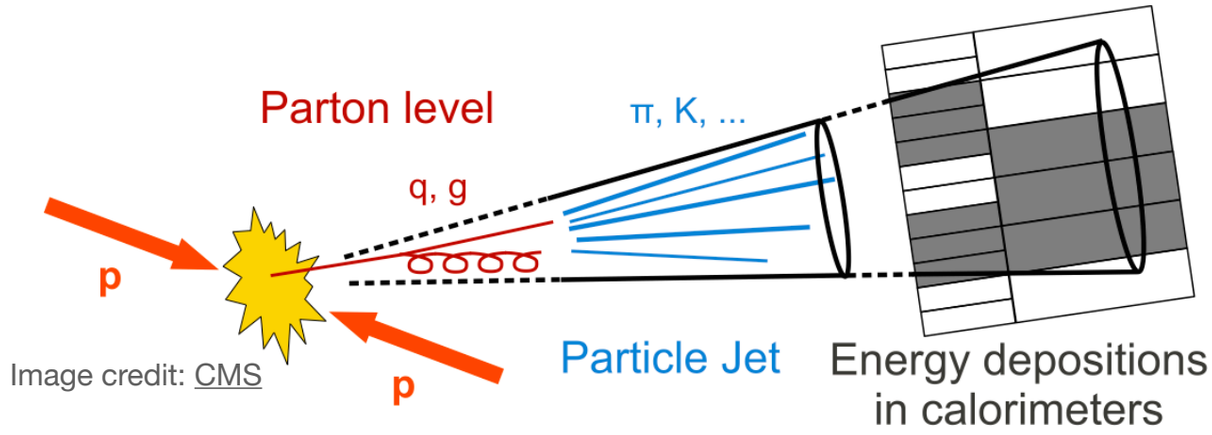
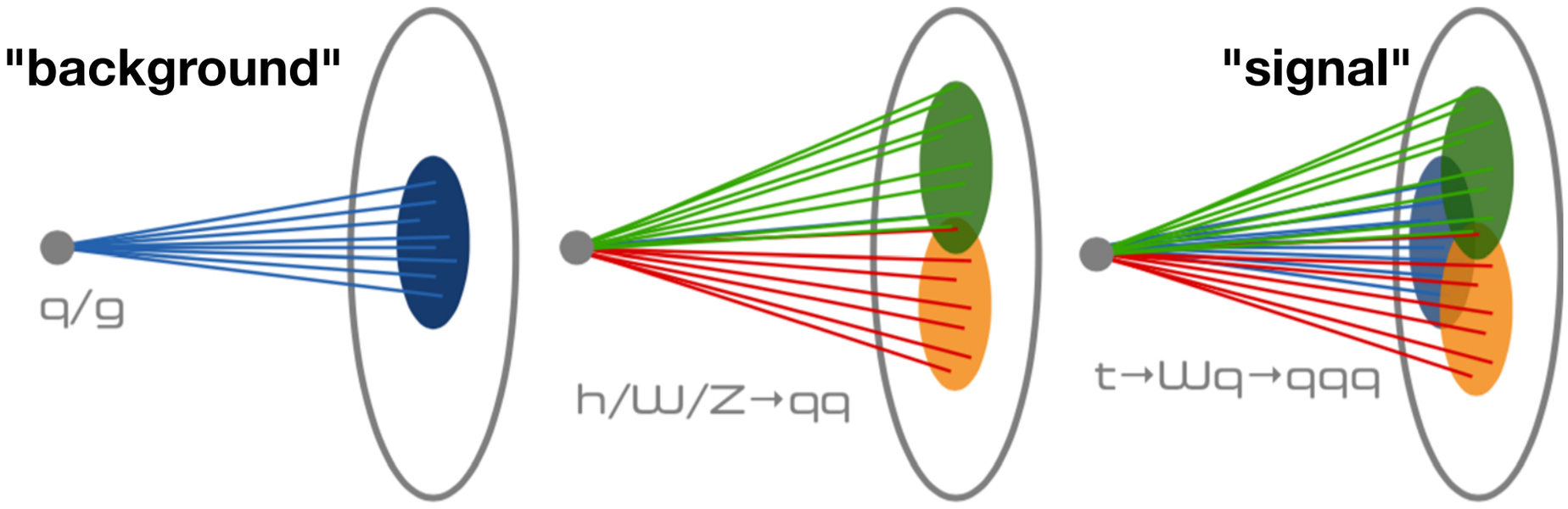


Image credit: CMS



jet constituents: set of particles
{..., (pT, η, ϕ, particle ID), ...}

→

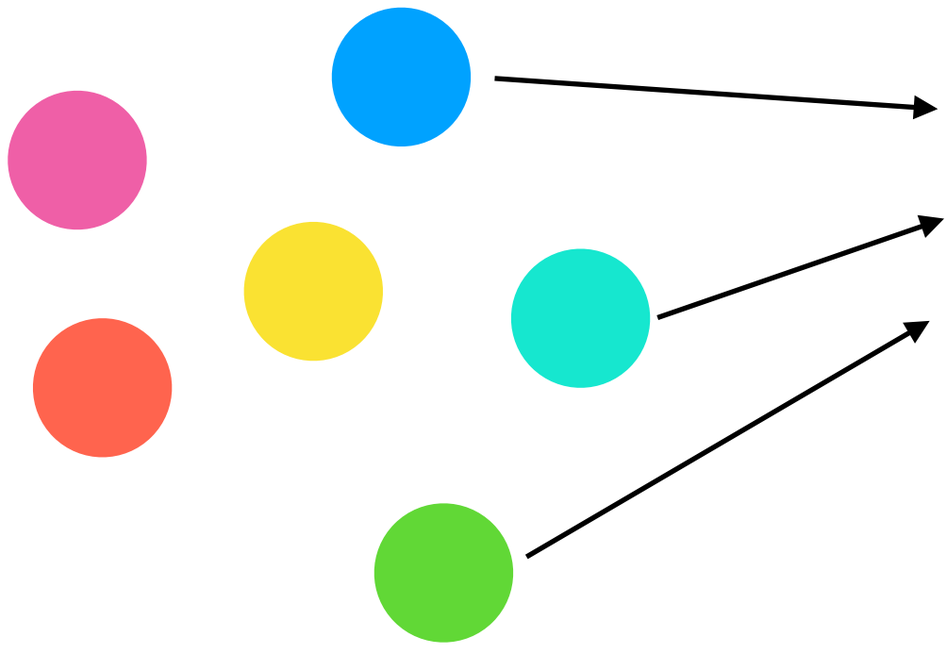
target:
jet originator particle ID

Moreno, E.A., Cerri, O., Duarte, J.M. *et al.* JEDI-net: a jet identification algorithm based on interaction networks. *Eur. Phys. J. C* **80,** 58 (2020). https://doi.org/10.1140/epjc/s10052-020-7608-4

Data representation

set of inputs with N constituents, M features

{..., (pT, η, ϕ, particle ID), ...}



jet constituents

feature matrix (N, M)

pT (GeV)	η	ϕ	particle ID
12.3	*1.2*	*0.5*	*pi+*
11.8	*1.24*	*0.45*	*K0*
10.4	*1.18*	*0.43*	*pi-*
9.8	*1.39*	*...*	*e-*
6.4	*...*	*...*	*...*
5.3	*...*	*...*	

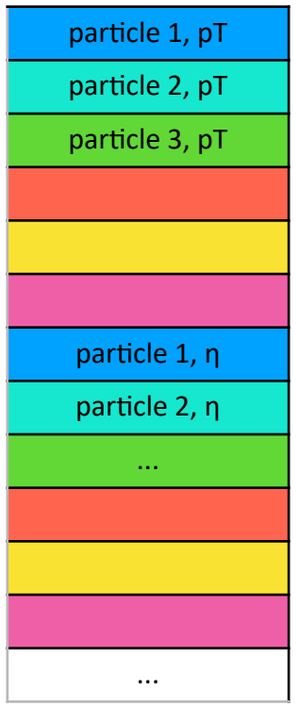
Set of feature vectors + ordering → feature matrix

Simple neural network

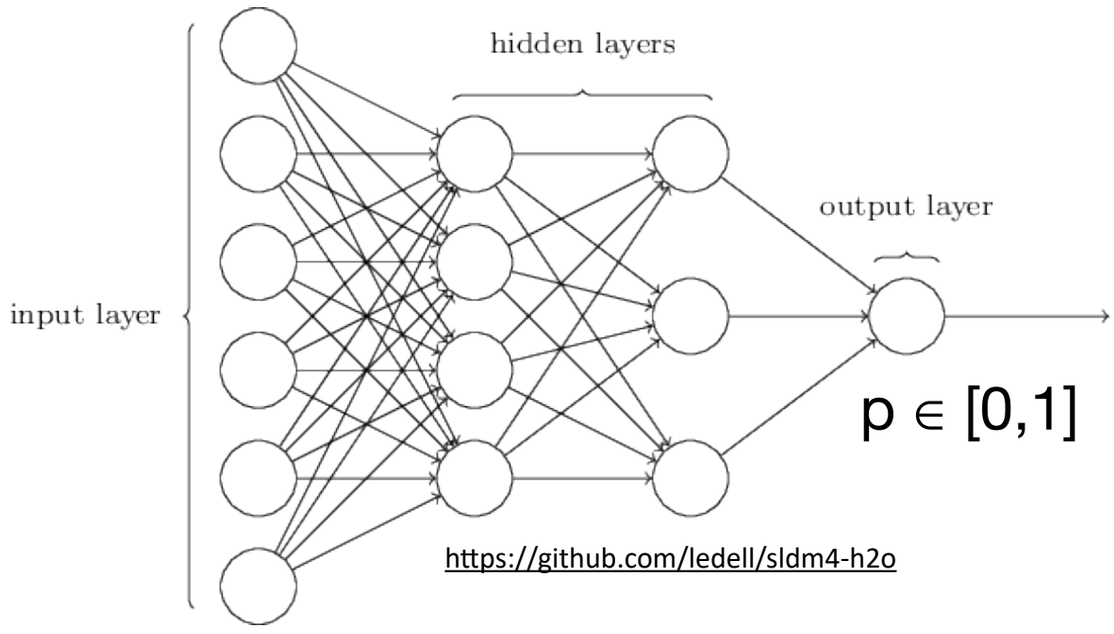
feature matrix (N, M)

pT (GeV)	η	ϕ	particle ID
12.3	1.2	0.5	pi+
11.8	1.24	0.45	K0
10.4	1.18	0.43	pi-
9.8	1.39	...	e-
6.4	...	...	...
5.3	...	...	

flat NxM feature vector



map feature vector to an output



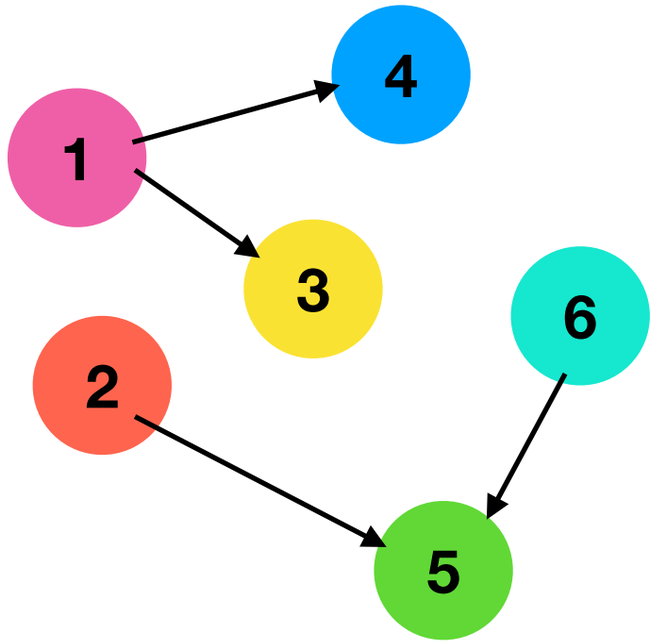
Order: The ordering is important! A feedforward network trained with e.g. pT-descending ordering would not work with pT-ascending. Which ordering is optimal?

Representation: What if for each jet you want to classify, the number of constituents N varies? Need to make all feature matrices the same size (e.g. with 0 padding).

Structure: All-to-all connectivity. Every constituent in the input layer can affect every other constituent in the next layer.

Graph structure

graph = set of nodes/vertices/elements + edges between them



Edges represented as a index pairs
edges = [(1,4), (1,3), (2,5), (6,5)]

Or as an NxN adjacency matrix
(possibly sparse)

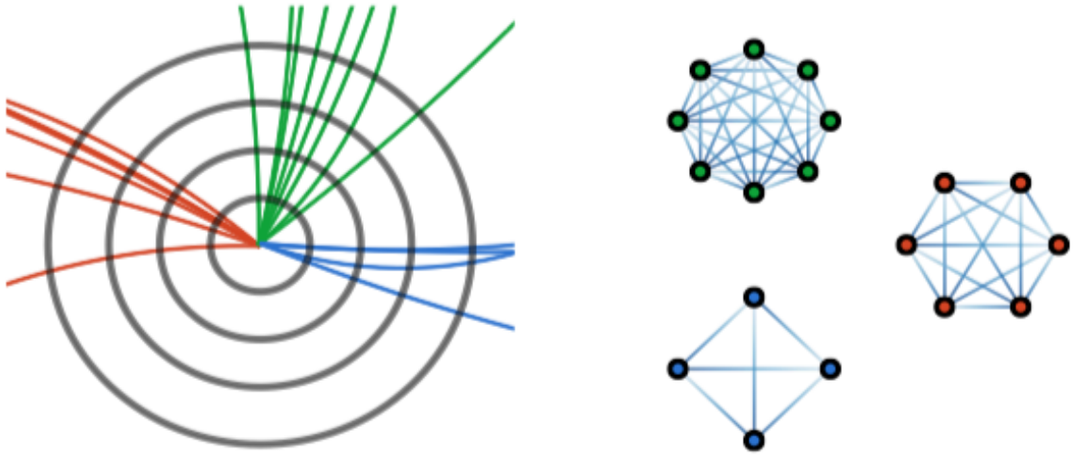
$$A_{ij} = \begin{bmatrix} 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Where do we get this graph structure?

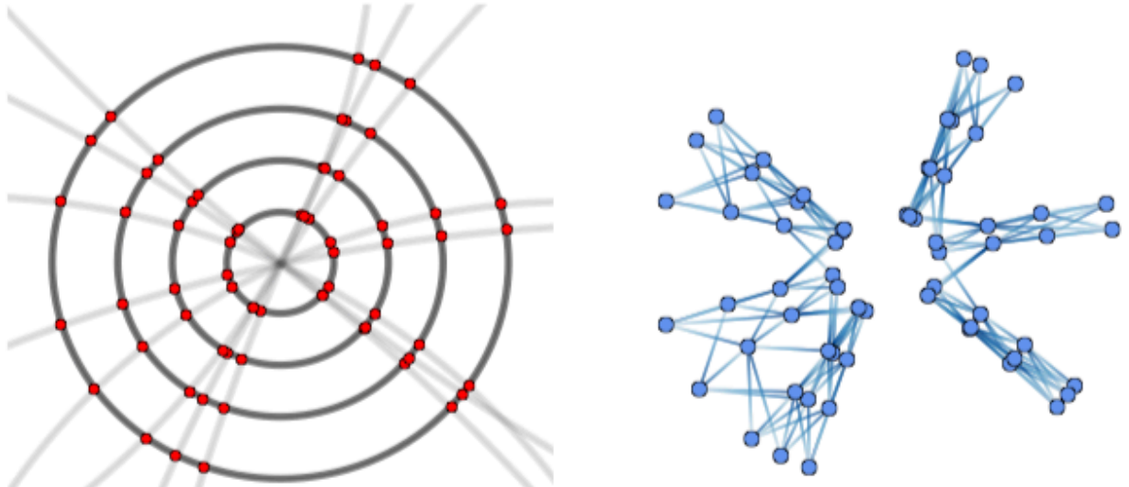
1. All-to-all connections, in case of small input sets.
2. From physics priors: connect "nearby" elements in advance
3. Optimize as a part of the learning process (Graph Structure Learning)

Example graph structures

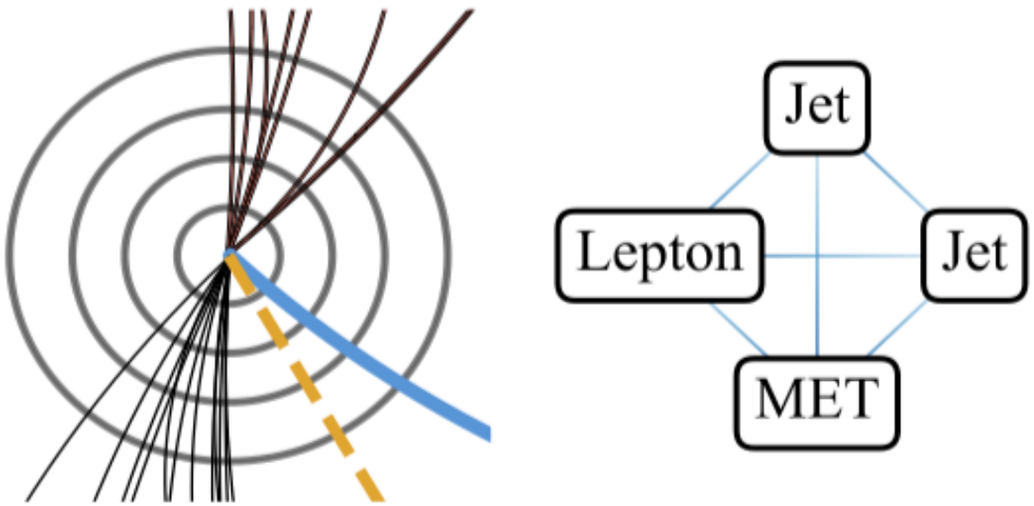
Jet constituents (all-to-all)



