

Graph Neural Networks

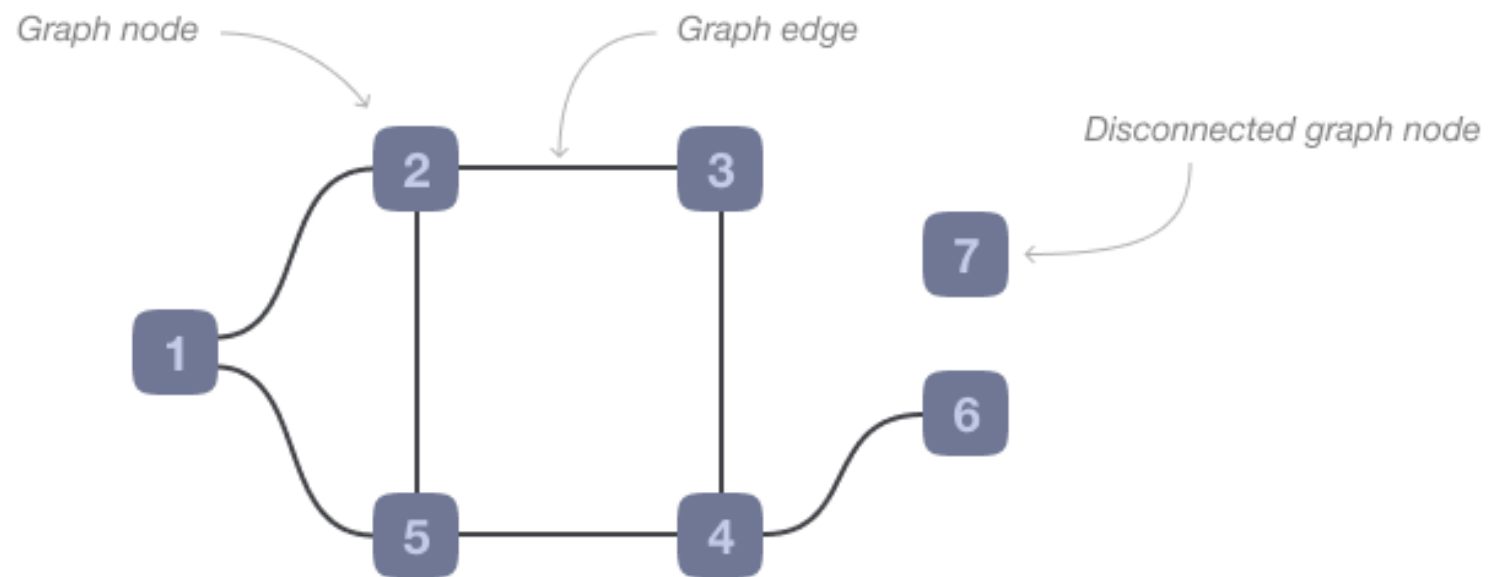
Nikita Kazeev

Plan

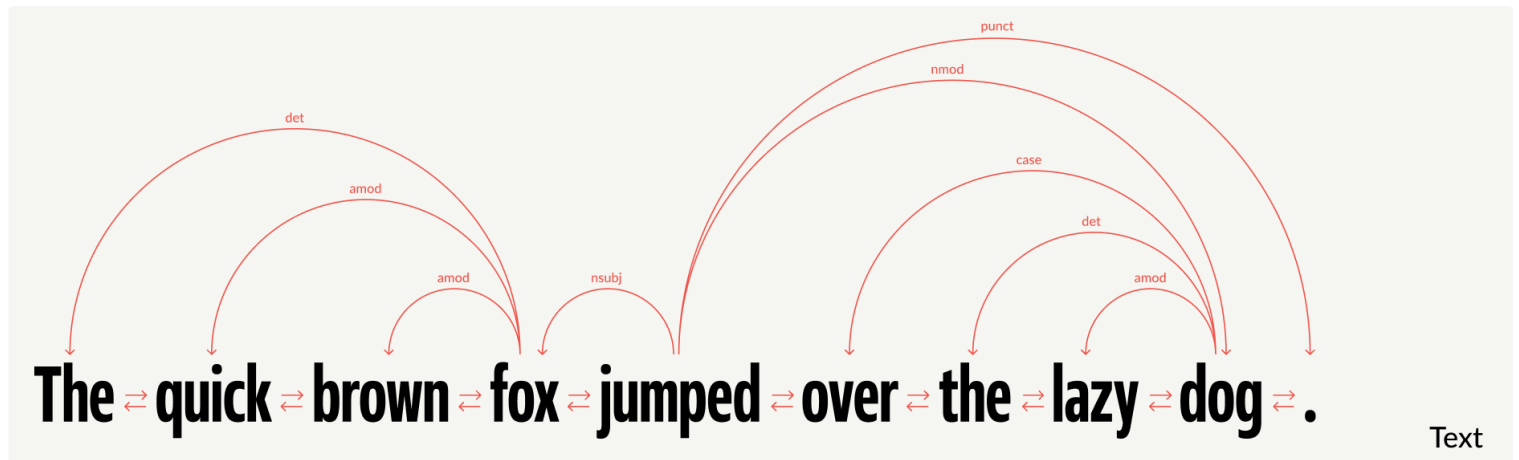
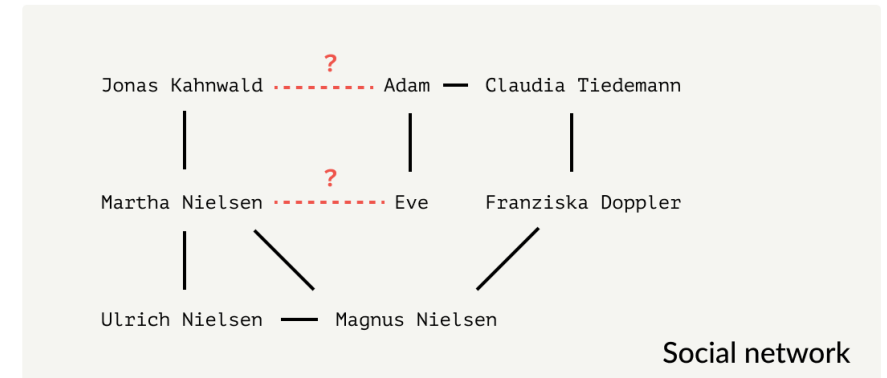
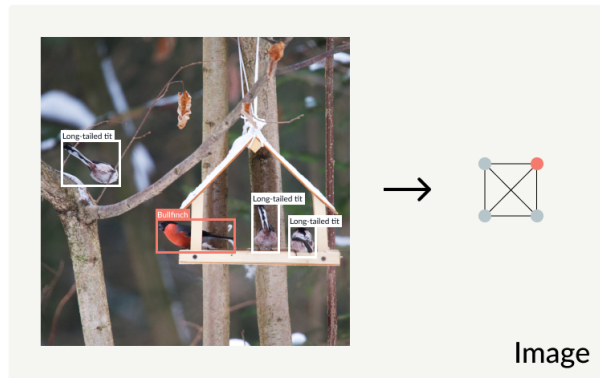
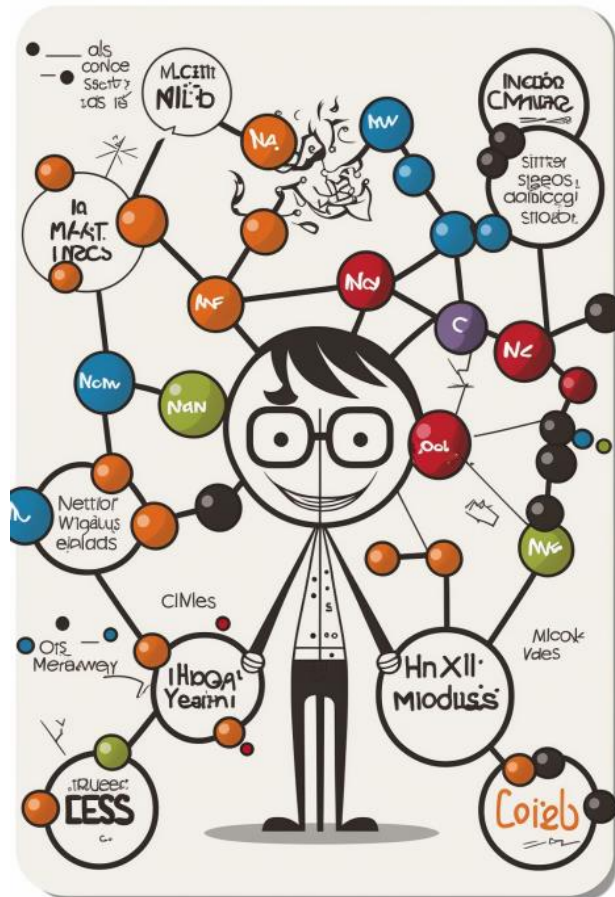
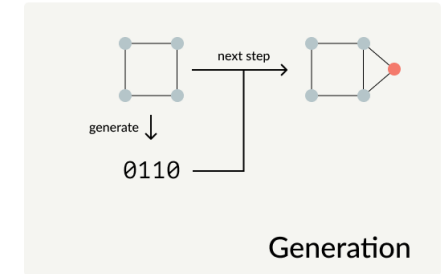
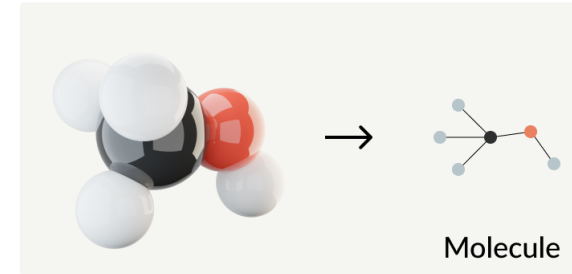
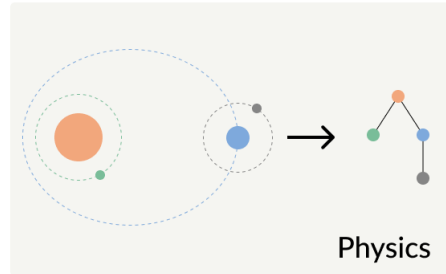
- The conceptual advantages of graph neural networks
- Graph neural networks (GNNs)
- GNNs in high energy physics
- GNNs in low energy physics

What is the point of graphs?

A graph

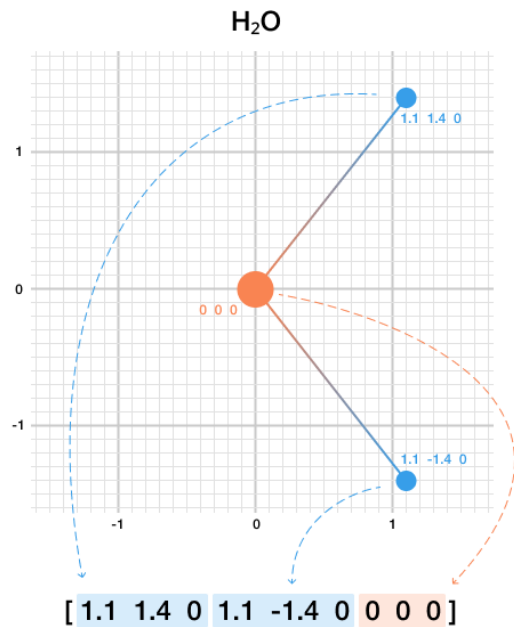


Graphs of the world



Toy (not really) problem: predict potential energy of a molecule

Naïve attempt

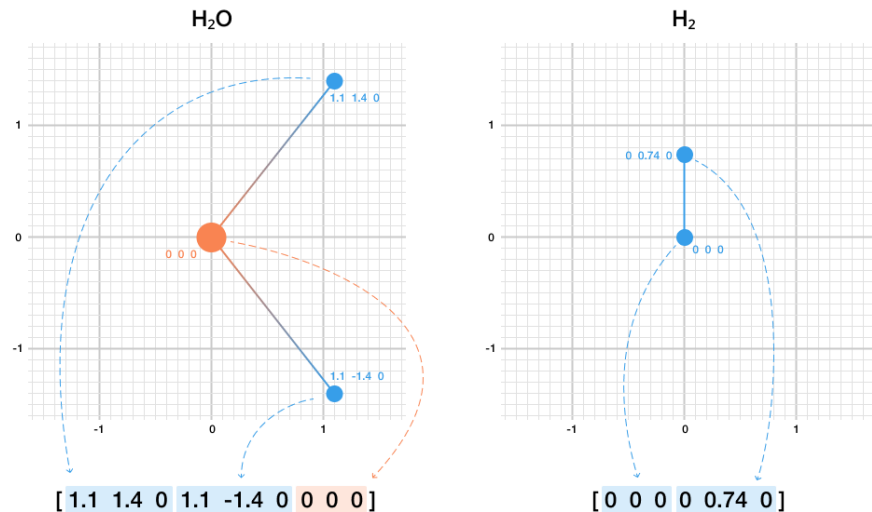


CatBoost

Profit?

No varying system size

No profit



```
[6] np.concatenate([h2o, h2], axis=0)
```

ValueError

Traceback (most recent call last)

<ipython-input-6-6c19194e29cb> in <cell line: 1>()

----> 1 np.concatenate([h2o, h2], axis=0)

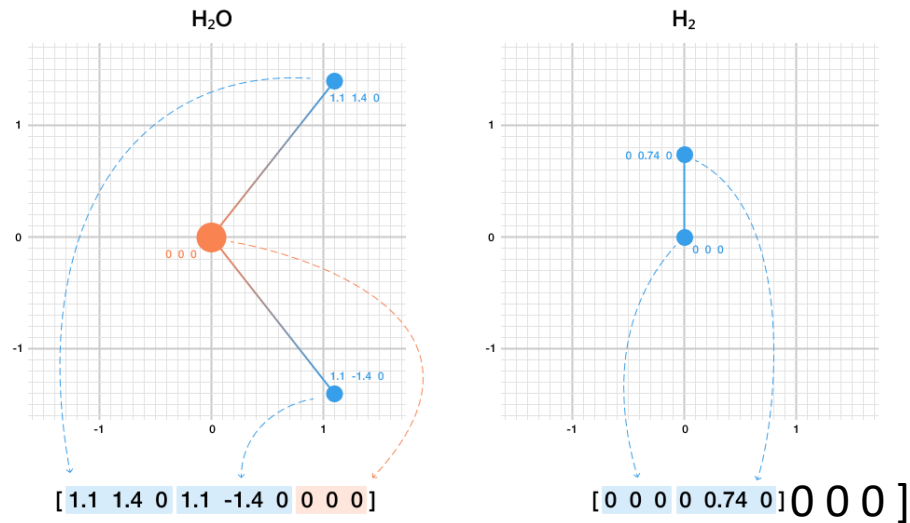
/usr/local/lib/python3.9/dist-packages/numpy/core/overrides.py in concatenate(*args, **kwargs)

ValueError: all the input array dimensions for the concatenation axis must match exactly, but along dimension 1, the array at index 0 has size 9 and the array at index 1 has size 6

SEARCH STACK OVERFLOW

No varying system size

No profit



```
[6] np.concatenate([h2o, h2], axis=0)
```

ValueError

Traceback (most recent call last)

<ipython-input-6-6c19194e29cb> in <cell line: 1>()

----> 1 np.concatenate([h2o, h2], axis=0)

/usr/local/lib/python3.9/dist-packages/numpy/core/overrides.py in concatenate(*args, **kwargs)

ValueError: all the input array dimensions for the concatenation axis must match exactly, but along dimension 1, the array at index 0 has size 9 and the array at index 1 has size 6

SEARCH STACK OVERFLOW

Fine, let's pad with 0's

No permutation invariance

No profit



Charge	x	y	z
1	1.1	-1.4	0
1	1.1	1.4	0
16	0	0	0

$= [1 \ 1.1 \ -1.4 \ 0 \ \mathbf{1 \ 1.1 \ 1.4 \ 0 \ 16 \ 0 \ 0 \ 0}]$

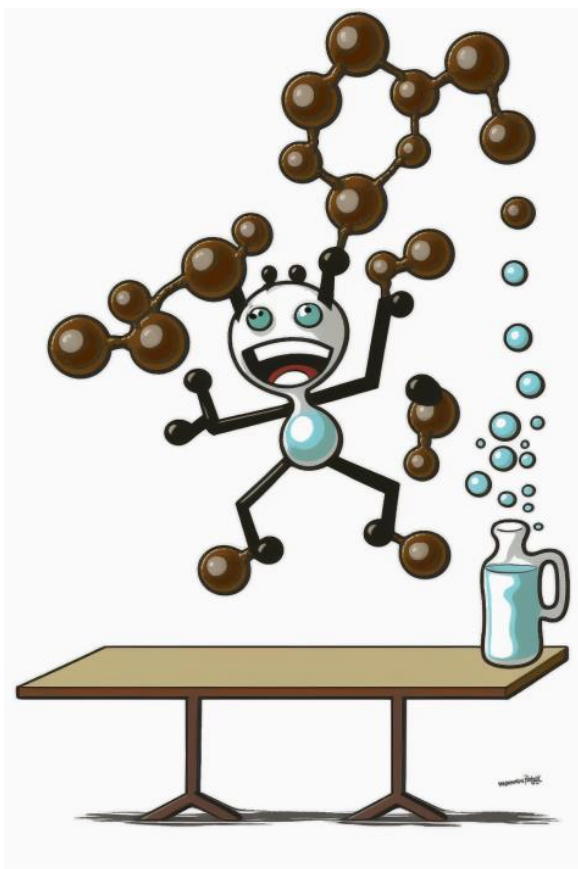
Why not this?

Charge	x	y	z
1	1.1	-1.4	0
16	0	0	0
1	1.1	1.4	0

$= [1 \ 1.1 \ 1.4 \ -1.4 \ 0 \ \mathbf{16 \ 0 \ 0 \ 0 \ 1 \ 1.1 \ 1.4 \ 0}]$

No translation invariance

No profit



Charge	x	y	z
1	1.1	-1.4	0
1	1.1	1.4	0
16	0	0	0

= [1 **1.1** -1.4 0 1 **1.1** 1.4 0 16 **0** 0 0]

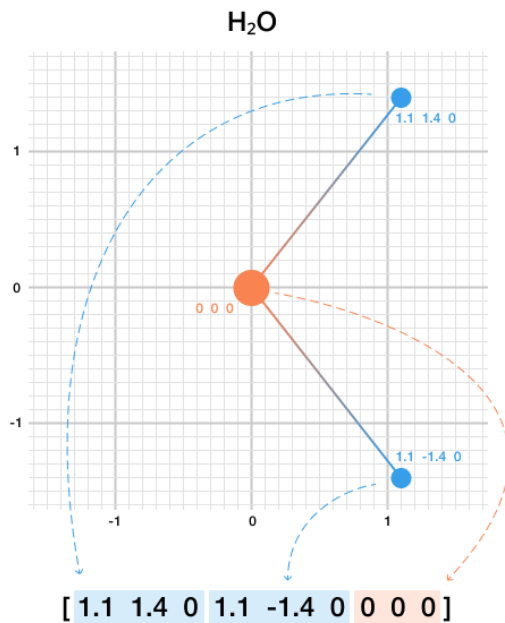
Why not this?

Charge	x	y	z
1	2.1	-1.4	0
1	2.1	1.4	0
16	1	0	0

= [1 **2.1** -1.4 0 1 **2.1** 1.4 0 16 **1** 0 0]

Toy (not really) problem: predict potential energy of a molecule

Naïve attempt



Augment with random permutations

Augment with random rotations and translations

Zero-padding up to maximum molecule size

Given enough data, this might work,
but incredibly data-inefficient



CatBoost

Nikita Kazeev, "Graph neural networks", MLHEP-2023

[illegible]

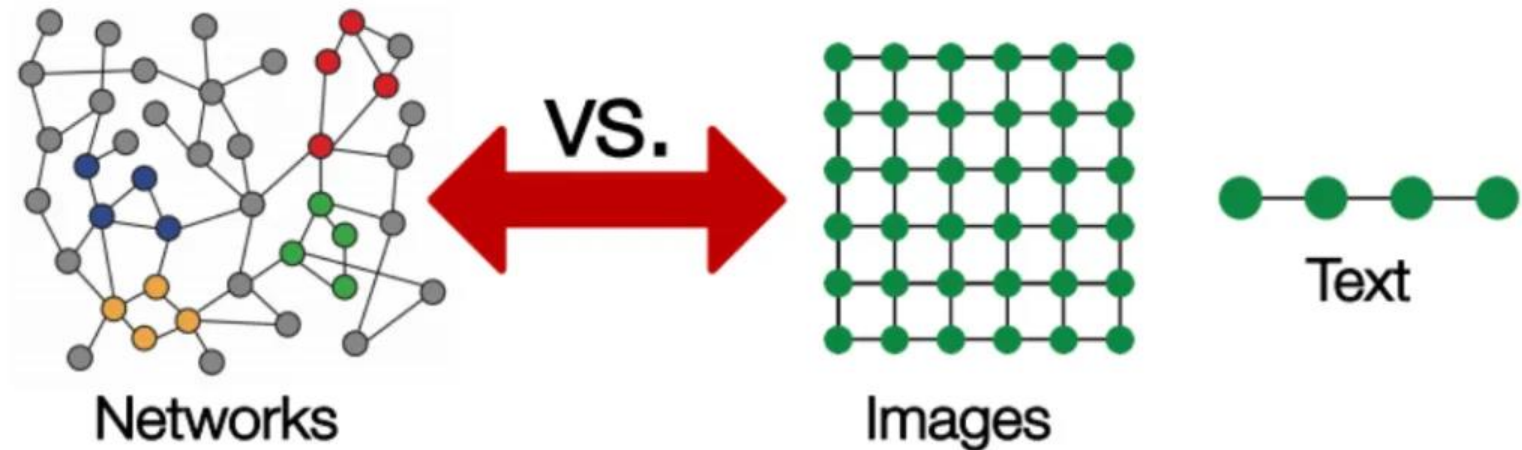
Graph as a representation

Easy to construct, but hard to machine learn

In machine learning inductive bias is everything

Graphs provide:

- Permutation invariance
- Different system size
- Locality of interactions



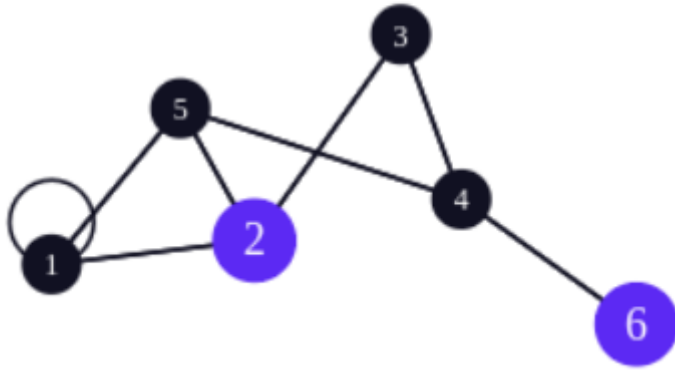
[Source](#)

Graph Neural Network (GNN)

Quizz

- What is the lowest number of edges a graph with V vertices can have?
- What is the highest number of edges an undirected graph without loops with V vertices can have?

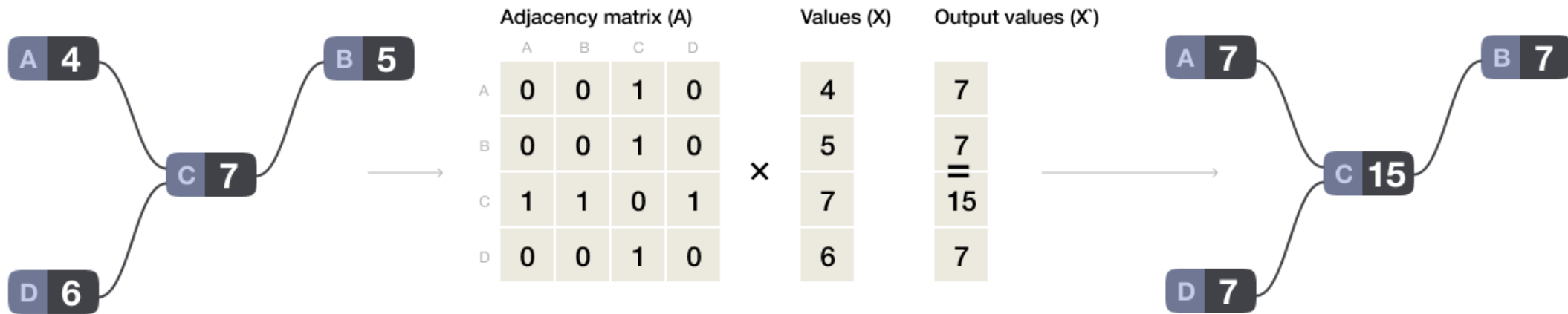
Adjacency matrix



	1	2	3	4	5	6
1	1	1	0	0	1	0
2	1	0	1	0	1	0
3	0	1	0	1	0	0
4	0	0	1	0	1	1
5	1	1	0	1	0	0
6	0	0	0	1	0	0

Hover over matrix to see corresponding nodes

Sum over neighbors as matrix operations



Meet sparse matrices

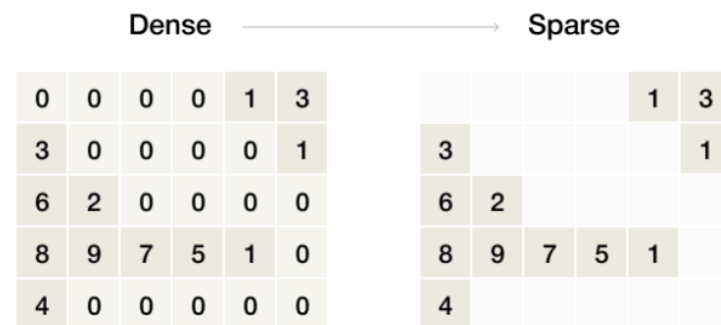
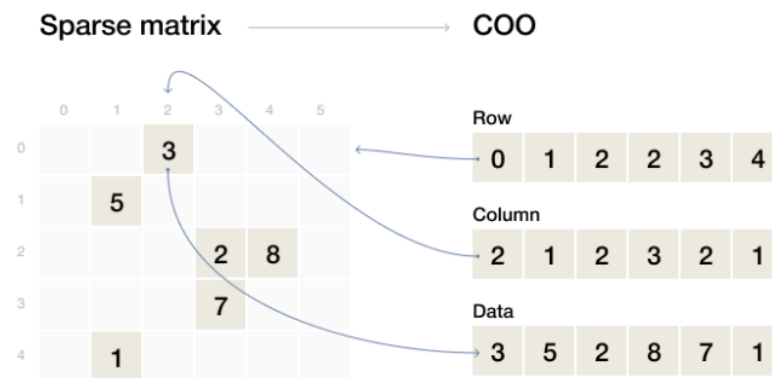


Figure 9. A sparse matrix contains only non-zero elements from a full matrix.