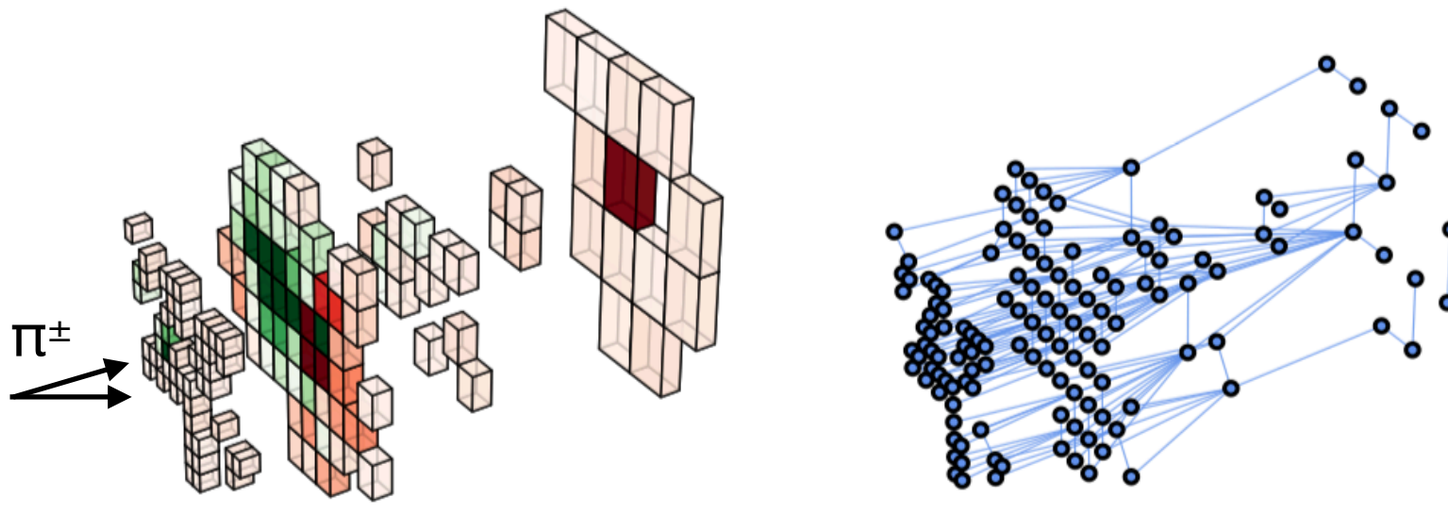
Particle tracking (neighborhood)



event constituents (all-to-all)

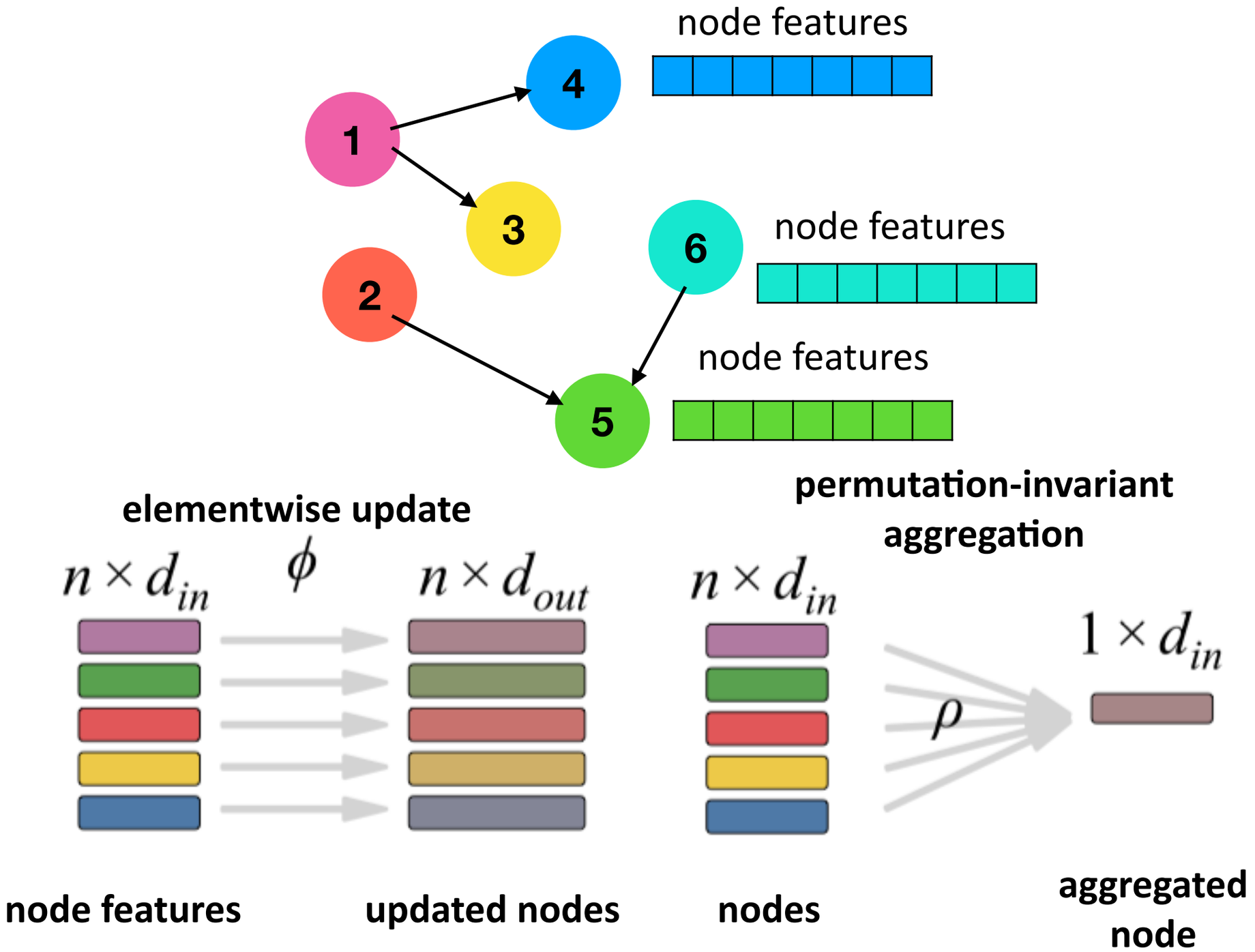


Multilayer calorimeter hits (neighborhood)



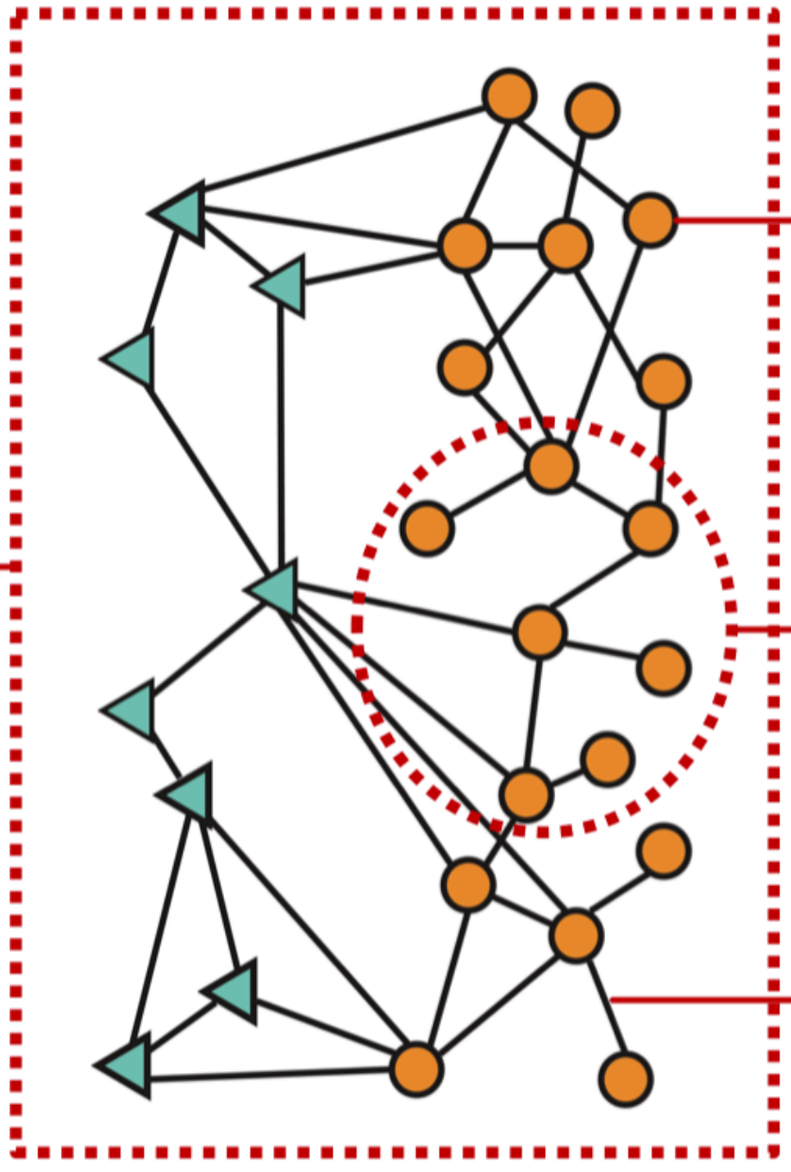
Graph Neural Networks in Particle Physics, Jonathan Shlomi, Peter Battaglia, Jean-Roch Vlimant, 2007.13681, 10.1088/2632-2153/abbf9a

Operations on a graph



Graph Neural Networks in Particle Physics, Jonathan Shlomi, Peter Battaglia, Jean-Roch Vlimant, 2007.13681, 10.1088/2632-2153/abbf9a

Graph problems



Node level

Particle Flow reconstruction, detector hit segmentation

Graph-level prediction, Graph generation

Jet tagging, event tagging

Community (subgraph) level

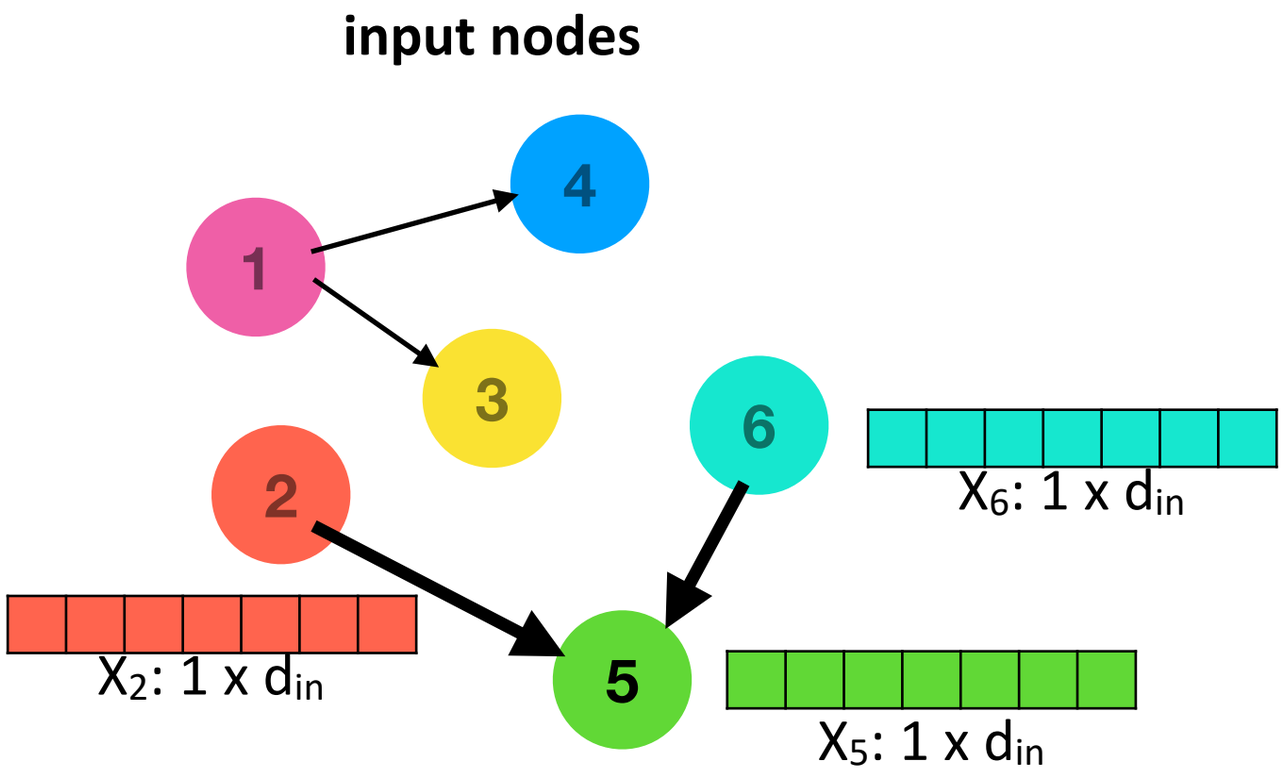
detector hit clustering

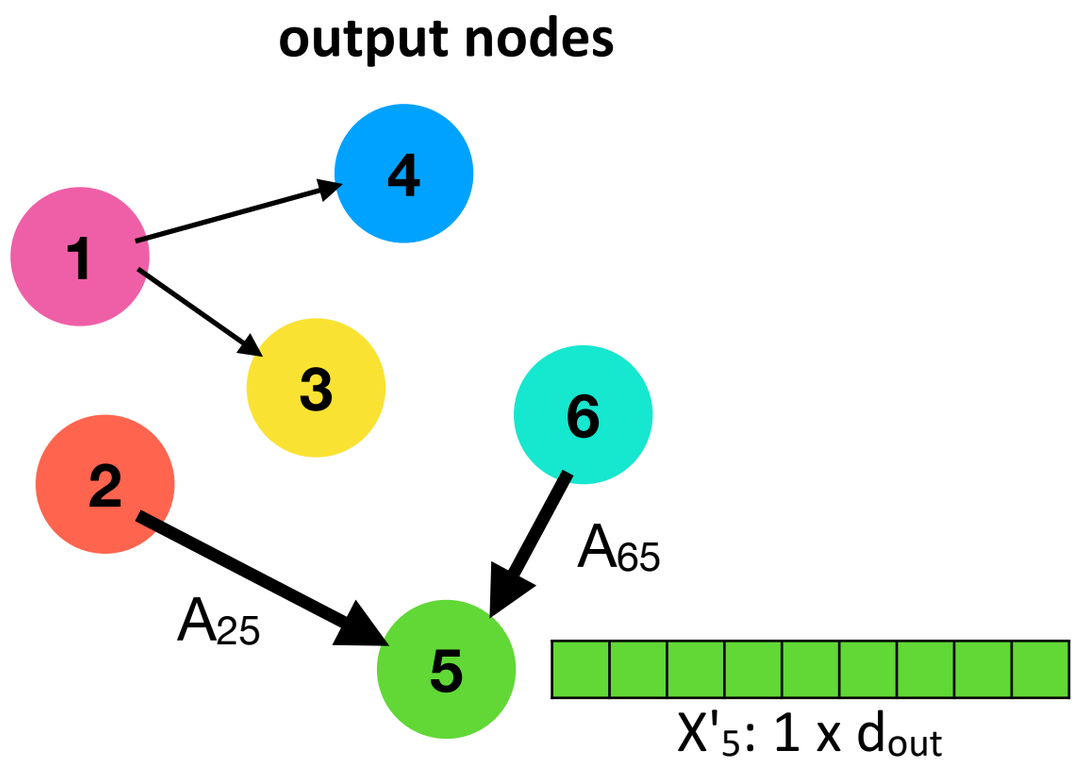
Edge-level

Track reconstruction

J. Leskovec et al [2021]

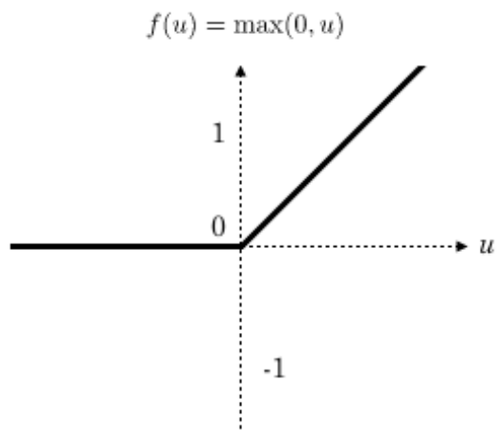
Graph Convolutional Network (GCN)