COOrdinate format: we store the coordinates and the element value for each element.

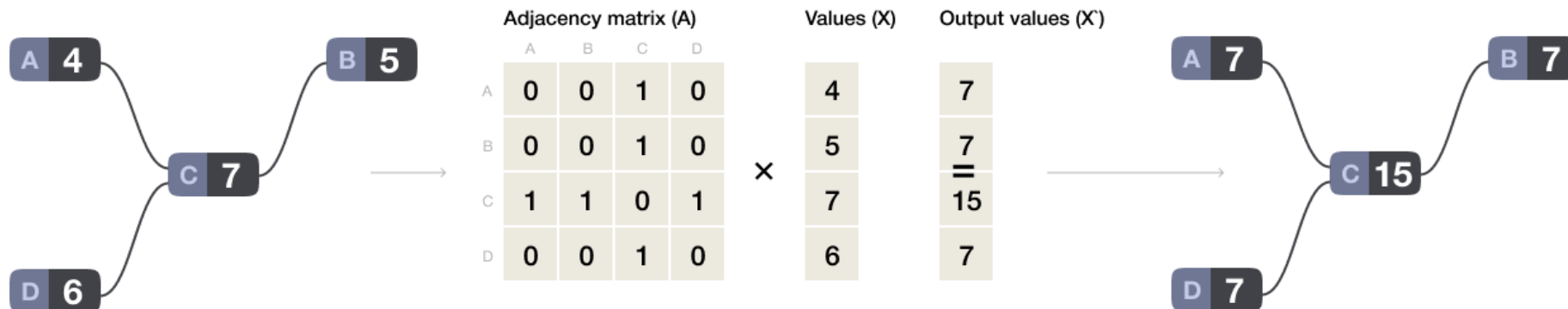
Graph convolution

From image to graph convolution

Input			Kernel		Output	
0	1	2	0	1	19	25
3	4	5	2	3	37	43
6	7	8				

Naive graph convolution. Not the final GCNN!

$$f_{v0.1}(X, A) = A \times X \times W$$



Aspecta!

We didn't include information about the features of the node itself

$$\hat{A} = A + I$$

$$f_{v0.3}(X, A) = \sigma(\hat{A} \times X \times W)$$

Non-linearity, e. g. ReLu

Sum convolution
for the node C:

$$A \quad 1 \quad + \quad B \quad 2 \quad + \quad C \quad 3 \quad + \quad D \quad 0 \quad = \quad 4$$

*Node B is not connected
to C so we ignore it*

Sum convolution
in matrix form:

Adjacency matrix
with loops (\hat{A})

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 \end{bmatrix} \times \begin{bmatrix} 1 \\ 2 \\ 3 \\ 0 \end{bmatrix} = \begin{bmatrix} 6 \\ 3 \\ 4 \\ 4 \end{bmatrix}$$

Values (X)

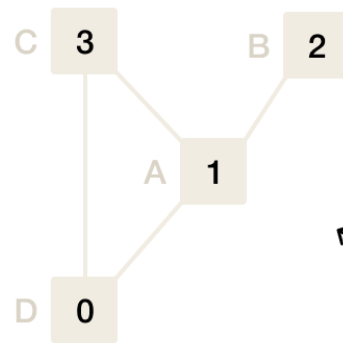
Sum convolution
for the node C

Last touch: normalization

$$f(X, A, W) = \sigma(\hat{D}^{-1/2} \hat{A} \hat{D}^{-1/2} X W).$$

where \hat{D} is a diagonal matrix with the node degrees of \hat{A} , $\hat{D}_{ii} = \sum_j \hat{A}_{ij}$.

Example: stage 0



Intuitive
representation

Mathematical representation
with all the matrices we need

Adjacency matrix (A)

$$\begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix}$$

Adjacency matrix
with loops (\hat{A})

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 \end{bmatrix}$$

Values (X)

$$\begin{bmatrix} 1 \\ 2 \\ 3 \\ 0 \end{bmatrix}$$

Node degree of \hat{A} (\hat{D})

$$\begin{bmatrix} 4 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 3 \end{bmatrix}$$

Inverted node degree roots of \hat{A} ($\hat{D}^{-1/2}$)

$$\begin{bmatrix} .5 & 0 & 0 & 0 \\ 0 & .7 & 0 & 0 \\ 0 & 0 & .58 & 0 \\ 0 & 0 & 0 & .58 \end{bmatrix}$$

Weights
matrix (W)

$$\begin{bmatrix} .5 \end{bmatrix}$$

$(1/3)^{0.5}$

Example: stage 2

Average
convolution
for the node C:

$$(A \ 1 + B \ 2 + C \ 3 + D \ 0) / 3 = 1.3$$

*Node B is not connected
to C so we ignore it*

Normalized
average
convolution
in matrix form:

$$\begin{matrix} \text{Inverted node degree} \\ \text{roots of } \hat{A} (\hat{D}^{-1/2}) \end{matrix} \begin{bmatrix} .5 & 0 & 0 & 0 \\ 0 & .7 & 0 & 0 \\ 0 & 0 & .58 & 0 \\ 0 & 0 & 0 & .58 \end{bmatrix} \times \begin{matrix} \text{Adjacency matrix} \\ \text{with loops } (\hat{A}) \end{matrix} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 \end{bmatrix} \times \begin{matrix} \text{Inverted node degree} \\ \text{roots of } \hat{A} (\hat{D}^{-1/2}) \end{matrix} \begin{bmatrix} .5 & 0 & 0 & 0 \\ 0 & .7 & 0 & 0 \\ 0 & 0 & .58 & 0 \\ 0 & 0 & 0 & .58 \end{bmatrix} \times \begin{matrix} \text{Values (X)} \\ \begin{bmatrix} 1 \\ 2 \\ 3 \\ 0 \end{bmatrix} \end{matrix} = \begin{bmatrix} 1.8 \\ 1.3 \\ 1.3 \\ 1.3 \end{bmatrix}$$

*Average
convolution
for the node C*

Example: stage 3

Neural
network
for the node C

$$\sigma (\overset{\text{Convolved value for C}}{1.3} \times \overset{\text{Weight}}{.5}) = \overset{\text{Output}}{.57}$$

Neural
network
in matrix form:

$$\sigma \left(\overset{\text{Matrix after convolution}}{\begin{bmatrix} 1.8 \\ 1.3 \\ 1.3 \\ 1.3 \end{bmatrix}} \times \overset{\text{Weights matrix (W)}}{\begin{bmatrix} .5 \end{bmatrix}} \right) = \begin{bmatrix} .72 \\ .57 \\ \textbf{.57} \\ .57 \end{bmatrix}$$

← Neural network output for the node C

Graph convolution: summary

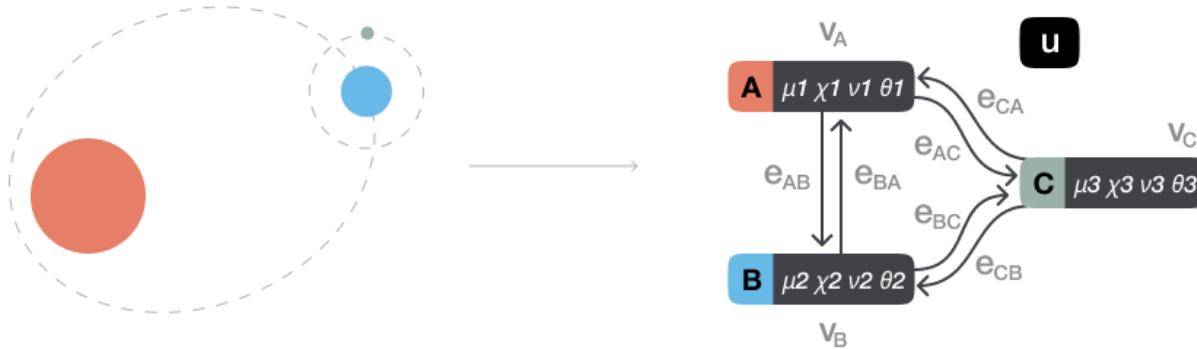
- Similar to convolution in images
- One layer averages the values among the neighbouring nodes
- With trainable weights and non-linearity, can be combined into a deep neural network

General GNN

GraphNN at a glance

- Input: graph, each edge and node has a state vector; global state vector
- Output: graph; global state vector
- Doesn't change connectivity
- Steps:
 - Compute new edge states
 - Compute new node states
 - Compute new global state

Example system and definitions

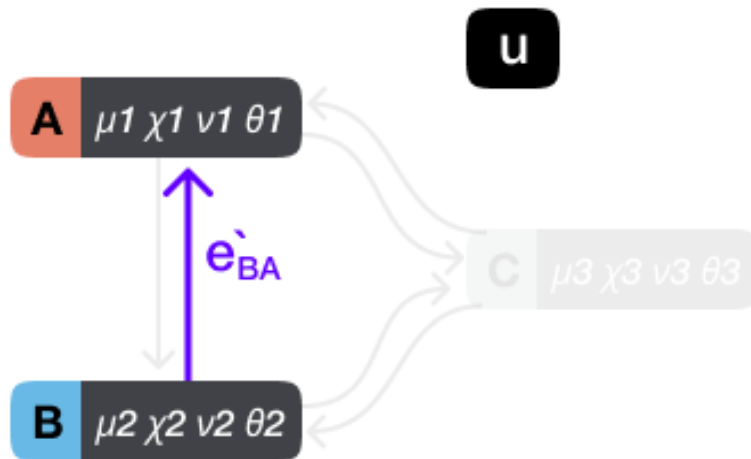


1. $\mathbf{u} \in \mathbb{R}^G$ is the global attribute, such as gravitational field;
2. $V = \{\mathbf{v}_i\}_{i=1:N^v}$ is the set of the attributes of the nodes. For the n -body problem \mathbf{v}_i is a concatenation of position χ_i , velocity ν_i and mass μ_i
3. $E = \{(\mathbf{e}_k, r_k, s_k)\}_{k=1:N^e}$ is a set of directed edges, where vector \mathbf{e}_k is attribute of the edge, r_k is the index of the receiver node, and s_k is the index of the sender node. For the n -body problem, in the input we set $\mathbf{e}_k = \chi_{r_k} - \chi_{s_k}$, the relative positions of the bodies.

Edges update

$$\mathbf{e}'_k = \phi^e(\mathbf{e}_k, \mathbf{v}_{r_k}, \mathbf{v}_{s_k}, \mathbf{u})$$

MLP



Aggregation of the edges' states

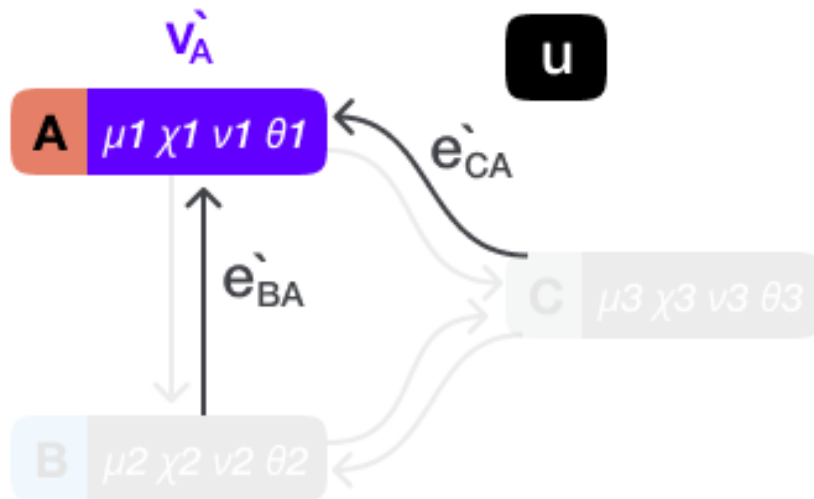
$$\bar{\mathbf{e}}_i = \rho^{e \rightarrow v} (E'_i),$$

where $E'_i = \{(\mathbf{e}'_k, r_k, s_k)\}_{r_k=i, k=1:N^e}$ is the set of all edges incident to i —th vertex; $\rho^{e \rightarrow v}$ is the aggregation function. In the n —body example, it is sum, as the total force equals to the sum of forces. Common other choices are min, max and average.

Node update

$$\mathbf{v}'_i = \phi^v(\bar{\mathbf{e}}_i, \mathbf{v}_i, \mathbf{u}),$$

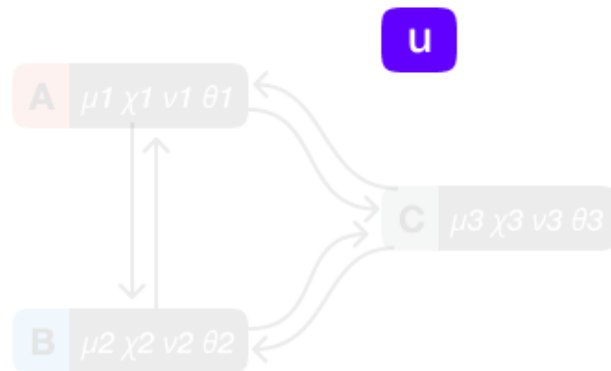
where ϕ^v is an arbitrary function commonly represented by a fully-connected neural network. The new node state depends on the states of all the incoming edges, the current node state, and the global state. In the n -body example, this is the velocity and position update.



Aggregation of states of all the edges and nodes and global state update

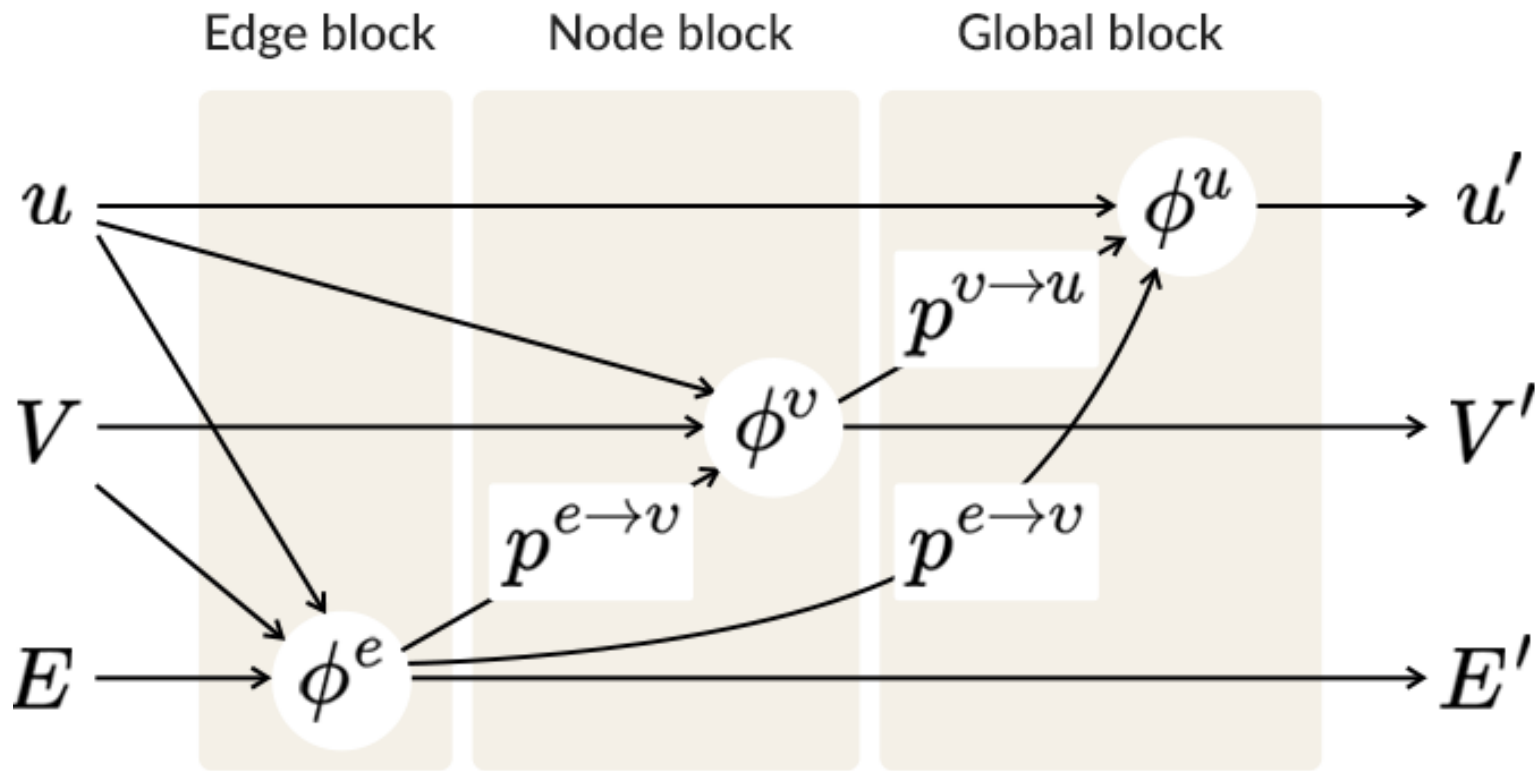
$$\bar{\mathbf{e}}' = \rho^{e \rightarrow u}(E')$$

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



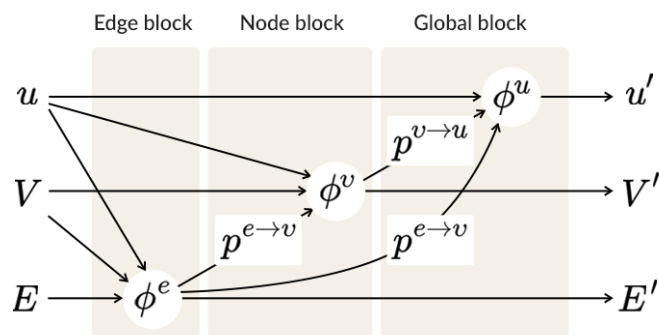
GNN block summary

(a) Full GN block

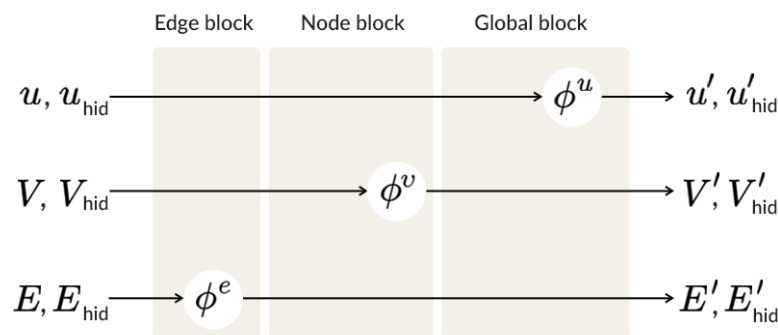


GNN kinds

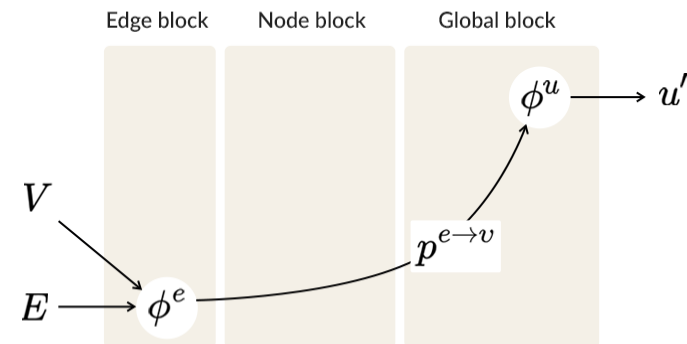
(a) Full GN block



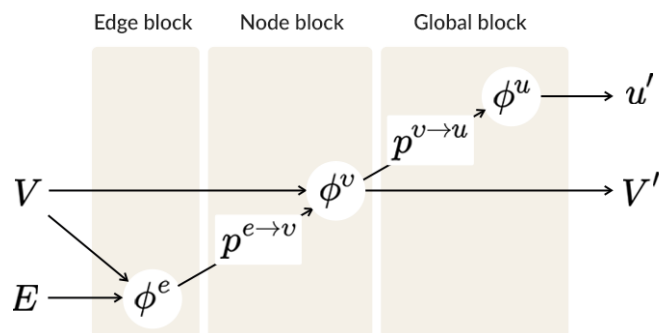
(b) Independent recurrent block



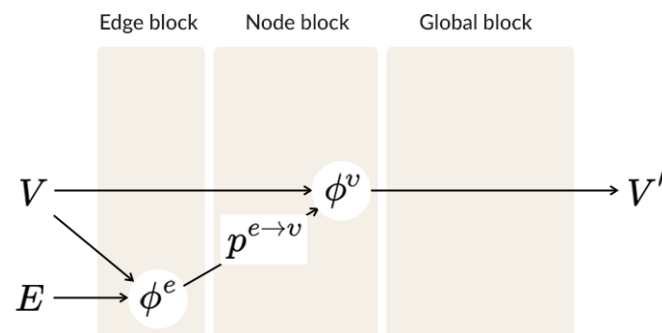
(e) Relation network



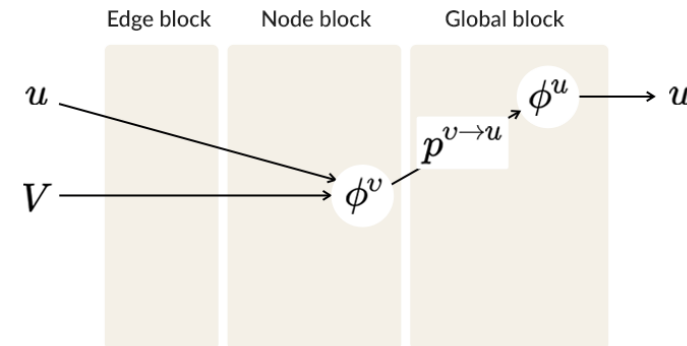
(c) Message-passing neural network



(d) Non-local neural network



(f) Deep set



Deep GraphNN

- Input has the same structure as output, stack layers to make it deep
- Can mark any of the states as the output
- Trainable via back-propagation
- Suitable for predicting global, node and edge targets

Graph neural networks in HEP

Jet tagging

[Moreno, Eric A., et al. "JEDI-net: a jet identification algorithm based on interaction networks." The European Physical Journal C 80 \(2020\): 1-15.](#)

Physics

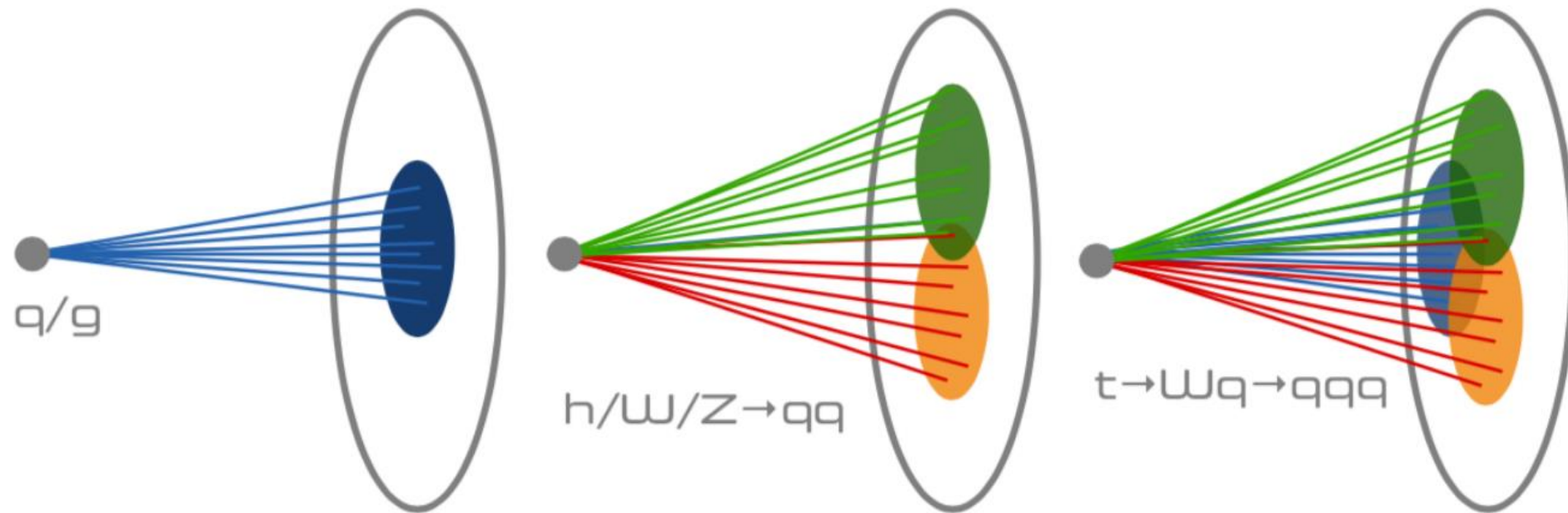
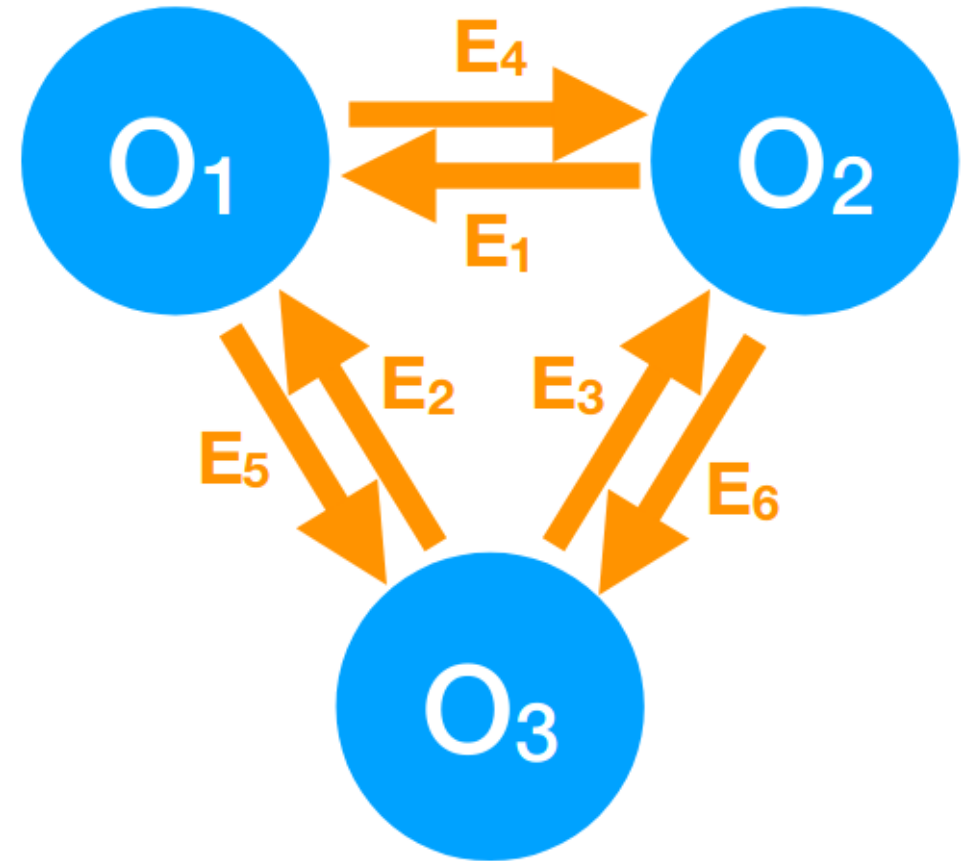