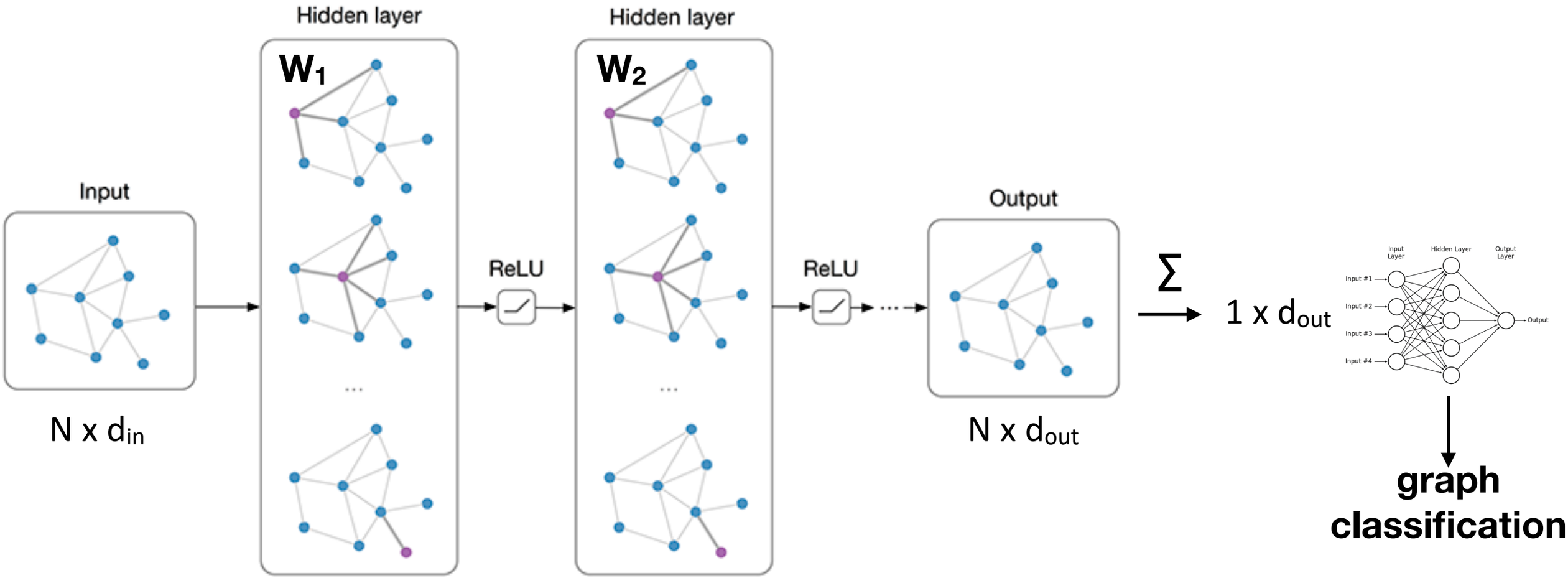
trainable weight matrix **W: d_{in} x d_{out}**

update rule: **$X'_i \rightarrow ReLU[A_{ji} \cdot (X_j \cdot W)]$**



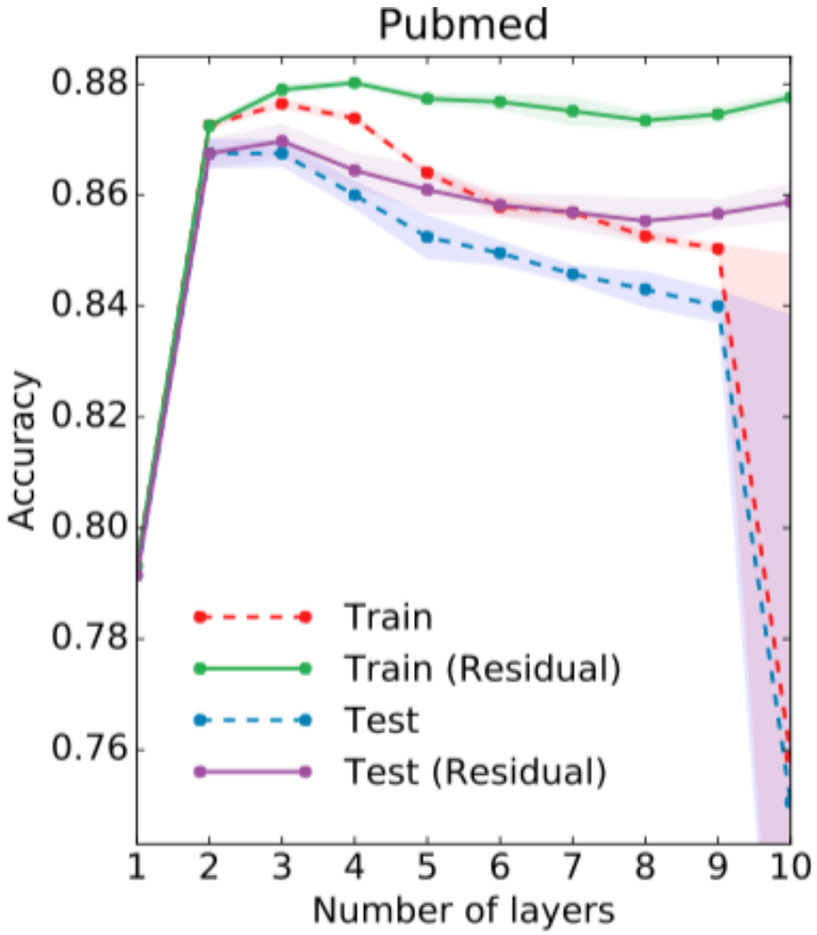
e.g. $X'_5 = ReLU[A_{65} (X_6 \cdot \Theta) + A_{25} (X_2 \cdot \Theta)]$

GCN properties



- A trainable weight matrix W_i (d_{in} x d_{out}) in layer *i* shared across all nodes
- The input and output is a graph. The node features are transformed, the graph structure does not change.
- The GCN is permutation-invariant: it does not matter in which order the set of nodes is formatted as a matrix for computations, due to the permutation-invariant aggregation funcion
- A very nice overview can be found from Kipf & Welling: https://tkipf.github.io/graph-convolutional-networks/

Node smoothing



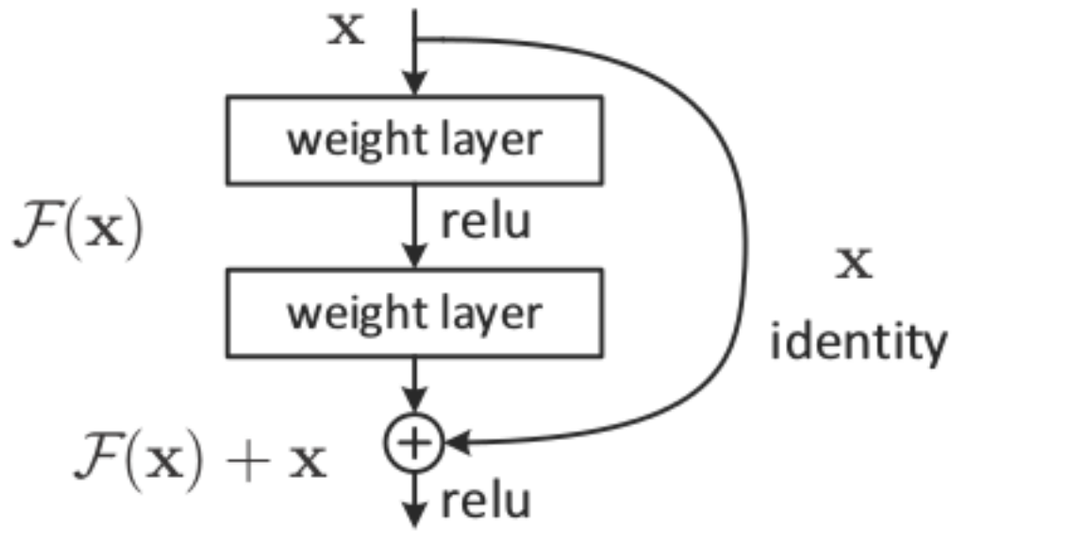


Figure 2. Residual learning: a building block.

Deep GCN without skip connections → oversmoothing, performance drops

Kipf, Thomas N., and Max Welling. "Semi-supervised classification with graph convolutional networks." *arXiv preprint arXiv:1609.02907* (2016).

Message passing

- Different types of graph-related algorithms can be formulated in the **message passing** language
- Nodes pass messages to their neighbors
- Aggregate the messages and update the node state

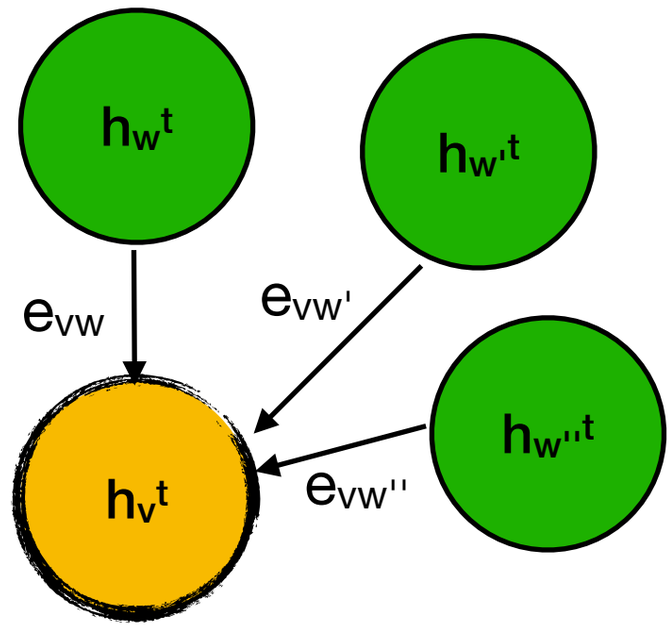
learnable message function

edge features

message $$m_v^{t+1} = \sum_{w \in N(v)} M_t(h_v^t, h_w^t, e_{vw})$$

node features $$h_v^{t+1} = U_t(h_v^t, m_v^{t+1})$$

learnable update function



Gilmer, Justin, et al. "Neural message passing for quantum chemistry." *International Conference on Machine Learning*. PMLR, 2017.

GCN as message passing

Incoming message on a node	$m_v^{t+1} = \sum_{w \in N(v)} M_t(h_v^t, h_w^t)$
Message function	$M_t(h_v^t, h_w^t) \propto A_{vw} h_w^t$
Node update rule	$h_v^{t+1} = U_v^t(h_v^t, m_v^{t+1})$
Node update function	$U_v^t(h_v^t, m_v^{t+1}) = \mathbf{ReLU}(W^t m_v^{t+1})$

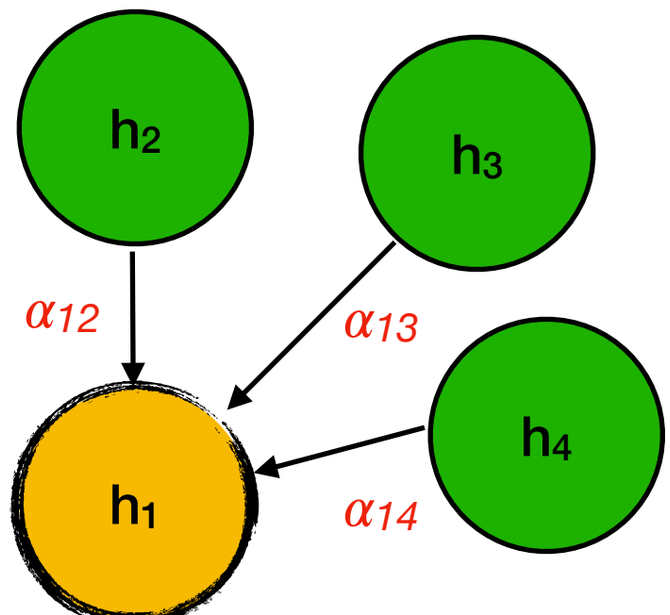
Gilmer, Justin, et al. "Neural message passing for quantum chemistry." *International Conference on Machine Learning*. PMLR, 2017.

Graph Attention (GAT)

Compute an attention coefficient α_{ij} between two pairs of connected nodes.

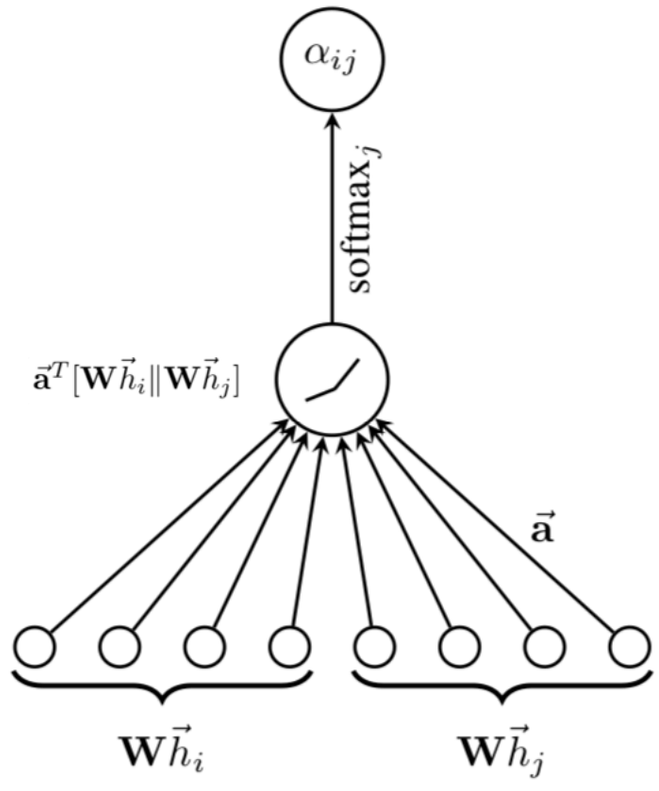
Trainable attention vector **a**, feature weight vector **W.**

$$\alpha_{ij} = \frac{\exp\left(\text{LeakyReLU}\left(\vec{\mathbf{a}}^T[\mathbf{W}\vec{h}_i \| \mathbf{W}\vec{h}_j]\right)\right)}{\sum_{k\in\mathcal{N}_i}\exp\left(\text{LeakyReLU}\left(\vec{\mathbf{a}}^T[\mathbf{W}\vec{h}_i \| \mathbf{W}\vec{h}_k]\right)\right)}$$



Update the node feature vector based on nearby attention coefficients.

$$\vec{h}'_i = \sigma\left(\sum_{j\in\mathcal{N}_i}\alpha_{ij}\mathbf{W}\vec{h}_j\right)$$



Inputs are graphs: N x d_{in}

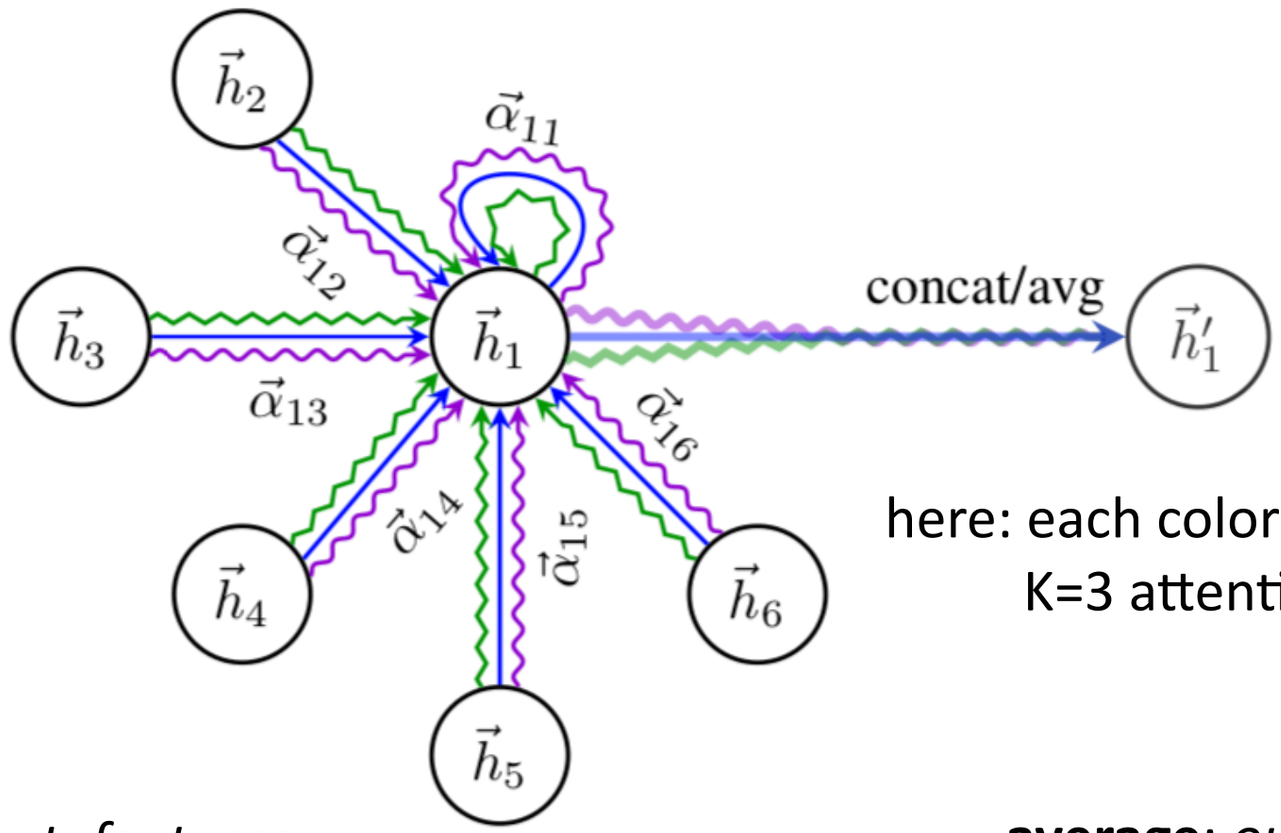
Outputs are graphs: N x d_{out}

Attention vector **a** can be interpreted as feature-to-feature association.

Veličković, Petar, et al. "Graph attention networks." *arXiv preprint arXiv:1710.10903* (2017).