Fig. 1 Pictorial representations of the different jet categories considered in this paper. Left: jets originating from quarks or gluons produce one cluster of particles, approximately cone-shaped, developing along the flight direction of the quark or gluon that started the cascade. Center: when produced with large momentum, a heavy boson decaying to quarks would result in a single jet, made of 2 particle clusters (usually referred to as prongs). Right: a high-momentum $t \rightarrow Wb \rightarrow q\bar{q}'b$ decay chain results in a jet composed of three prongs.

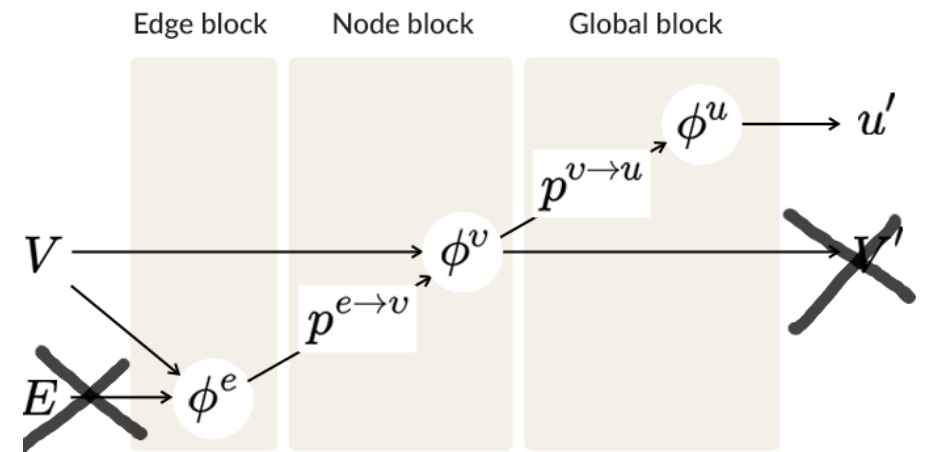
Graph construction

- Nodes: particles
- Node features: momentum (p_x, p_y, p_z) , energy E , absolute, and relative to the jet, 16 in total
- Connectivity: directional, full
- Edge features: None



GNN

- Single MLP layer of edge transformation
 - Input: 2 vertices' features
- Single MLP layer of vertex transformation
 - Edge features aggregated with sum
- Single MLP layer of global output
 - Input: vertices features
 - Aggregation:
 - Sum
 - None, particles ordered by p_t



Almost message-passing GNN

Results

Jet category	DNN	GRU	CNN	JEDI-net	JEDI-net with $\sum O$
TPR for FPR=10%					
gluon	0.830 ± 0.002	0.740 ± 0.014	0.700 ± 0.008	0.878 ± 0.001	0.879 ± 0.001
light quarks	0.715 ± 0.002	0.746 ± 0.011	0.740 ± 0.003	0.822 ± 0.001	0.818 ± 0.001
W boson	0.855 ± 0.001	0.812 ± 0.035	0.760 ± 0.005	0.938 ± 0.001	0.927 ± 0.001
Z boson	0.833 ± 0.002	0.753 ± 0.036	0.721 ± 0.006	0.910 ± 0.001	0.903 ± 0.001
top quark	0.917 ± 0.001	0.867 ± 0.006	0.889 ± 0.001	0.930 ± 0.001	0.931 ± 0.001
TPR for FPR=1%					
gluon	0.420 ± 0.002	0.273 ± 0.018	0.257 ± 0.005	0.485 ± 0.001	0.482 ± 0.001
light quarks	0.178 ± 0.002	0.220 ± 0.037	0.254 ± 0.007	0.302 ± 0.001	0.301 ± 0.001
W boson	0.656 ± 0.002	0.249 ± 0.057	0.232 ± 0.006	0.704 ± 0.001	0.658 ± 0.001
Z boson	0.715 ± 0.001	0.386 ± 0.060	0.291 ± 0.005	0.769 ± 0.001	0.729 ± 0.001
top quark	0.651 ± 0.003	0.426 ± 0.020	0.504 ± 0.005	0.633 ± 0.001	0.632 ± 0.001

Table 3 True positive rates (TPR) for the optimized JEDI-net taggers and the three alternative models (DNN, CNN, and GRU), corresponding to a false positive rate (FPR) of 10% (top) and 1% (bottom). The largest TPR value for each case is highlighted in bold.

Jet tagging: advanced

[Gong, Shiqi, et al. "An efficient Lorentz equivariant graph neural network for jet tagging." Journal of High Energy Physics 2022.7 \(2022\): 1-22.](#)

Physics

The jet tagging result should not depend on the spatial orientation or the Lorentz boost of a jet.

Data

- [G. Kasieczka, T. Plehn and M. Russel, Top quark tagging reference dataset \(2019\)](#)
 - Simulated
 - ATLAS detector
 - 1.2M training entries, 400k validation entries, and 400k testing entries
 - entries represents a single jet whose origin is either an energetic top quark, a light quark, or a gluon
- Quark-gluon tagging
 - Simulated
 - No detector simulation
 - 1.6M/200K/200K for training, validation, and testing.
 - $Z(\rightarrow \nu\nu) + (u, d, s)$ and $Z(\rightarrow \nu\nu) + g$

Graph construction

- Nodes: particles
- Node features:
 - 4-momentum vector $v_i = (E^i, p_x^i, p_y^i, p_z^i)$
 - $s_i = (s_1^i, s_2^i, \dots, s_\alpha^i)$ denote the scalars, such as mass, charge and particle identity information, etc.
- Connectivity: full
- Edge features: None

Recap: Minkowski metric

Minkowski metric. Consider the 4-dimensional space-time \mathbb{R}^4 with basis $\{e_i\}_{i=0}^3$. We define a bilinear form $\eta : \mathbb{R}^4 \times \mathbb{R}^4 \rightarrow \mathbb{R}$ as follows. For $u, v \in \mathbb{R}^4$, we set $\eta(u, v) = u^T J v$ where $J = \text{diag}(1, -1, -1, -1)$ is the Minkowski metric. The Minkowski inner product of two vectors $u = (t, x, y, z)$ and $v = (t', x', y', z')$ is defined as $\langle u, v \rangle = \eta(u, v) = tt' - xx' - yy' - zz'$. The Minkowski norm of a vector $u = (t, x, y, z)$ is defined to be $\|u\| = \sqrt{\eta(u, u)} = \sqrt{t^2 - x^2 - y^2 - z^2}$.

Recap: Lorentz group

Lorentz transformation. The Lorentz group is defined to be the set of all matrices that preserve the bilinear form η . Restricting the inertial frames to be positively oriented and positively time-oriented, we obtain proper orthochronous Lorentz group, denoted as $SO(1,3)^+$. The 3d spatial rotation group $SO(3)$ is a subgroup of $SO(1,3)^+$. Additionally, the Lorentz boost is also included. Given two inertial frames $\{e_i\}_{i=0}^3$ and $\{e'_i\}_{i=0}^3$, the relative velocity vector β and the boost factor γ are defined by $e'_0 = \gamma e_0 + \sum_{i=1}^3 \gamma \beta_i e_i$ where $\gamma = (1 - \beta^2)^{-1/2}$.

Recap: Lorentz group equivariance

Lorentz group equivariance. Let $T_g : V \rightarrow V$ and $S_g : U \rightarrow U$ be group actions of $g \in G$ on sets V and U , respectively. We say a function $\phi : V \rightarrow U$ is equivariant to group G if

$$\phi(T_g(v)) = S_g(\phi(v)) \quad (2.1)$$

holds for all $v \in V$ and $g \in G$. In this work, we only consider the case that the type of the output is a scalar or vector. Therefore, we explore the following equivariance on a set of particles $V \in \mathbb{R}^{N \times 4}$. Let Q be the Lorentz transformation, the Lorentz equivariance of $\phi(\cdot)$ means:

$$Q\phi(v) = \phi(Qv), \quad \text{for } \phi(v) \in \mathbb{R}^4; \quad (2.2)$$

$$\phi(v) = \phi(Qv), \quad \text{for } \phi(v) \in \mathbb{R}. \quad (2.3)$$

Note that when the output is a scalar, the group equivariance equals the group invariance.

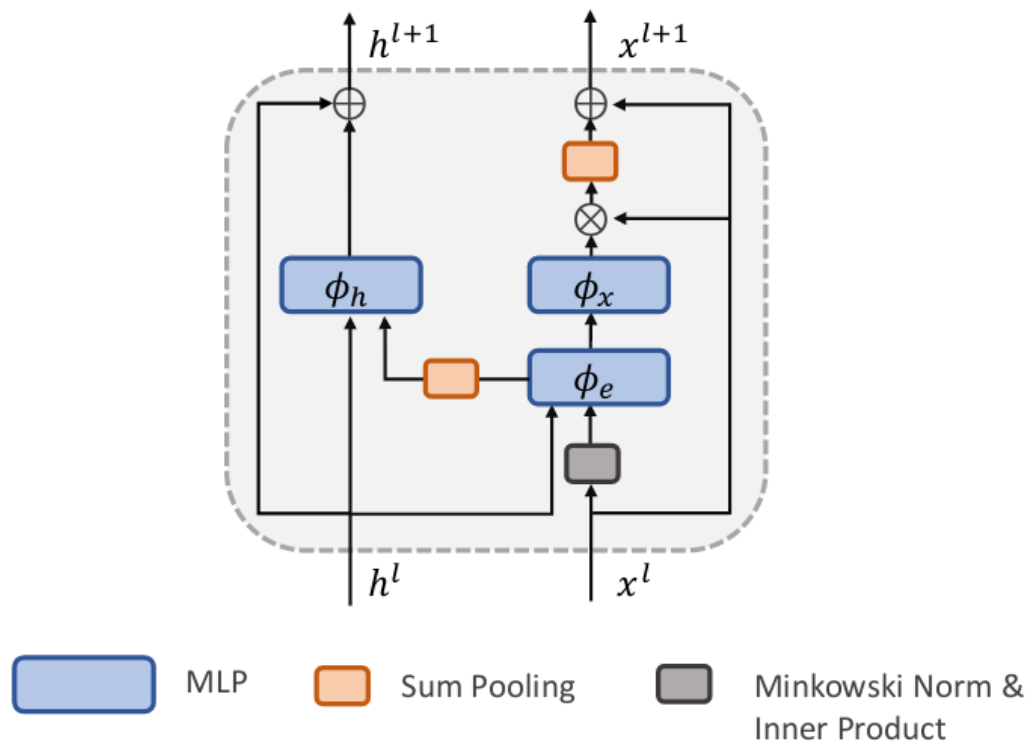
The core idea

Proposition 3.1. [64] *A continuous function $\phi : (\mathbb{R}^{N \times 4}) \rightarrow \mathbb{R}^4$ is Lorentz-equivariant if and only if*

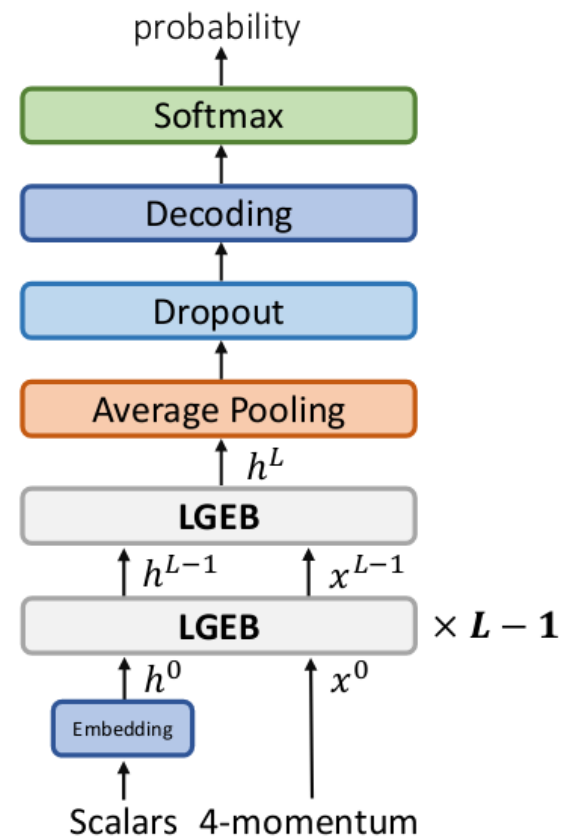
$$\phi(v_1, v_2, \dots, v_N) = \sum_{i=1}^N g_i(\langle v_i, v_j \rangle_{j=1}^N) v_i, \quad (3.1)$$

where g_i are continuous Lorentz-invariant scalar functions, and $\langle \cdot, \cdot \rangle$ is the Minkowski inner product.

GNN



Lorentz Group Equivariant Block (LGEGB)



LorentzNet

Results: top tagging

Model	Accuracy	AUC	$1/\varepsilon_B$ ($\varepsilon_S = 0.5$)	$1/\varepsilon_B$ ($\varepsilon_S = 0.3$)
ResNeXt	0.936	0.9837	302 ± 5	1147 ± 58
P-CNN	0.930	0.9803	201 ± 4	759 ± 24
PFN	0.932	0.9819	247 ± 3	888 ± 17
ParticleNet	0.940	0.9858	397 ± 7	1615 ± 93
EGNN	0.922	0.9760	148 ± 8	540 ± 49
LGN	0.929	0.9640	124 ± 20	435 ± 95
LorentzNet	0.942	0.9868	498 ± 18	2195 ± 173

Table 1. Performance comparison between LorentzNet and other representative algorithms on top tagging dataset. The results for LorentzNet and EGNN are averaged on 6 runs. The results for other baselines are referred to [36, 37, 61].

Results: quark-gluon tagging

Model	Accuracy	AUC	$1/\varepsilon_B$ ($\varepsilon_S = 0.5$)	$1/\varepsilon_B$ ($\varepsilon_S = 0.3$)
ResNeXt	0.821	0.8960	30.9	80.8
P-CNN	0.827	0.9002	34.7	91.0
PFN	—	0.9005	34.7 ± 0.4	—
ParticleNet	0.840	0.9116	39.8 ± 0.2	98.6 ± 1.3
EGNN	0.803	0.8806	26.3 ± 0.3	76.6 ± 0.5
LGN	0.803	0.8324	16.0	44.3
LorentzNet	0.844	0.9156	42.4 ± 0.4	110.2 ± 1.3

Table 2. Performance comparison between LorentzNet and other representative algorithms on quark-gluon tagging dataset. The results for LorentzNet, EGNN and LGN are averaged on 6 runs. The results for other baselines are referred to [36, 37].

Reconstructing particle showers in electromagnetic calorimeters in OPERA

Belavin, Vladislav, Ekaterina Trofimova, and Andrey Ustyuzhanin. "Segmentation of EM showers for neutrino experiments with deep graph neural networks." Journal of Instrumentation 16.12 (2021): P12035.

Physics

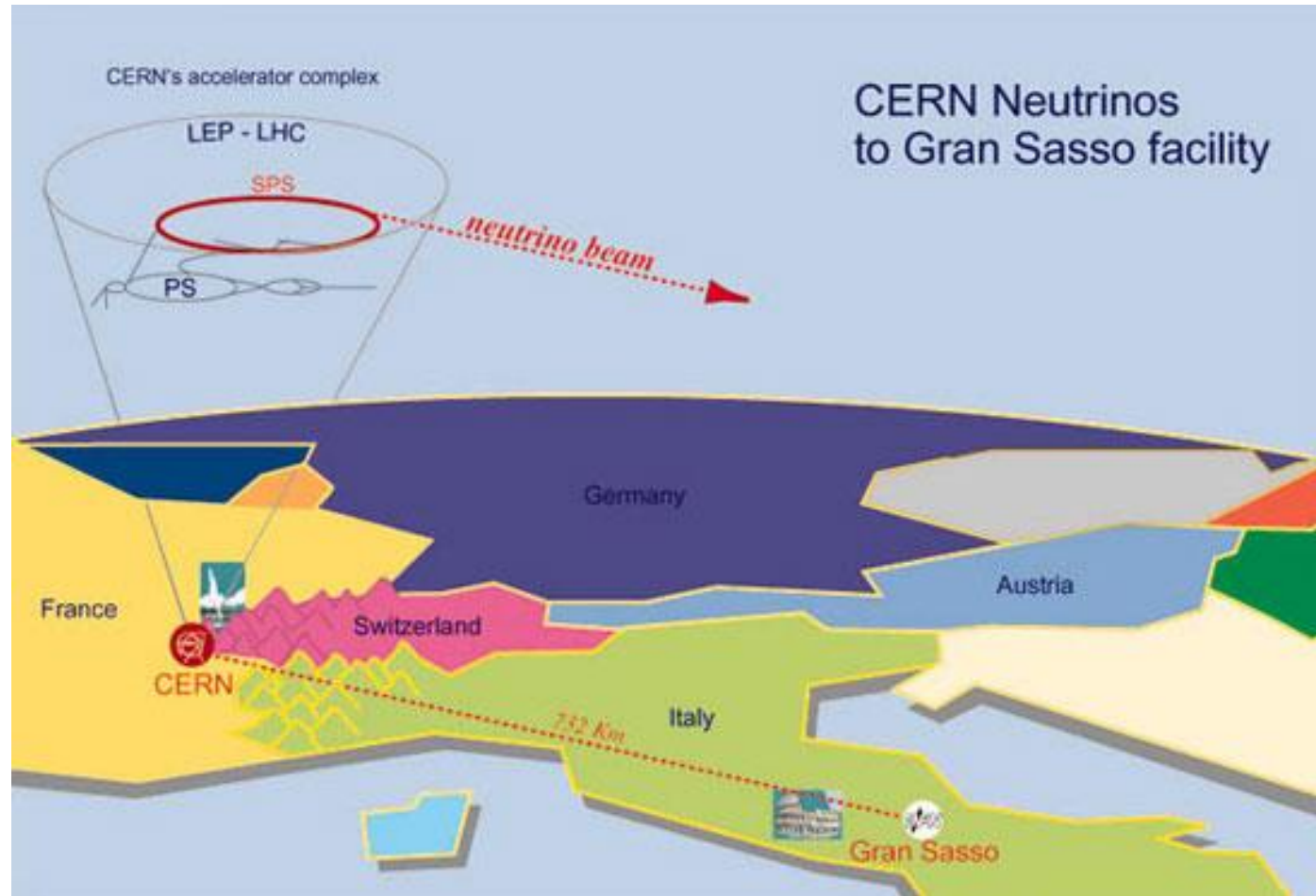
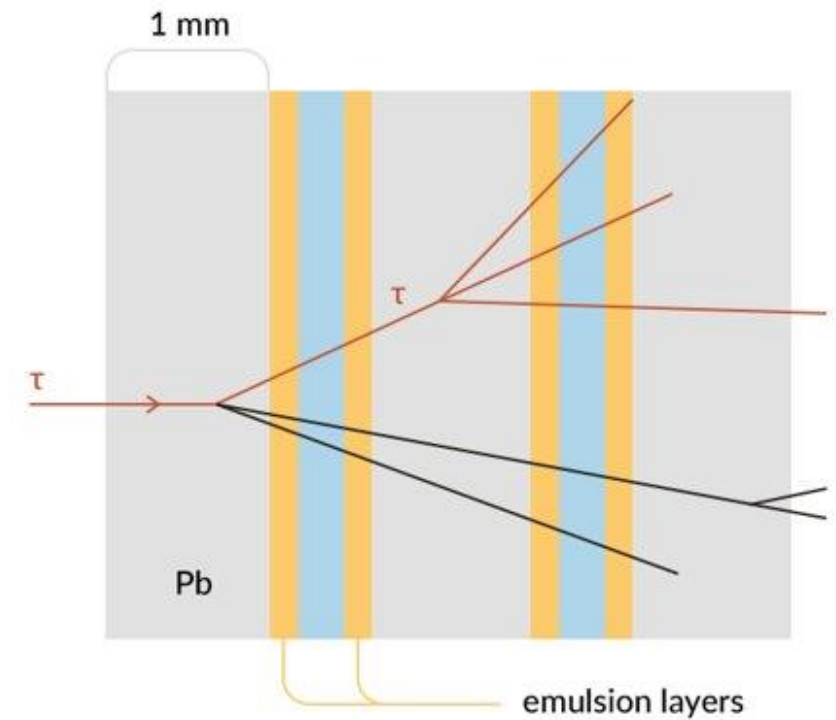
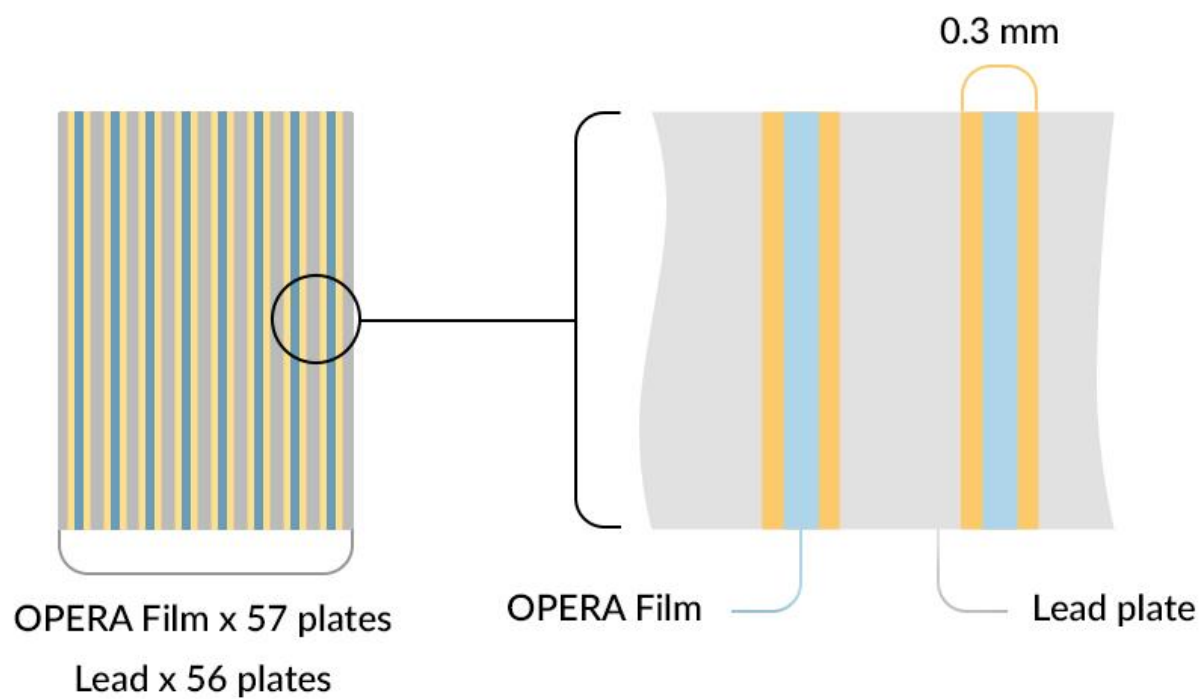


Image: CERN

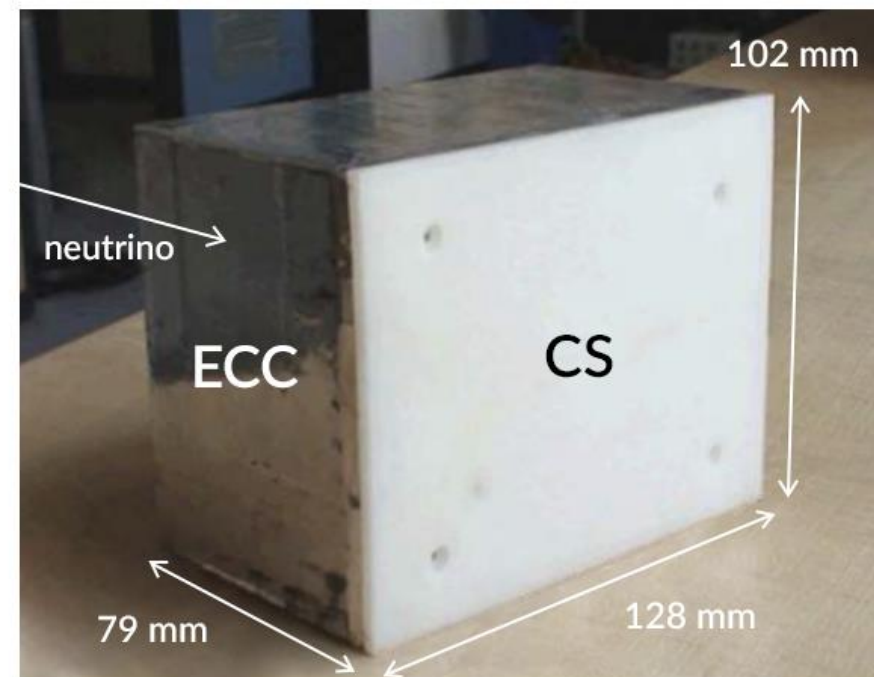


[Source](#)