Multi-head GAT

Instead of a single attention coefficient α_{ij} per a node pair, compute K independent values α_{ij}^k.



here: each colored line is one of K=3 attention heads.

concatenate: $K \times$ *out_features*

$$\vec{h}'_i = \Big\Vert_{k=1}^{K} \sigma\left(\sum_{j \in \mathcal{N}_i} \alpha_{ij}^k \mathbf{W}^k \vec{h}_j\right)$$

average: *out_features*

$$\vec{h}'_i = \sigma\left(\frac{1}{K}\sum_{k=1}^{K}\sum_{j \in \mathcal{N}_i} \alpha_{ij}^k \mathbf{W}^k \vec{h}_j\right)$$

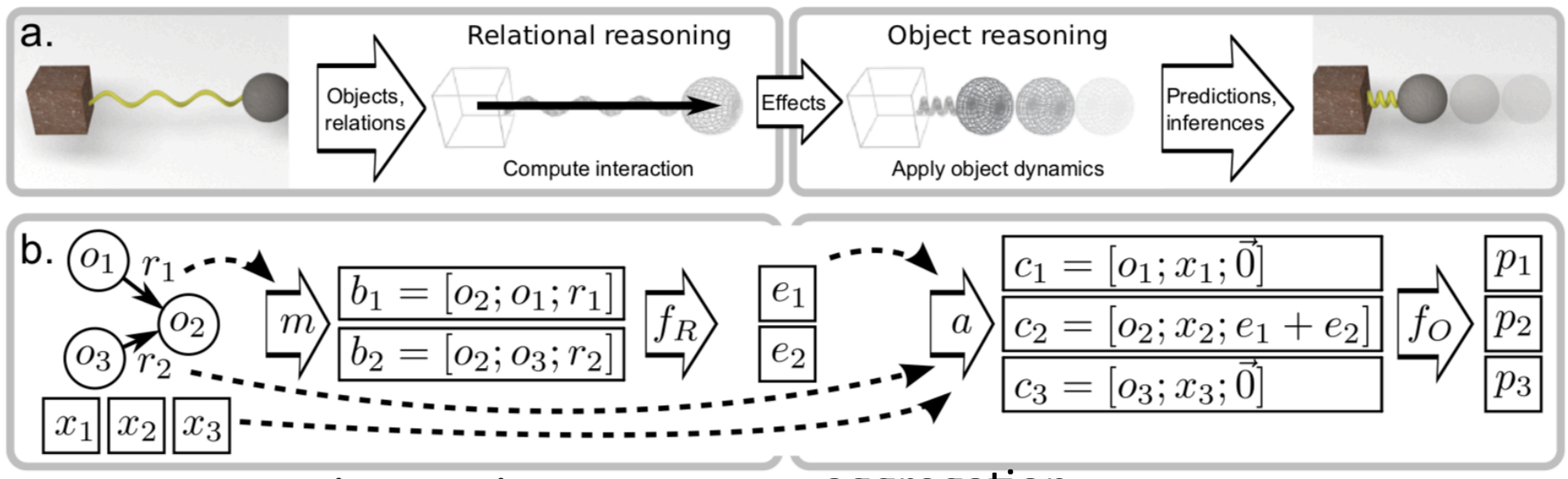
Veličković, Petar, et al. "Graph attention networks." *arXiv preprint arXiv:1710.10903* (2017).

Interaction network (IN)

In the Interaction Network (2016), the message function M_t and the node update function U_t are given as generic neural networks operating on concatenated node and edge inputs.

objects o_i,
relations r_i

externals x_i



updated objects p_i

interaction terms

effects

aggregation

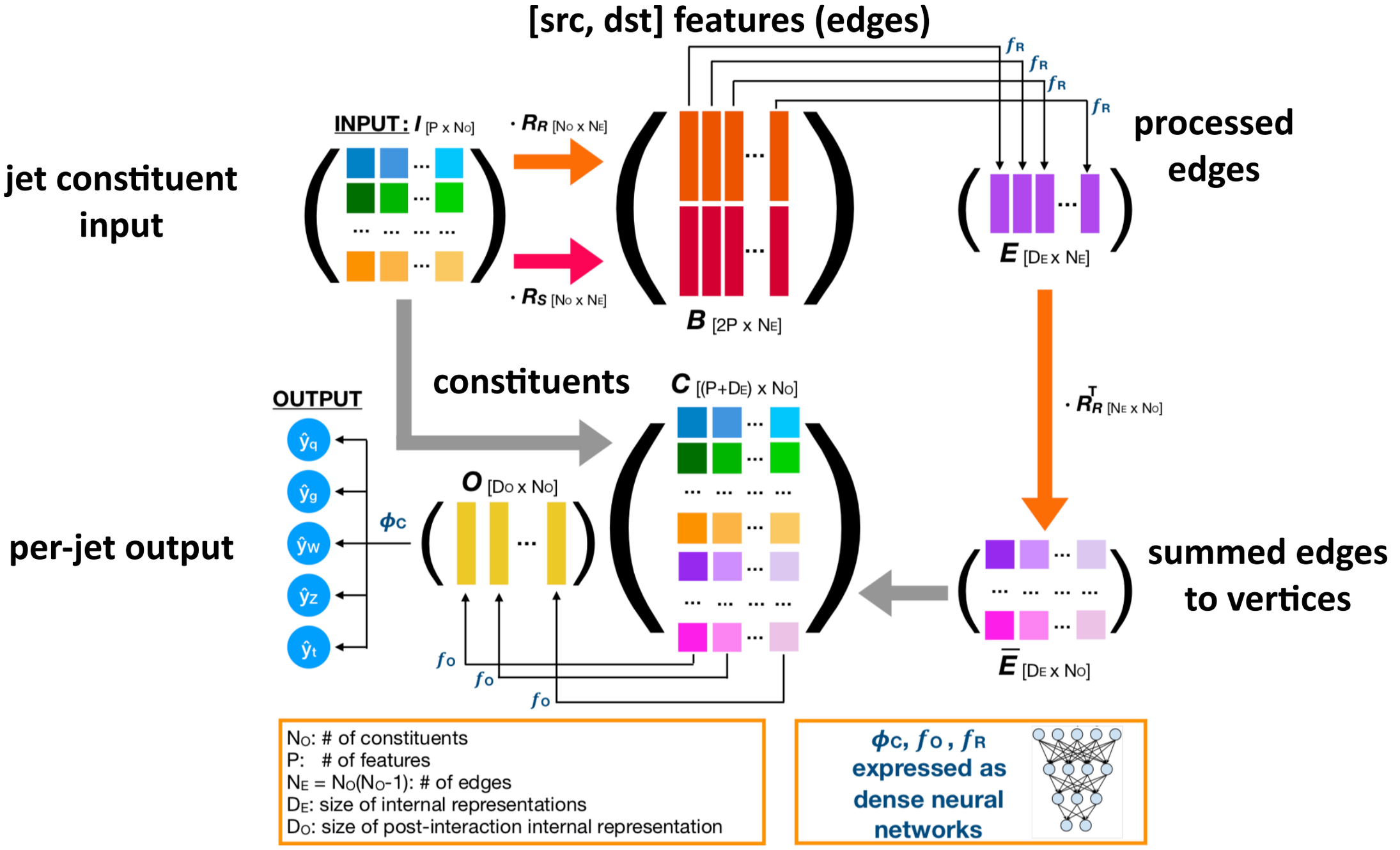
object model f_O

$$f_R(b_1) = \mathbf{MLP}([o_2; o_1; r_1]) = e_1$$

$$f_O(c_2) = \mathbf{MLP}([o_2; x_2; e_1 + e_2]) = p_2$$

Battaglia, Peter W., et al. "Interaction networks for learning about objects, relations and physics." *arXiv preprint arXiv:1612.00222* (2016).

IN for jet tagging



Moreno, E.A., Cerri, O., Duarte, J.M. *et al.* JEDI-net: a jet identification algorithm based on interaction networks. *Eur. Phys. J. C* **80,** 58 (2020). https://doi.org/10.1140/epjc/s10052-020-7608-4

IN for particle tracking

Fully connected: 1000 nodes -> 500k edges, not feasible!

Set up an initial sparse hit graph based on node proximity.

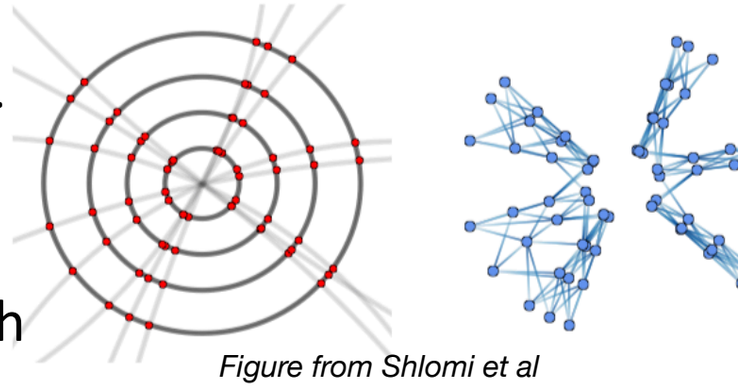
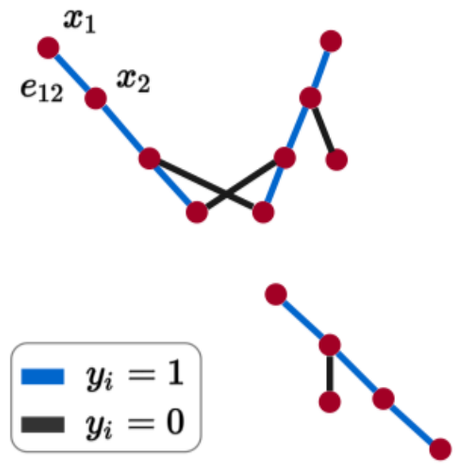


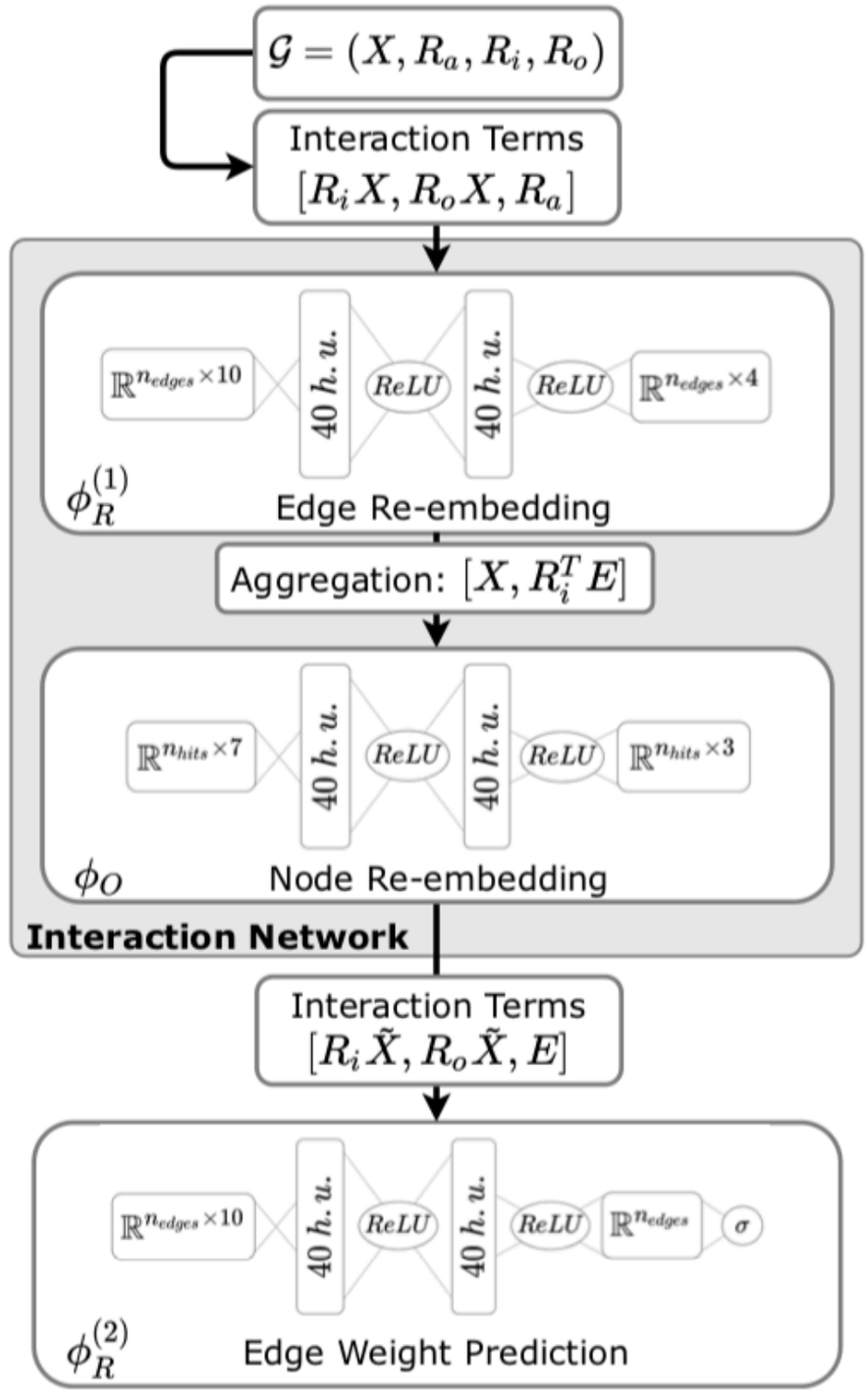
Figure from Shlomi et al

Classify possible edges as true/false based on actual track information, predict edge weight

X: node features (nodes × 3)
R_a: edge features (edges × 4)
R_i, R_o: incoming/outgoing edge matrix
$R_{i,o}X$: incoming/outgoing nodes (edges x 3)



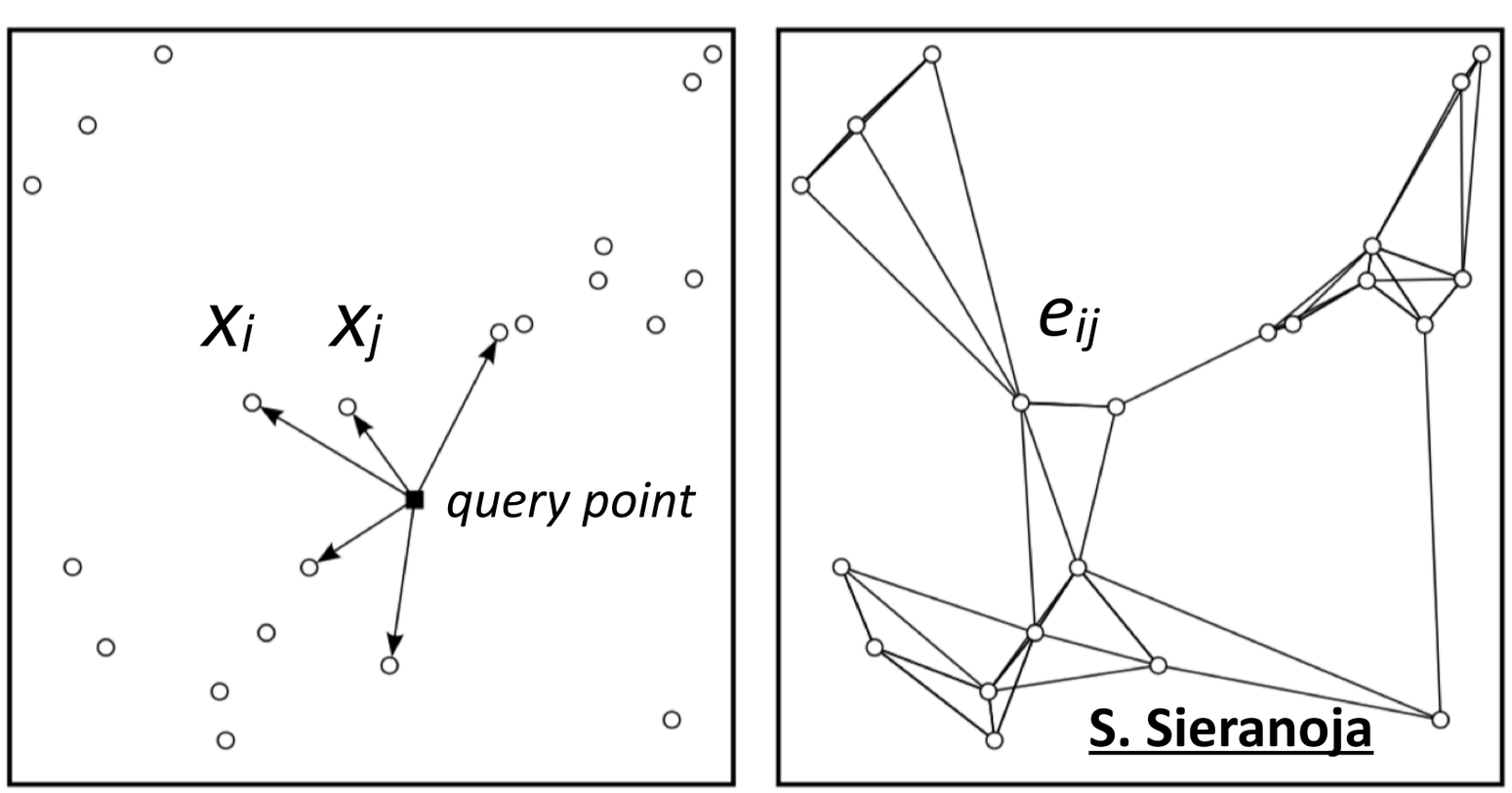
Hitgraph View
DeZoort et al



DeZoort, Gage, et al. "Charged particle tracking via edge-classifying interaction networks." *arXiv preprint arXiv:2103.16701* (2021).

Dynamic graph with kNN

- In the previous examples with GCN, GAT and IN, the graph was static and defined/known in advance
- The structure may not be known in advance, or may be inaccurate
- Construct dynamically: point cloud $\{x_i\}$ → for each point x_i, find k closest neighbors $\{x_j\}$, edges $\{e_{ij}\}$

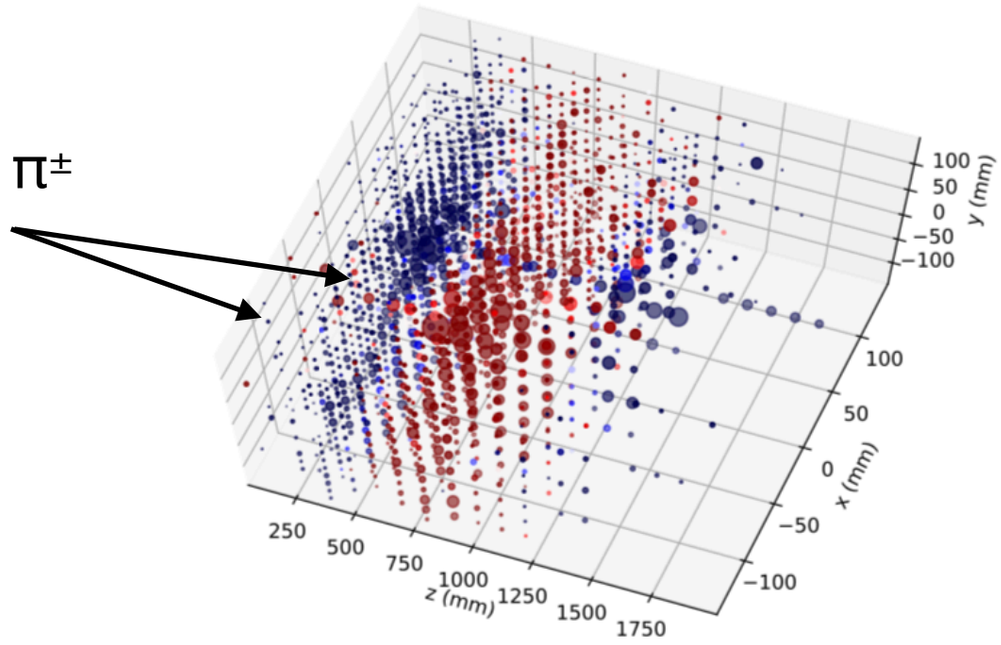


Detector reconstruction

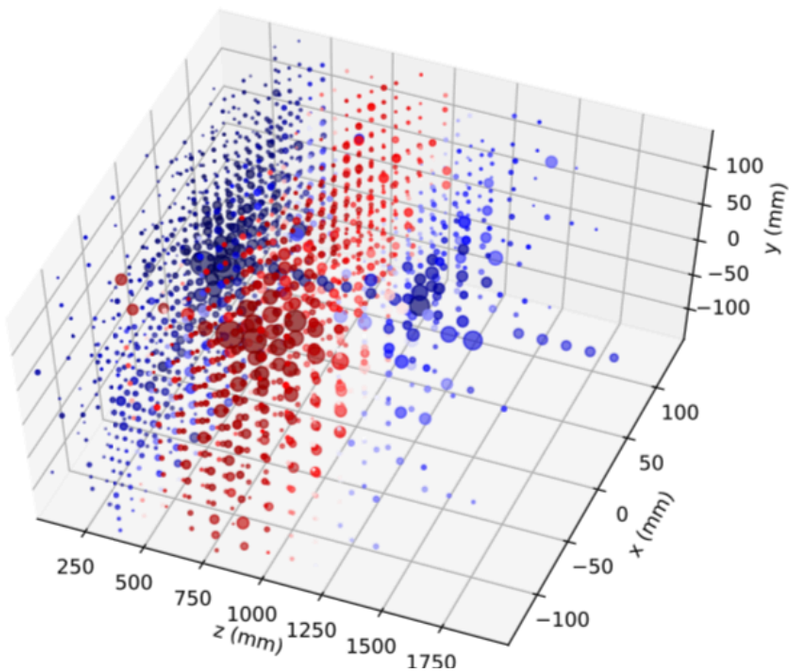
- kNN + sparse graph adjacency matrix: GravNet

- Cluster energy deposits from overlapping showers in a highly granular, layered tungsten detector simulation

- Predict the energy fraction of each sensor (I) belonging to each shower (K): p_{ik} vs t_{ik}



Two overlapping showers generated



reconstructed

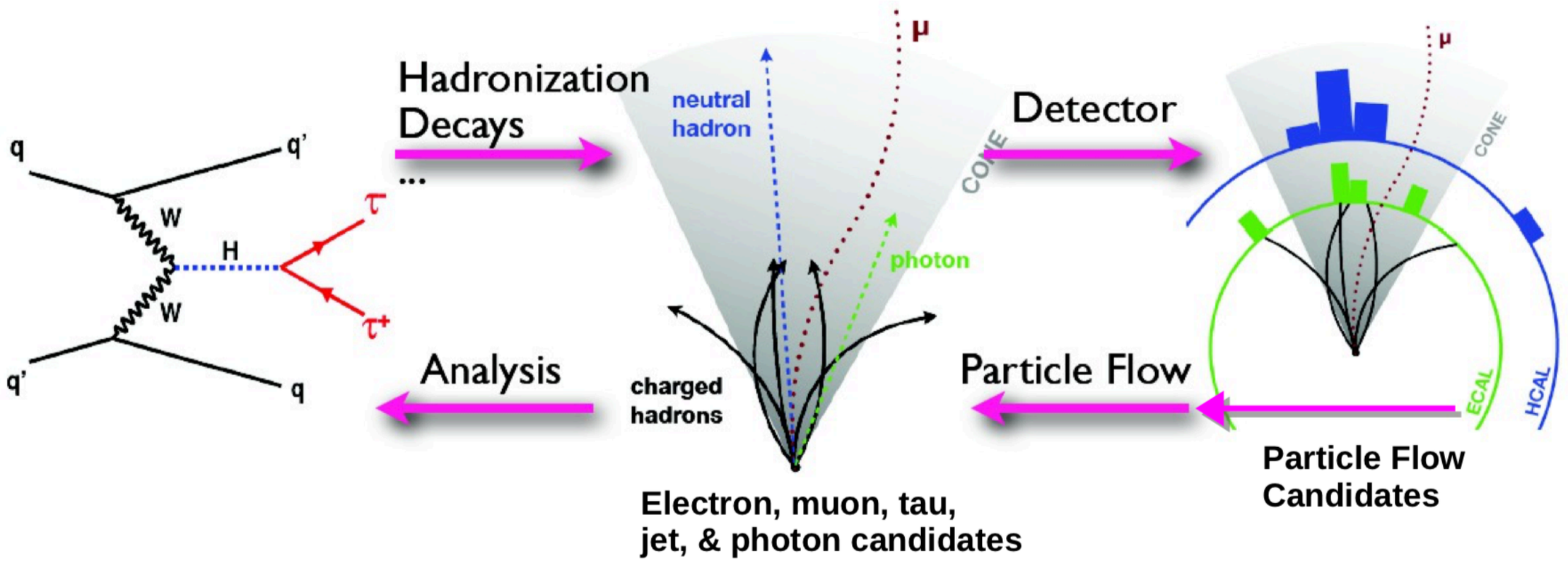
Qasim, Shah Rukh, et al. "Learning representations of irregular particle-detector geometry with distance-weighted graph networks." *The European Physical Journal C* 79.7 (2019): 1-11.

Particle Flow reconstruction

hard interaction

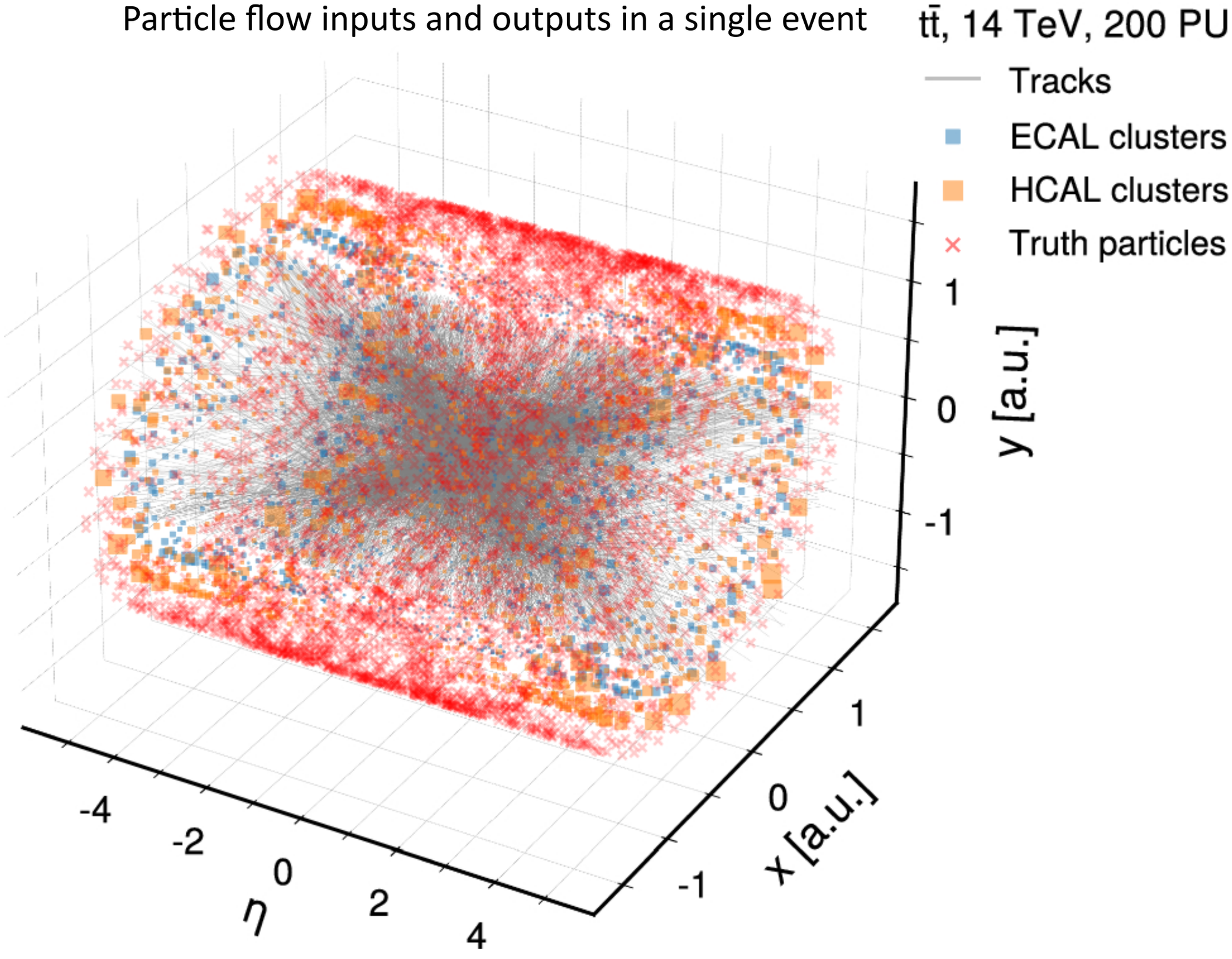
visible final state particles

visible detector hits



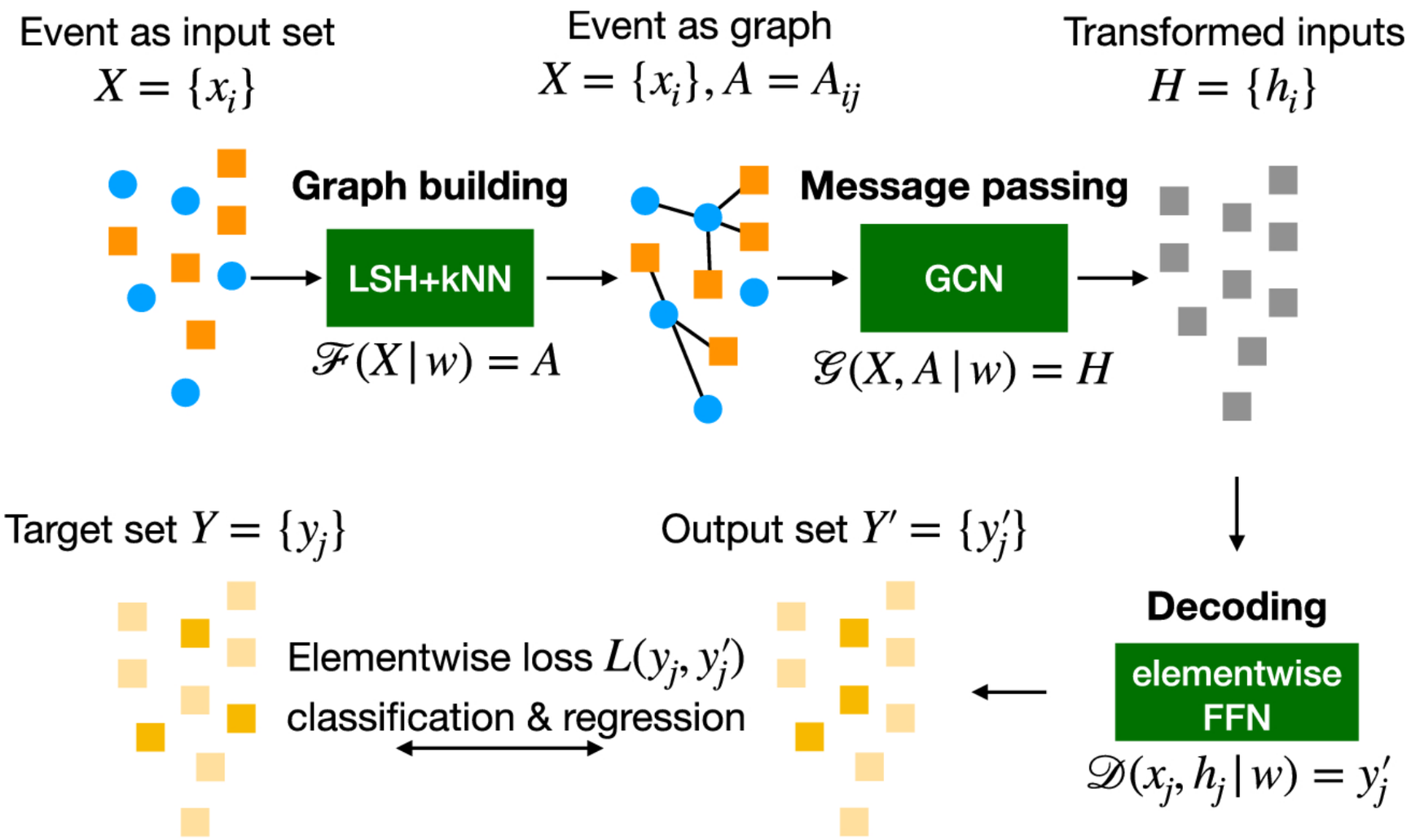
The Particle Flow algorithm combines elements across different detectors to a global particle-level representation of the collision.

Particle flow inputs and outputs in a single event



Pata, J., Duarte, J., Vlimant, JR. *et al.* MLPF: efficient machine-learned particle-flow reconstruction using graph neural networks. *Eur. Phys. J. C* **81,** 381 (2021)

GNNs for Particle Flow



$$x_i = [\text{type}, p_\text{T}, E_\text{ECAL}, E_\text{HCAL}, \eta, \phi, \eta_\text{outer}, \phi_\text{outer}, q, \ldots], \quad \text{type} \in \{\text{track, cluster}\}$$

$$y_j = [\text{PID}, p_\text{T}, E, \eta, \phi, q, \ldots], \quad \text{PID} \in \{\text{none, charged hadron, neutral hadron}, \gamma, e^\pm, \mu^\pm\}$$