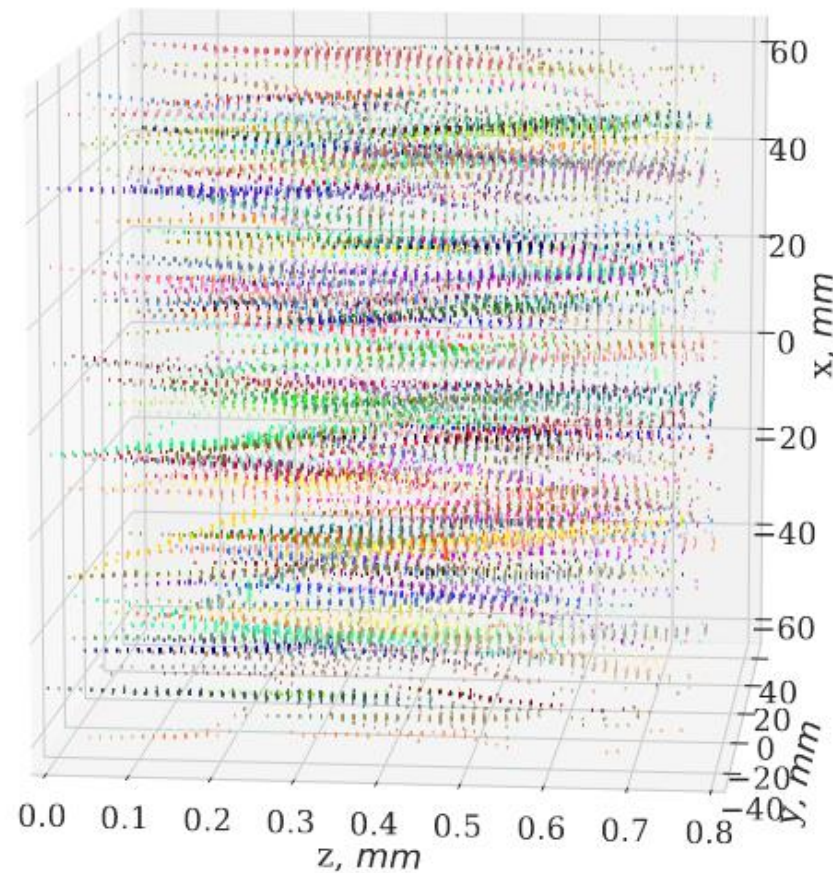
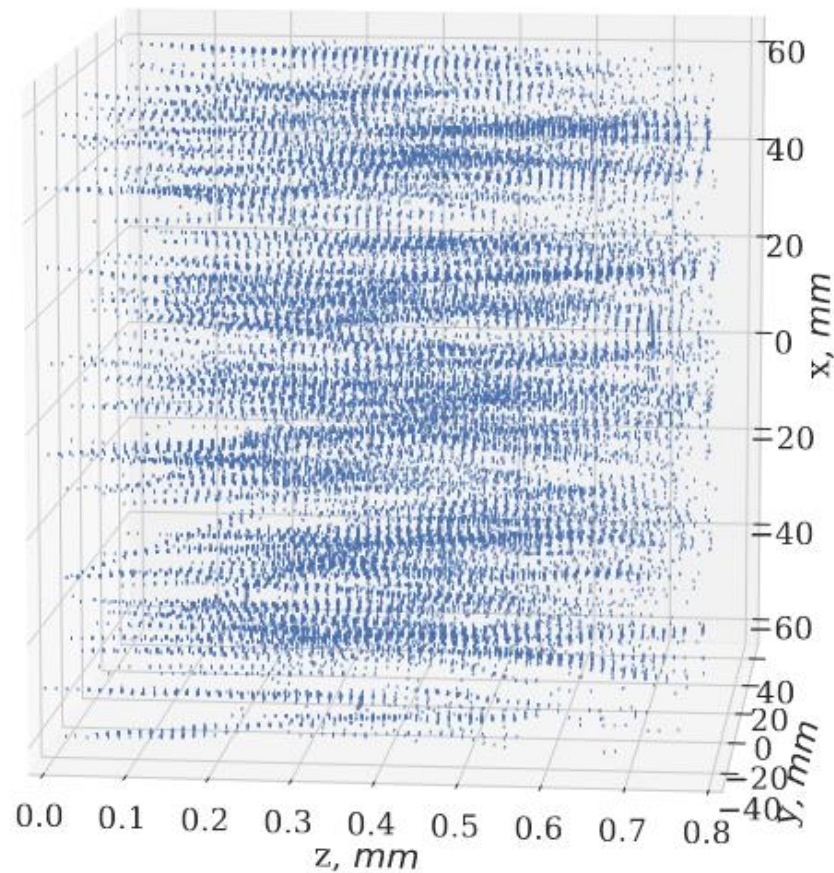
Detector



[Source](#)

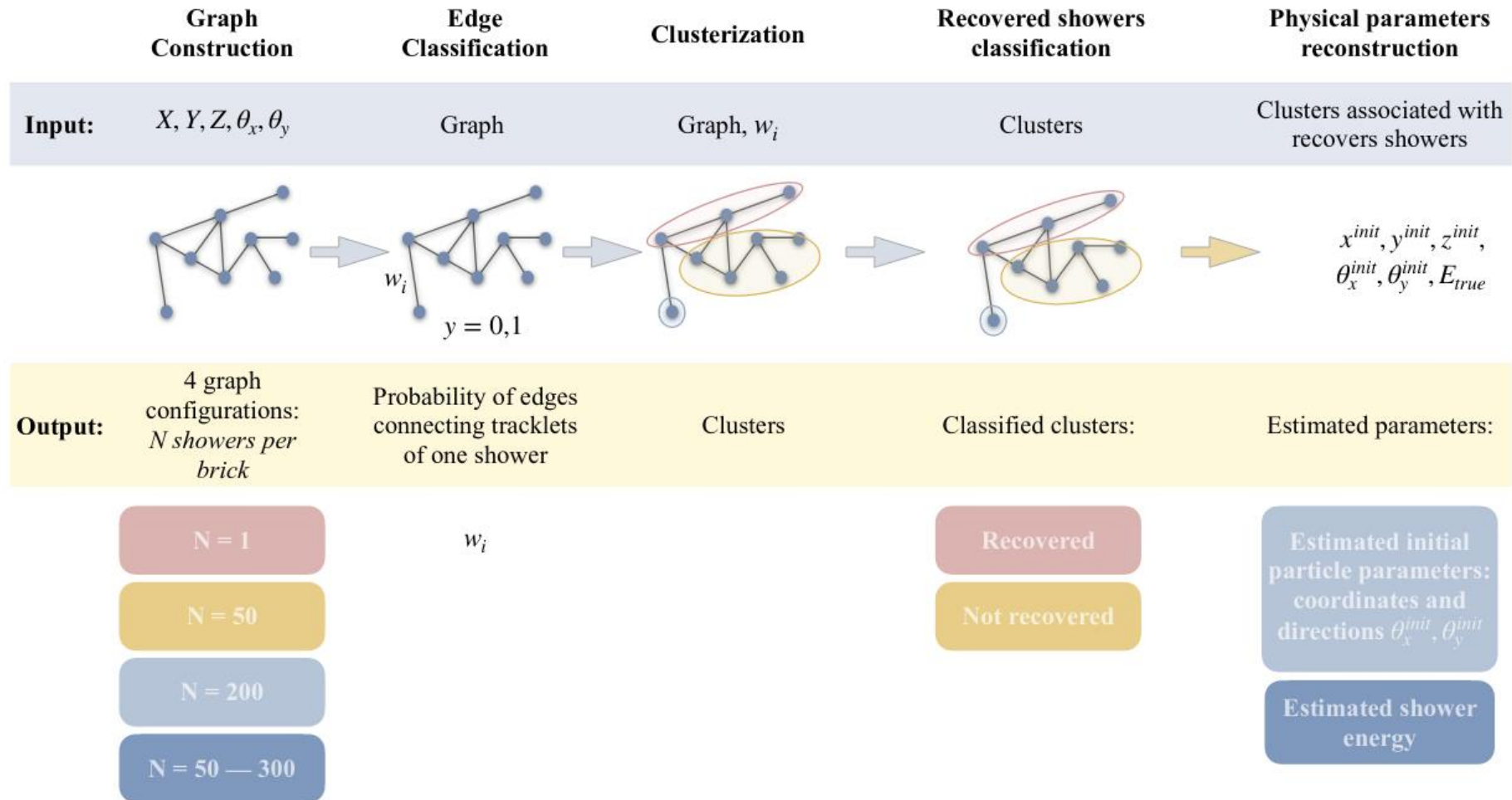


Event



[Source](#)

Algorithm overview



Graph construction

- Vertices: tracklets

- Vertex features

1. Raw parameters of the tracklet: $x, y, z, \theta_x, \theta_y$

2. Engineered trigonometric features: $\phi = \arctan\left(\frac{y}{x}\right), \frac{\sqrt{x^2+y^2}}{z}, \frac{x}{z}, \frac{y}{z}, \frac{\sin(\phi)+\cos(\phi)}{\phi}$

- Connections: 10 nearest neighbours, as defined by integral distance (red area in the figure)

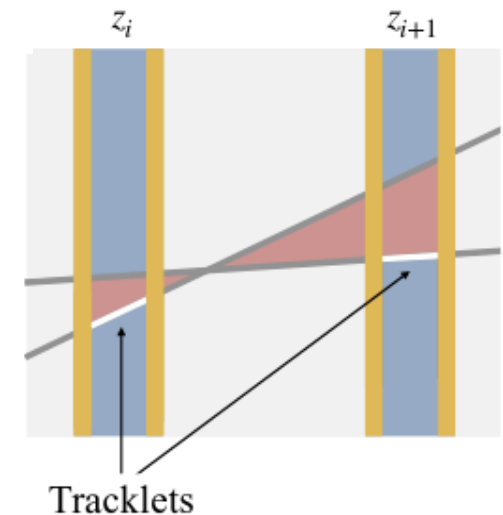
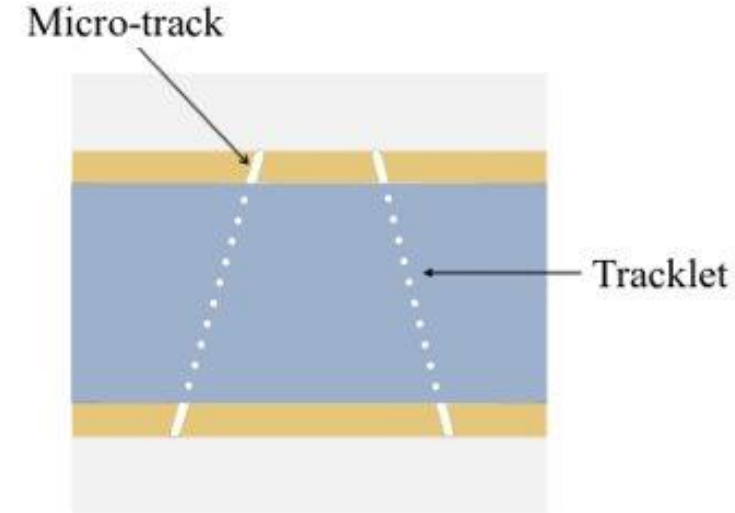
- Edge features:

1. Integral distance [1]

2. Impact parameter (IP) projections on the X axis: $\frac{x_1-x_2-(z_1-z_2)\cdot\theta_{x_i}}{z_1-z_2}; i \in \{1, 2\}$

3. IP projections on the Y axis

4. Tracklet pairs energy and likelihood estimates

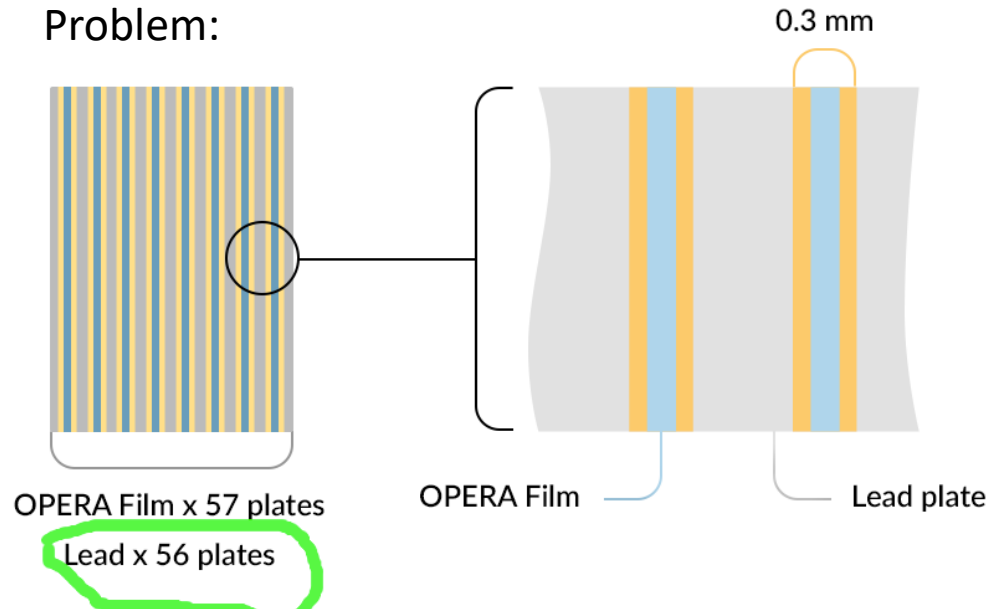


EmulsionConv

Start with EdgeConv

$$\mathbf{e}'_k = \phi^e(\mathbf{e}_k, \mathbf{v}_{r_k}, \mathbf{v}_{s_k} - \mathbf{v}_{r_k})$$
$$\mathbf{v}'_i = \frac{\sum_{k:r_k=i} \mathbf{e}'_k + \mathbf{v}_i}{2}$$