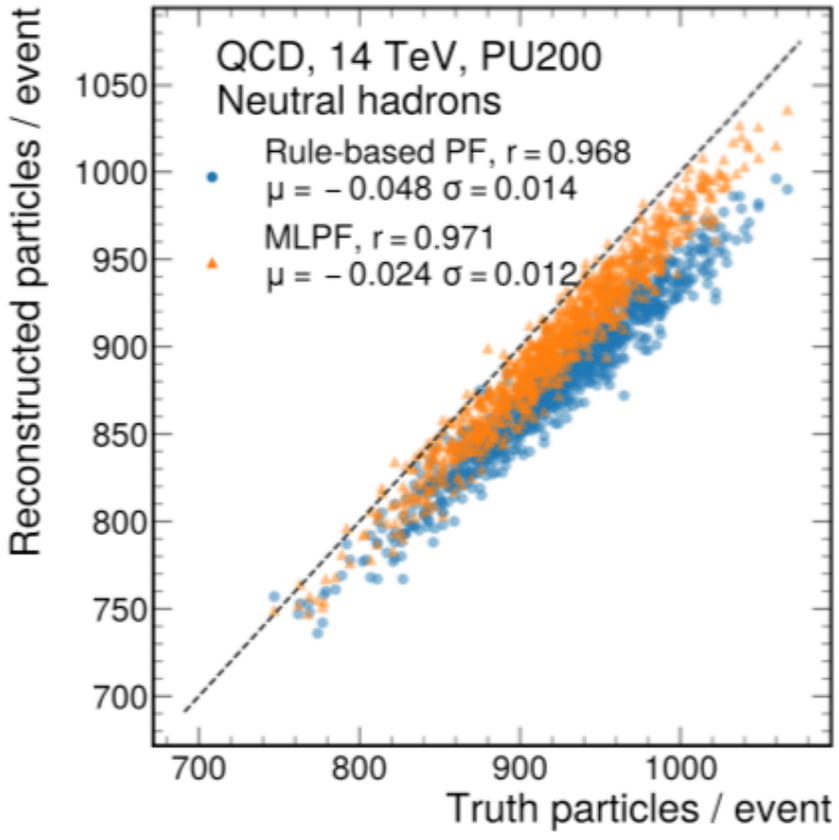
$$h_i \in \mathbb{R}^{256}$$

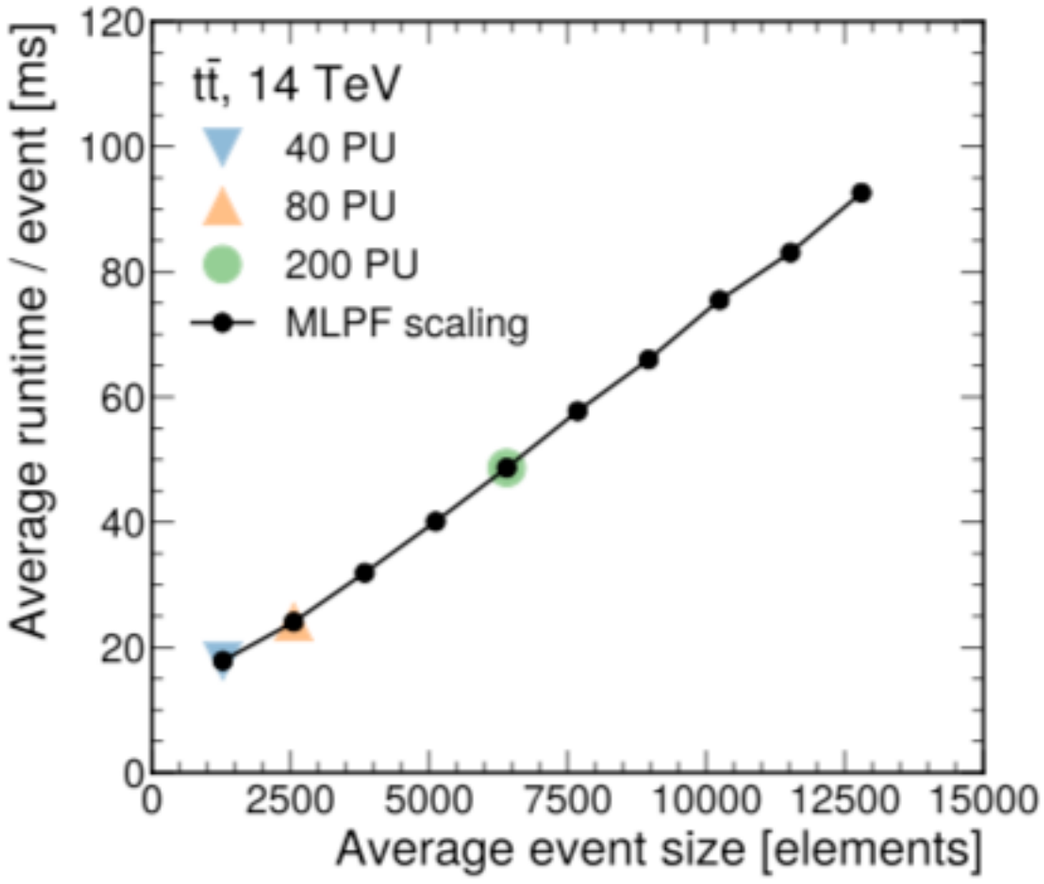
Trainable neural networks: $\mathcal{F}, \mathcal{G}, \mathcal{D}$

● - track, ■ - calorimeter cluster, ■ - encoded element

■ - target (predicted) particle, ■ - no target (predicted) particle

Performance in simulation





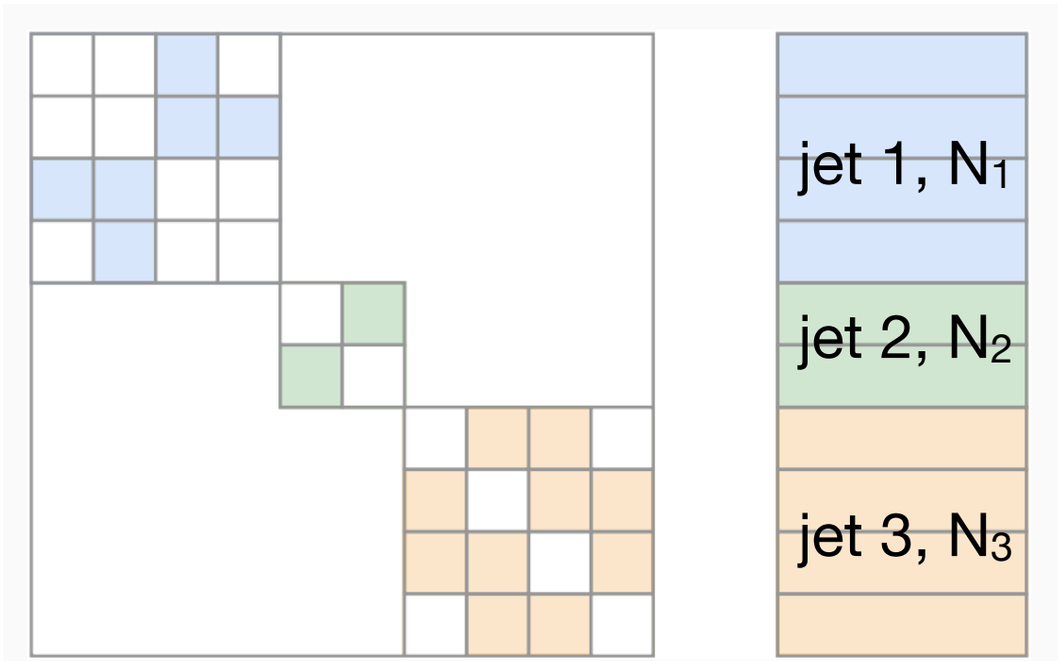
Computational modes

Suppose we have a dataset of jets we want to classify, each jet having N_i constituents.
NN training often requires batching the data to average gradient updates.

Disjoint

features: $(N_1 + N_2 + ...) \times D$

adjacency: $(N_1 + N_2 + ...)^2$

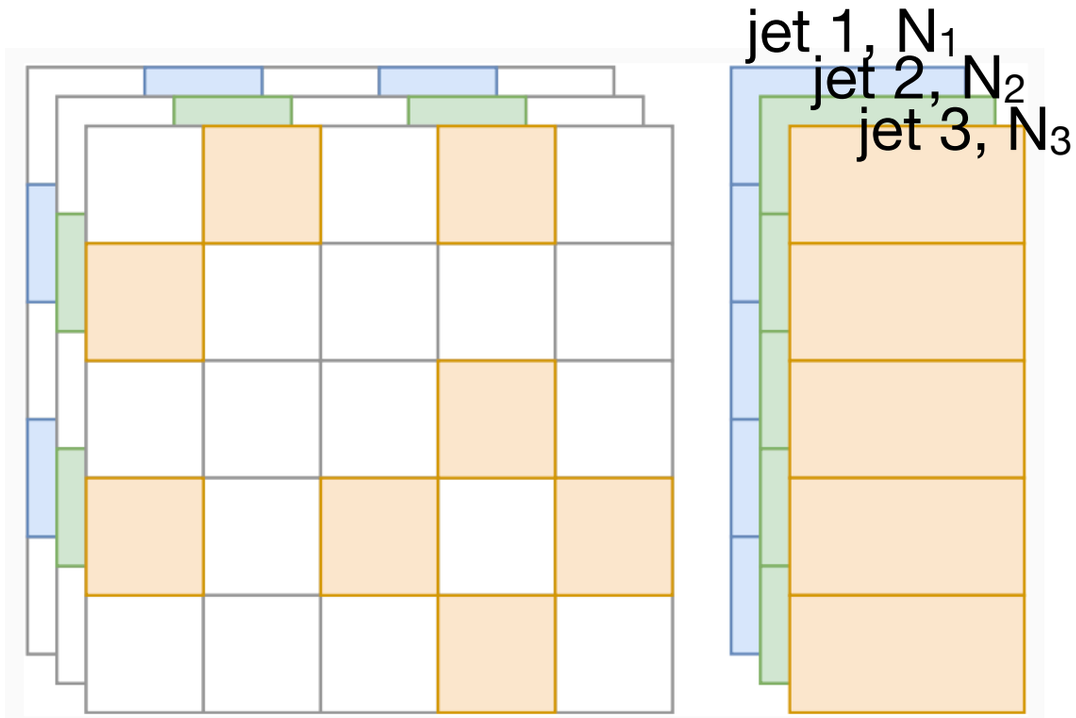


Adjacency is typically sparse.
Suitable for large inputs (N > 1000).
Typically requires on-the-fly computation (e.g. pytorch).

Batched

features: B x N x D

adjacencies: B x N x N

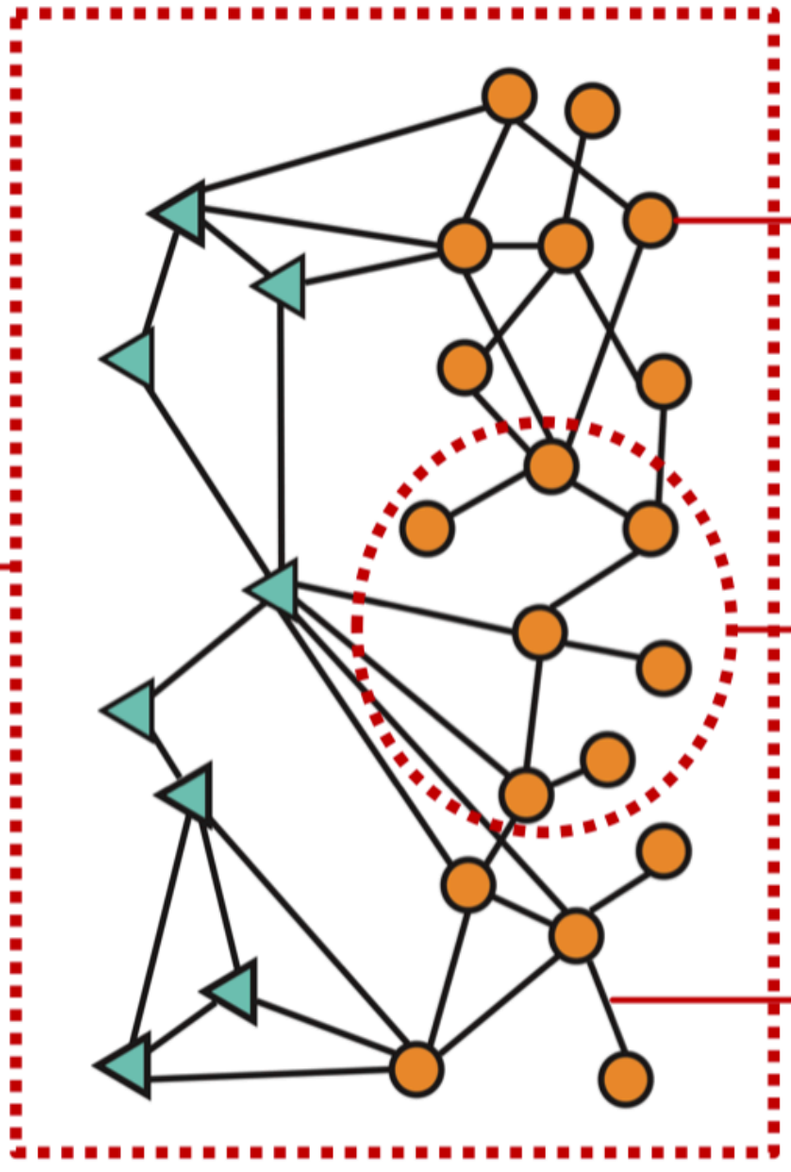


Adjacency is typically dense.
Graphs may be zero-padded / masked to size N.
Suitable for small inputs (<1000) and static computational graphs (e.g. tensorflow).

https://graphneural.network/data-modes/

Recap

Graph problems



Node level

Particle Flow reconstruction, detector hit segmentation

Graph-level prediction, Graph generation

Jet tagging, event tagging

Community (subgraph) level

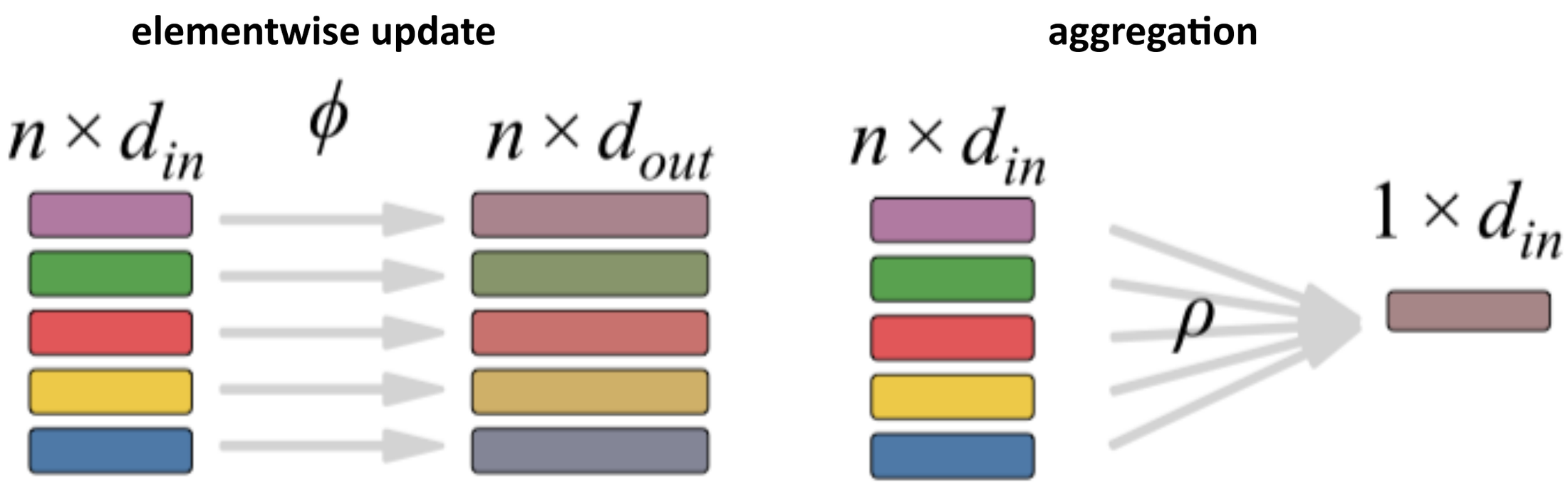
detector hit clustering

Edge-level

Track reconstruction

J. Leskovec et al [2021]

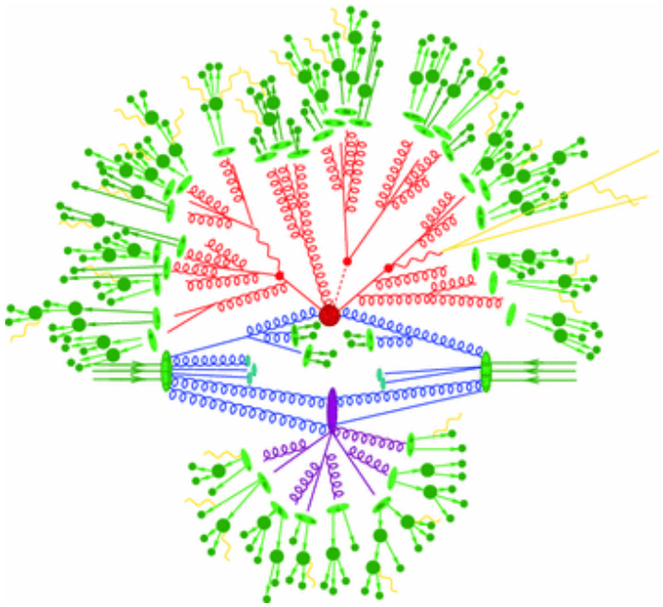
Graph operations



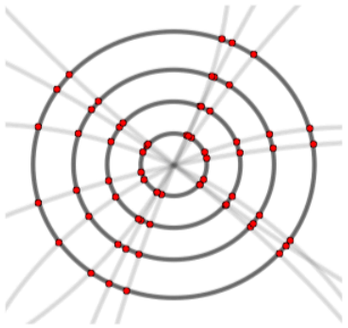
Invariance with respect to permutations!

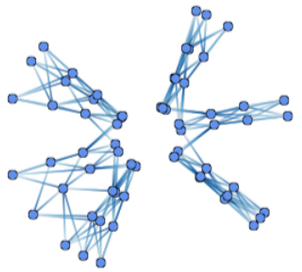
Graph structure

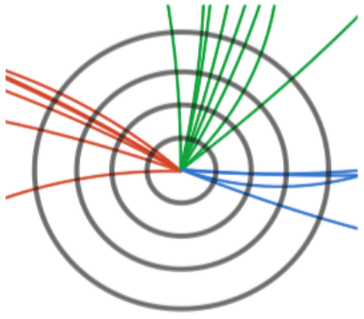
Defined by the process
(but not necessarily observable)

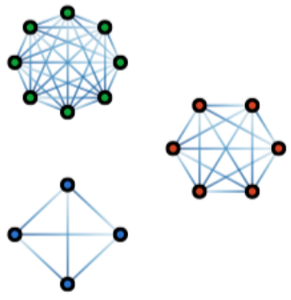


Assumed (static)

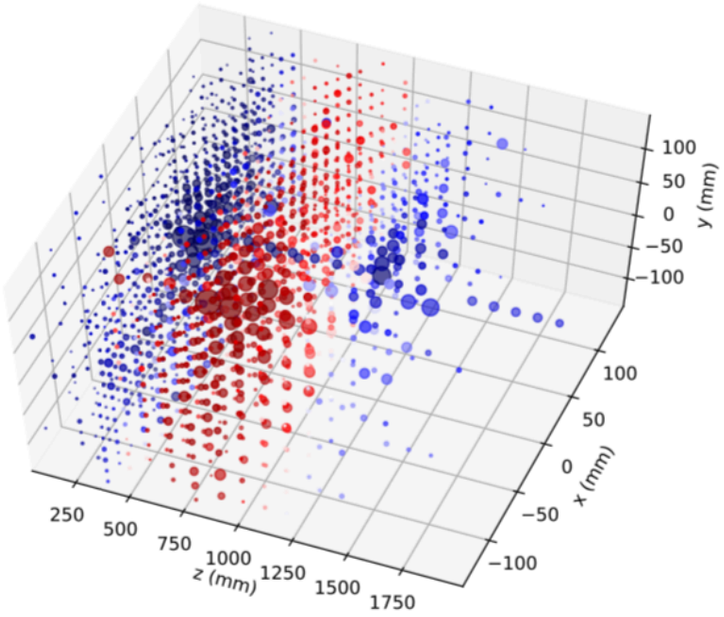


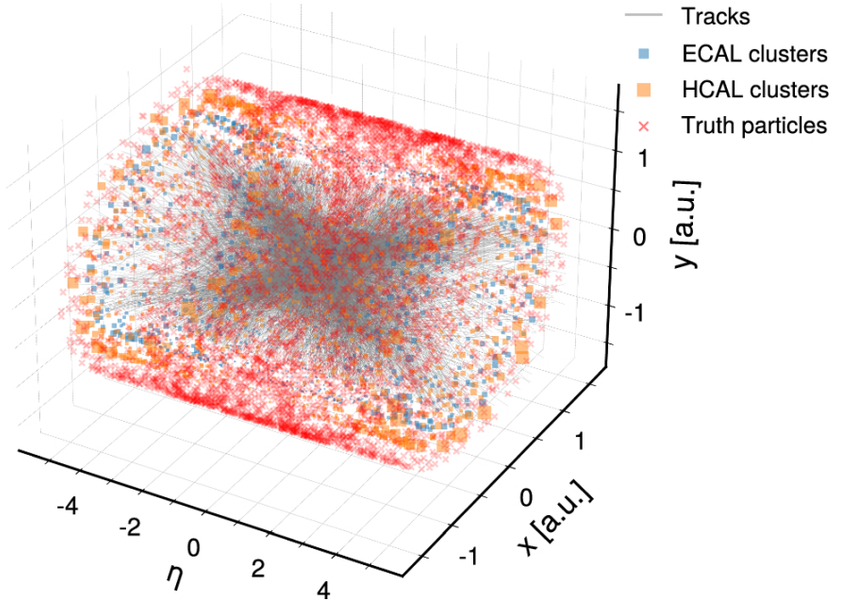


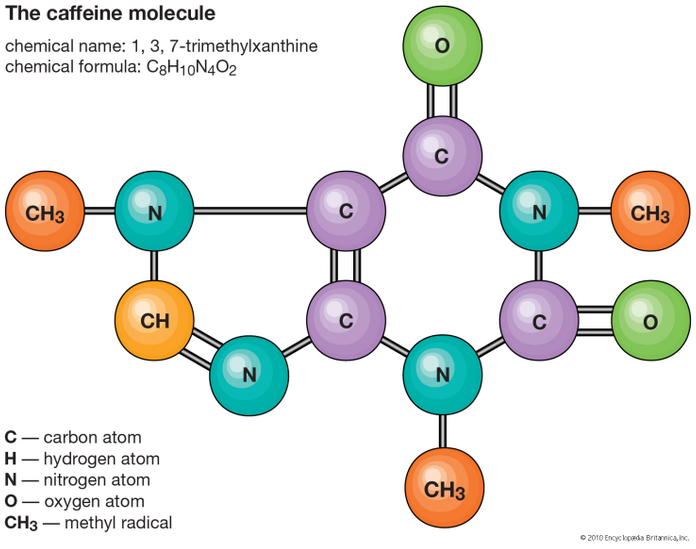




Learned (dynamic)







Advantages of GNNs

- Encode physics priors in the graph structure
- Invariance to permutations / ordering
- Sparse and irregular problem geometries
- Efficient computation and memory representation

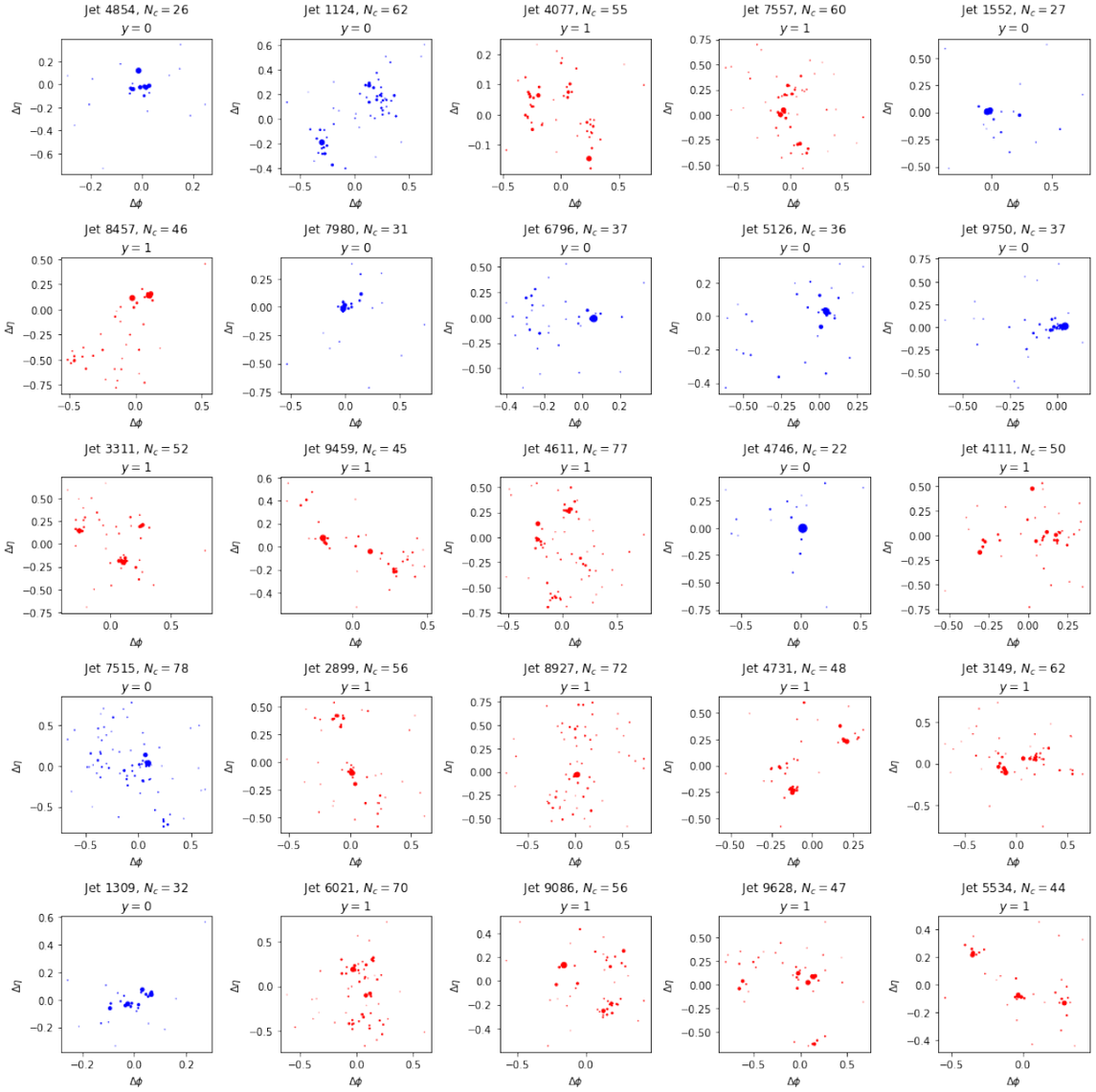
Useful references

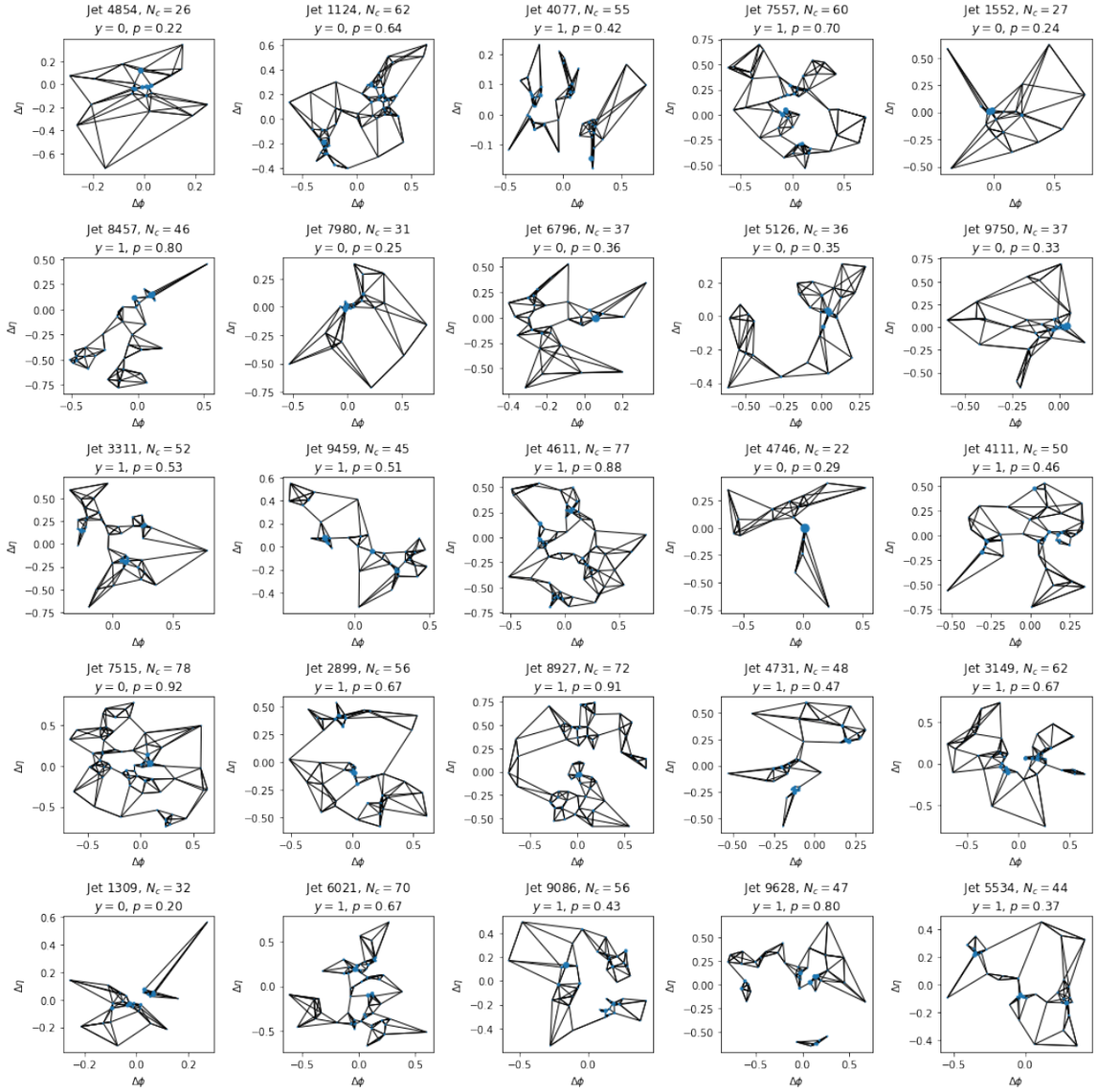
- HEPML Living Review: https://iml-wg.github.io/HEPML-LivingReview/

- ML on Graphs @ Stanford: http://web.stanford.edu/class/cs224w/

- Graph Representation Learning book (WIP): https://www.cs.mcgill.ca/~wlh/grl_book/

Practical exercise





Jupyter notebook: link

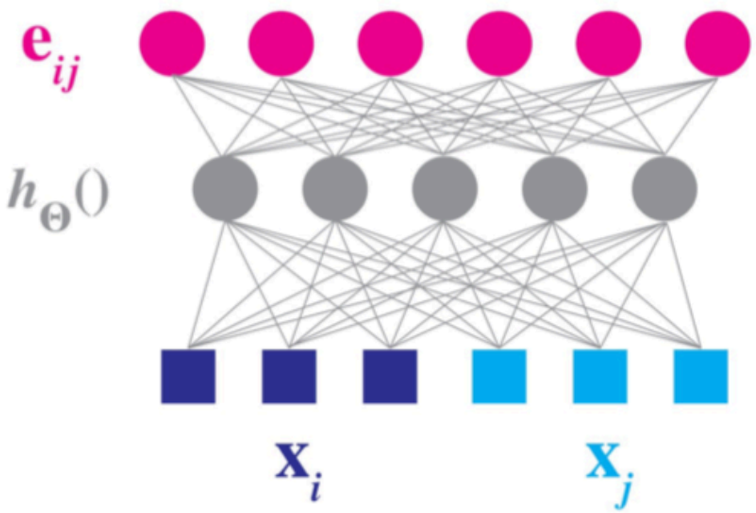
Acknowledgements

- Thanks to Jan Kieseler and Jean-Roch Vlimant for comments & discussion

Backup

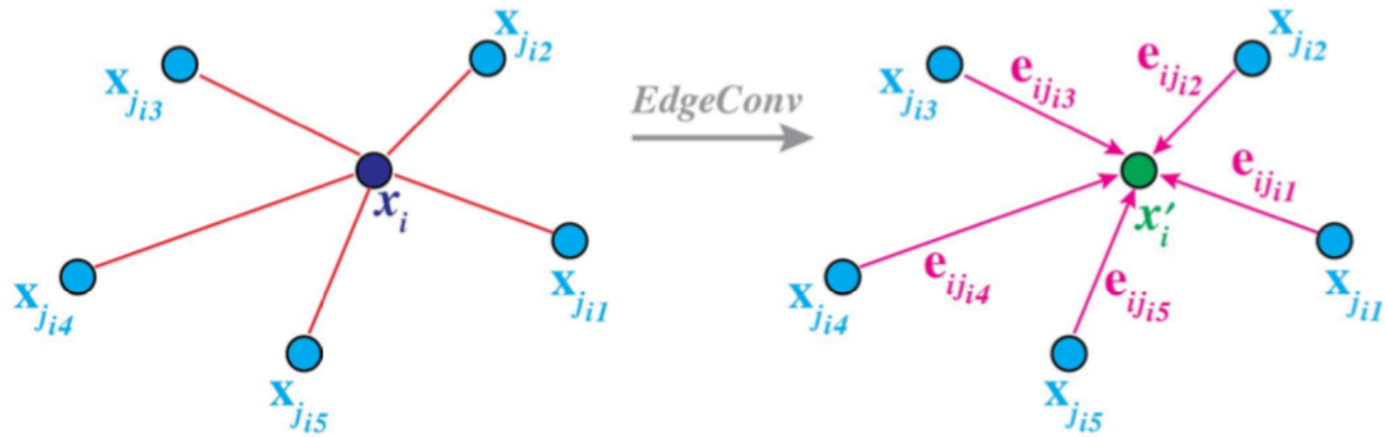
Dynamic graph CNN (DGCNN)

Construct neighbor graph: for each point x_i, find k closest neighbors $\{x_j\}$, edges $\{e_{ij}\}$



construct an edge feature using a learnable function

$$\boldsymbol{h_\Theta}(\boldsymbol{x}_i, \boldsymbol{x}_{i_j}) = \bar{\boldsymbol{h}}_{\boldsymbol{\Theta}}(\boldsymbol{x}_i, \boldsymbol{x}_{i_j} - \boldsymbol{x}_i),$$



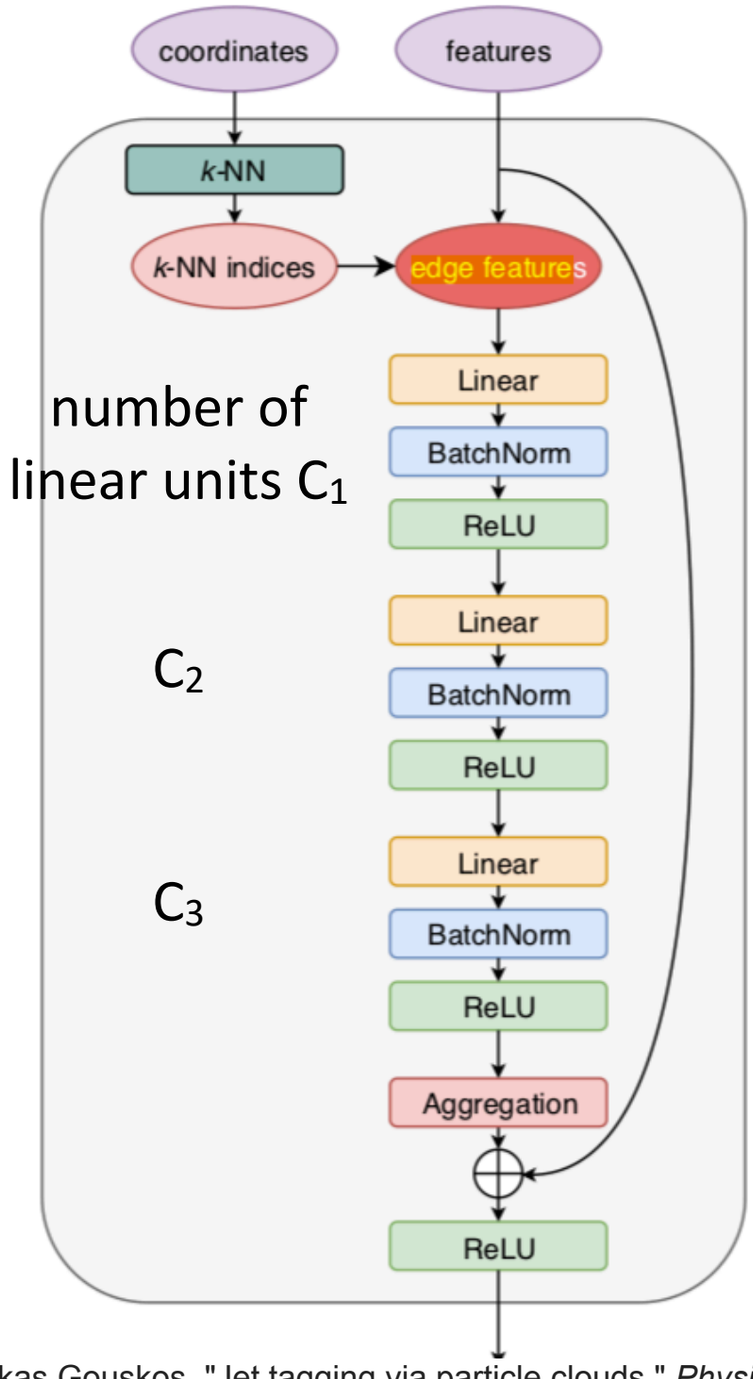
Compute the new point features x_i' using an aggregation over the edges

$$\boldsymbol{x}_i' = \square_{j=1}^{k} \boldsymbol{h_\Theta}(\boldsymbol{x}_i, \boldsymbol{x}_{i_j}),$$

Wang, Yue, et al. "Dynamic graph CNN for learning on point clouds." *Acm Transactions On Graphics (tog)* 38.5 (2019): 1-12.

ParticleNet: DGCNN in HEP

input coordinates (B, N, C)



input features: (B, N, F)

distance matrix: (B, N, N)

edge features: (B, N, K, 2*F)

$$\boldsymbol{h}_{\boldsymbol{\Theta}}(\boldsymbol{x}_i, \boldsymbol{x}_{i_j}) = \bar{\boldsymbol{h}}_{\boldsymbol{\Theta}}(\boldsymbol{x}_i, \boldsymbol{x}_{i_j} - \boldsymbol{x}_i),$$

edge convolutions

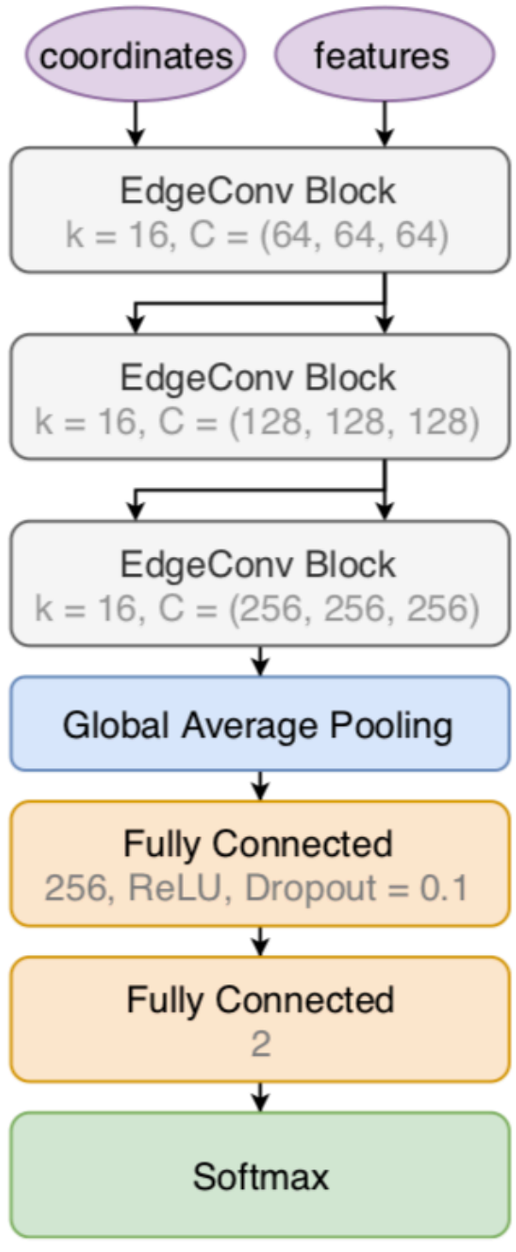
$$\boldsymbol{x}'_i = \square_{j=1}^{k} \boldsymbol{h}_{\boldsymbol{\Theta}}(\boldsymbol{x}_i, \boldsymbol{x}_{i_j}),$$

output features: (B, N, O)

Qu, Huilin, and Loukas Gouskos. "Jet tagging via particle clouds." *Physical Review D* 101.5 (2020): 056019.

ParticleNet full model

- Up to 100 highest-pT constituents of each jet
- relative η, φ coordinates wrt. the jet axis as coordinates
- Features are derived from 4-momentum (log transforms, ratios)
- Coordinates in subsequent layers are derived from previous layer outputs



(a) ParticleNet

Qu, Huilin, and Loukas Gouskos. "Jet tagging via particle clouds." *Physical Review D* 101.5 (2020): 056019.

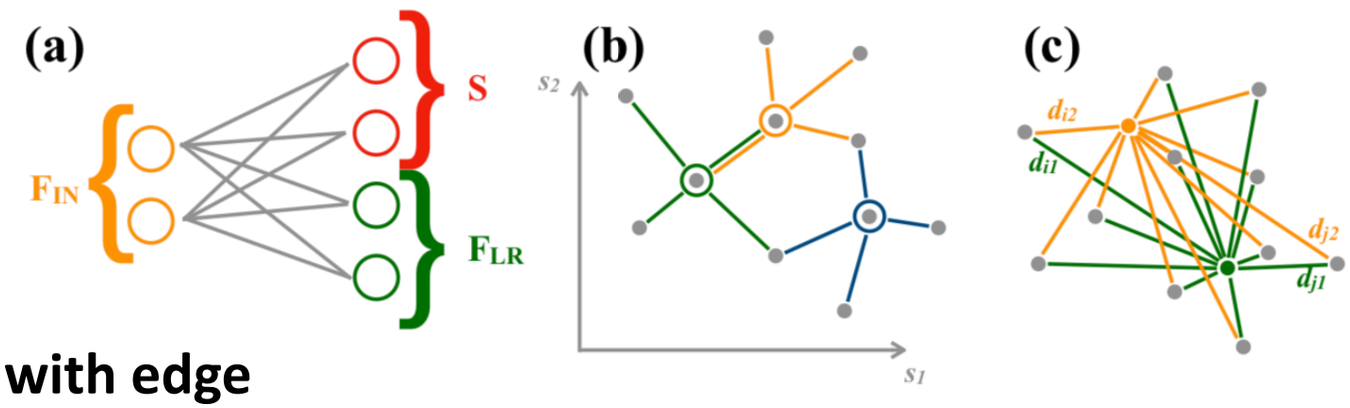
GravNet/GarNet

- Full, dense NxN distance matrix can be too large to store for N>few hundred, kNN can be expensive in a high-dimensional input space

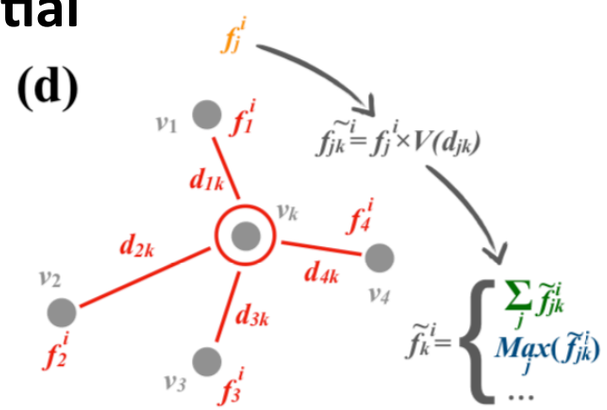
- In case low latency, low memory consumption is desirable, optimize by using a sparse adjacency matrix, separating spatial components and feature components

GravNet: full kNN graph on all nodes

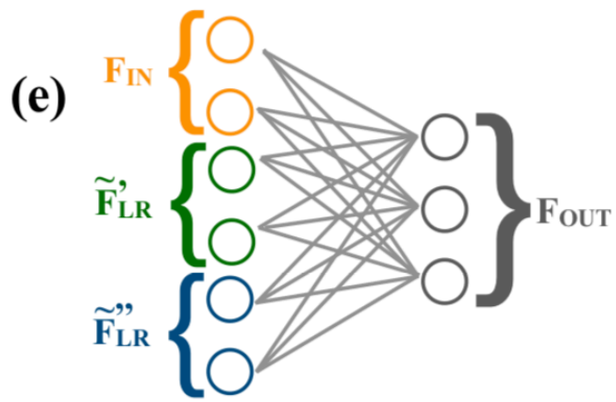
GarNet: choose a fixed number of aggregators



transform nodes with edge info + potential



aggregate

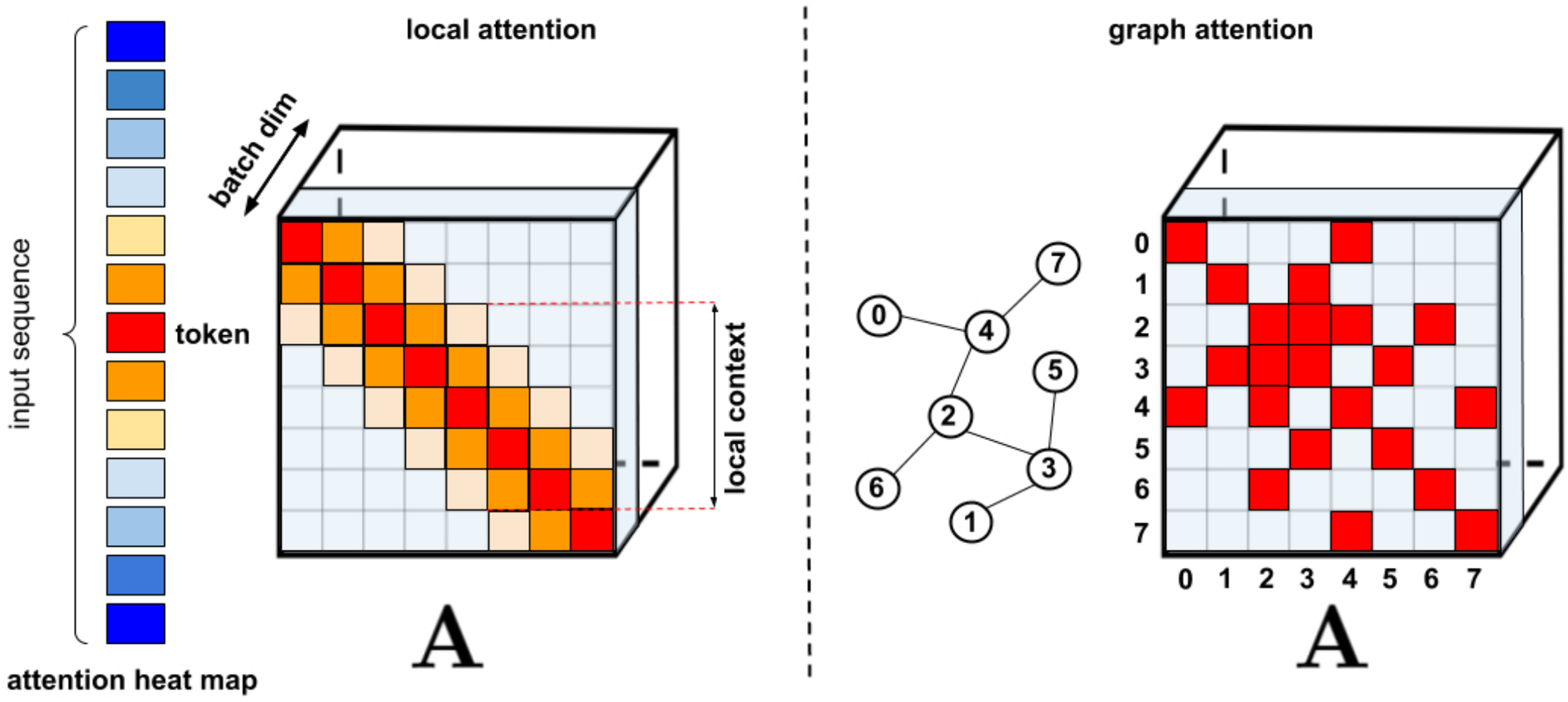


compute node output

Qasim, Shah Rukh, et al. "Learning representations of irregular particle-detector geometry with distance-weighted graph networks." *The European Physical Journal C* 79.7 (2019): 1-11.

GNNs and Transformers

Most state-of-the-art language processing models use an attention-based "transformer" architecture: a dense attention matrix with elements A_{ij} is computed between input elements x_i. The attention matrix **A** is used to successively transform the input elements.

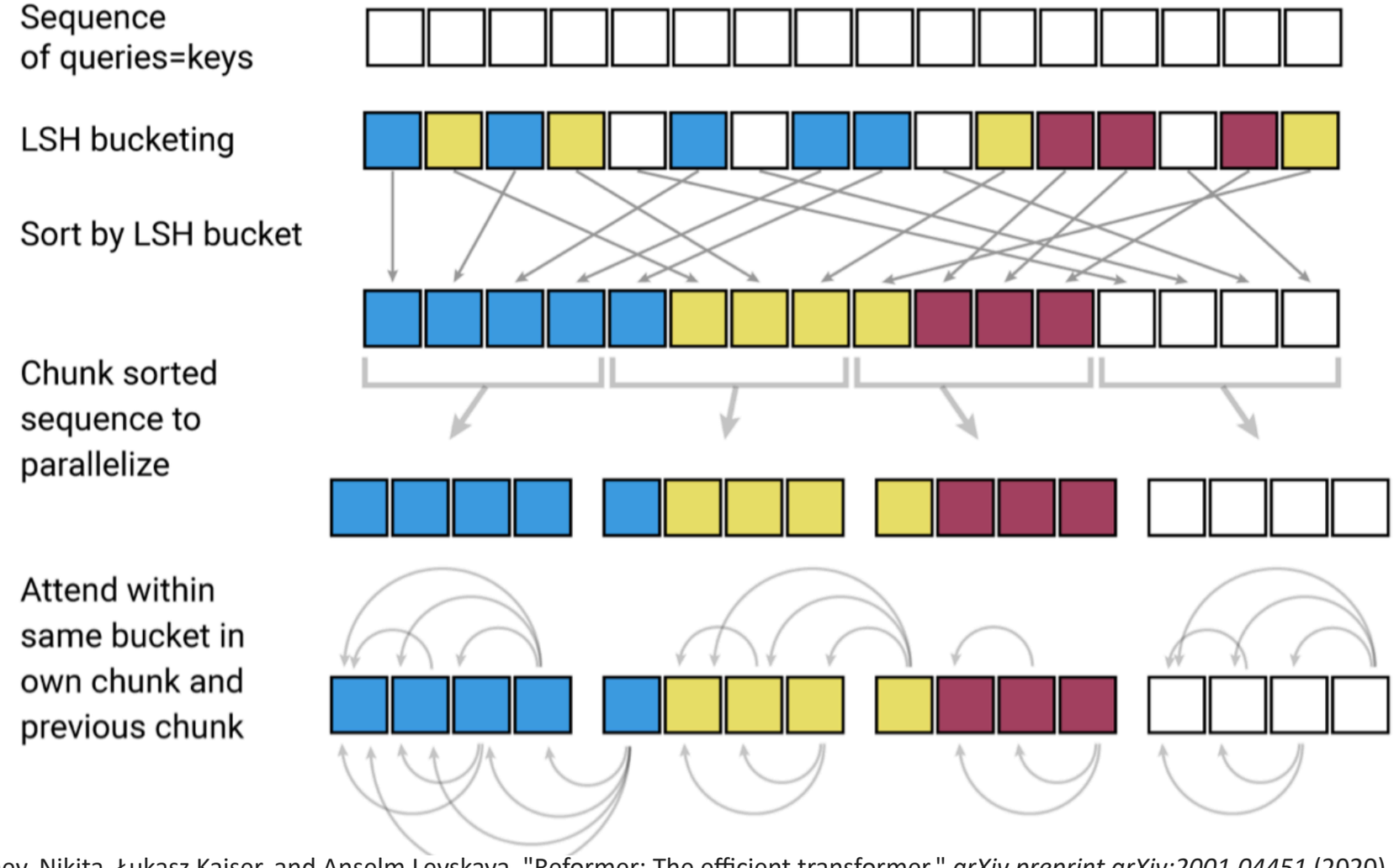


In GNNs, the learned graph adjacency is usually sparse, but is similarly used to propagate information between associated input elements to transform them.

https://ai.googleblog.com/2020/10/rethinking-attention-with-performers.html

Scalable pairwise operations

- Naive kNN graph construction (e.g. *tf.nn.top_k*) scales as $O(N^2)$ with the number of input nodes N
- For N>few hundred, this can be prohibitive (in memory and computation) and thus require splitting up the data
- To process long sequences or full events, there has been recent interest in models that scale better than quadratically, e.g. using an approximate bucketing based on Locality Sensitive Hashing



Kitaev, Nikita, Łukasz Kaiser, and Anselm Levskaya. "Reformer: The efficient transformer." *arXiv preprint arXiv:2001.04451* (2020).