Problem:

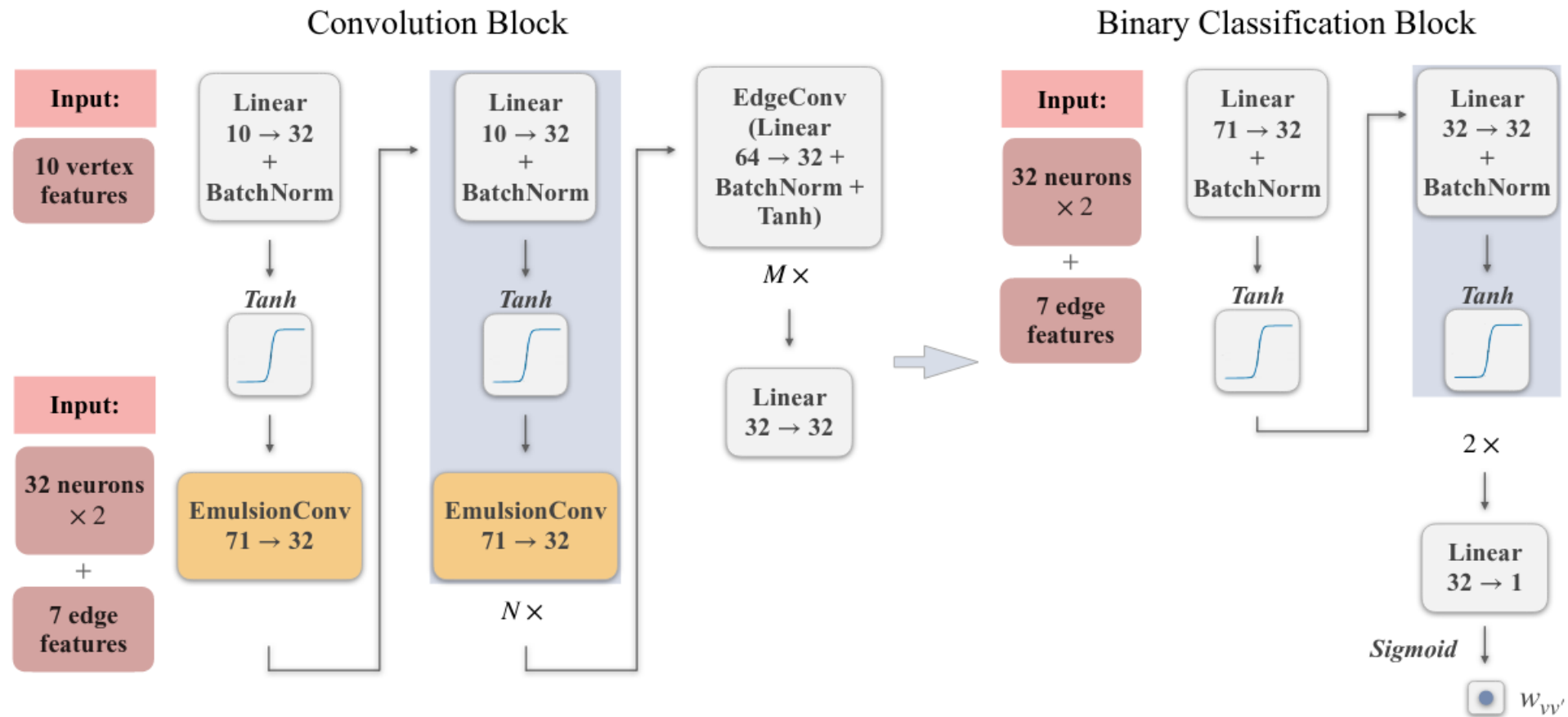


We need 56 EdgeConv layers to propagate through the brick

Solution:

1. Don't run all edge updates, and then vector updates, run them in the order of z
2. In each film, run full EdgeConv
3. In between the films, use the new vertex states

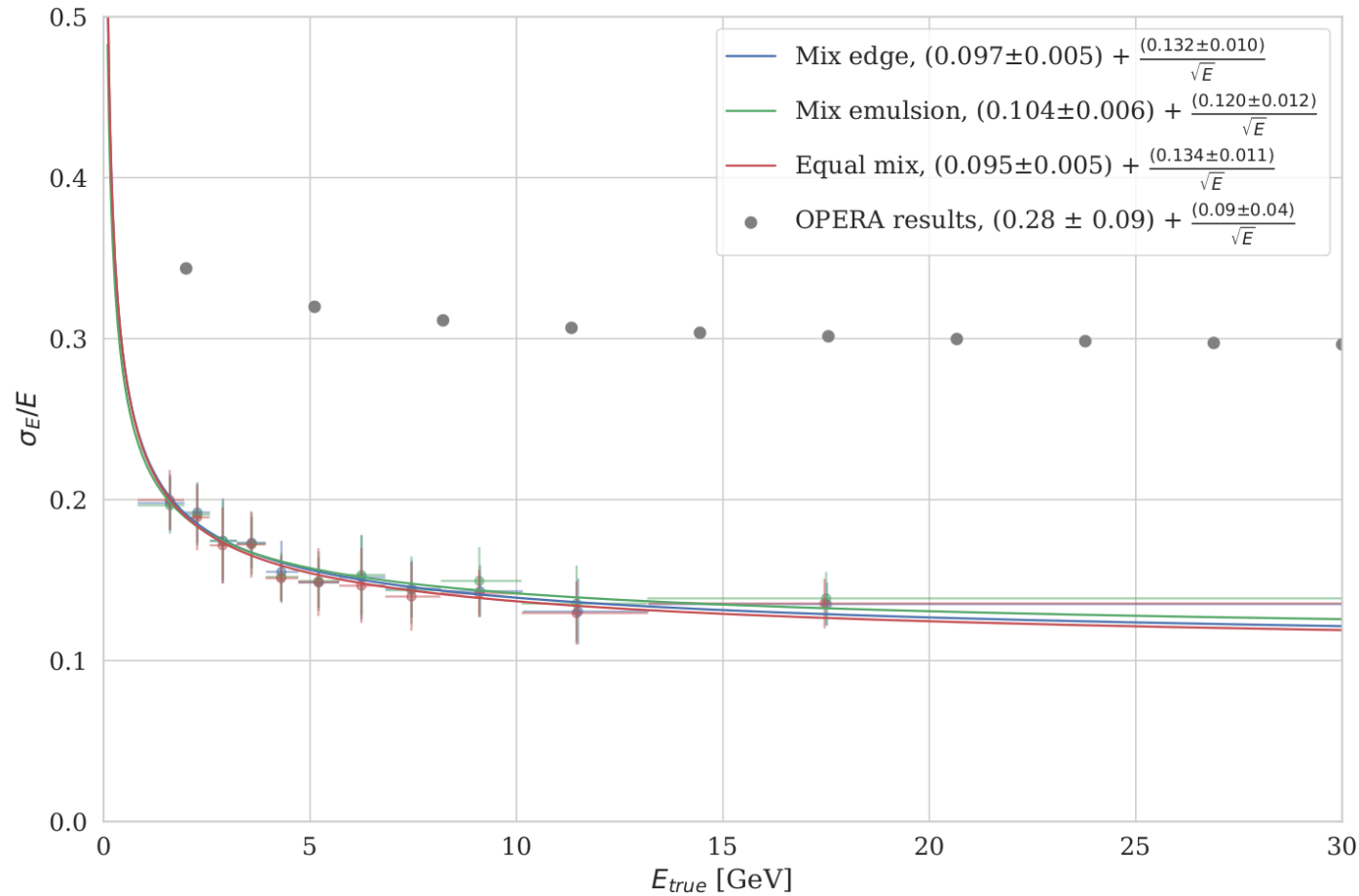
GNN predicting the probability that an edge connects tracklets from the same shower



Reconstruction

- Cluster tracklets into showers using GNN predictions as distance
 - Using modified HDBSCAN
- Predict whether the showers are well-recovered and are suitable for reconstruction
 - XGBoost over aggregated features
- Reconstruction!
 - The initial particles coordinates are estimated from the median position of the first three tracklets in the cluster along the z axis.
 - The direction is estimated by fitting a straight line to the first 20 tracklets.
 - The energy of the incoming particle is reconstructed with a linear regression as a function of the number of tracklets in the recovered shower.

Results



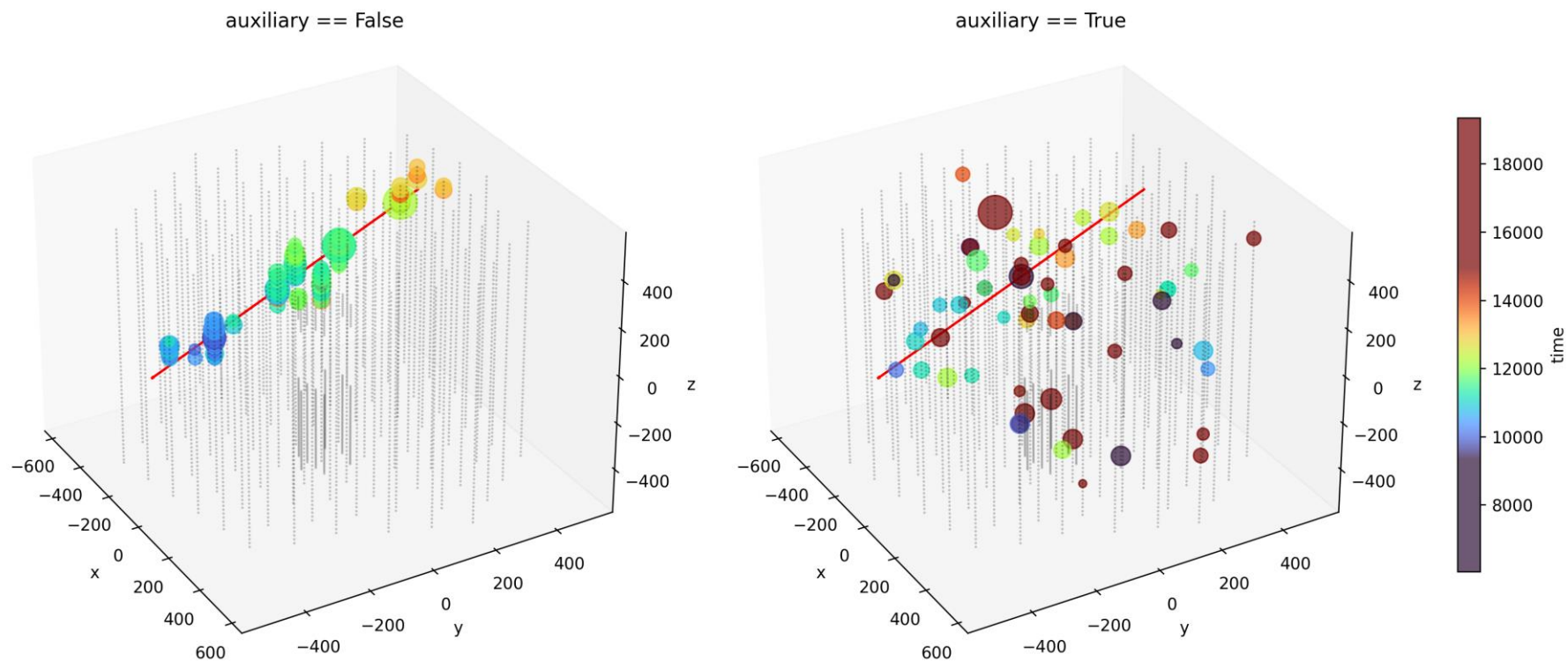
Energy resolution as a function of true energy. The solid line corresponds to fitted parameterized energy resolution. All variants of the proposed approach cut energy estimation uncertainty by around 50%.

Graph neural networks in low-energy physics

[Abbasi, R., et al. "Graph Neural Networks for low-energy event classification & reconstruction in IceCube." *Journal of Instrumentation* 17.11 \(2022\): P11003.](#)

Data

Example event from the dataset:
(azimuth = 4.86 rad, zenith = 1.96 rad)



Graph construction

- Nodes: observed pulses
- Connectivity: 8 nearest neighbors based on the Euclidean distance
- Edge features: None
- Node features:

Feature	Description	Unit
D_{xyz}	Position of DOMs in IceCube coordinates	m
t	Pulse time relative to trigger time	ns
q	Charge of a pulse	P.E.
QE	Quantum efficiency of PMT	-

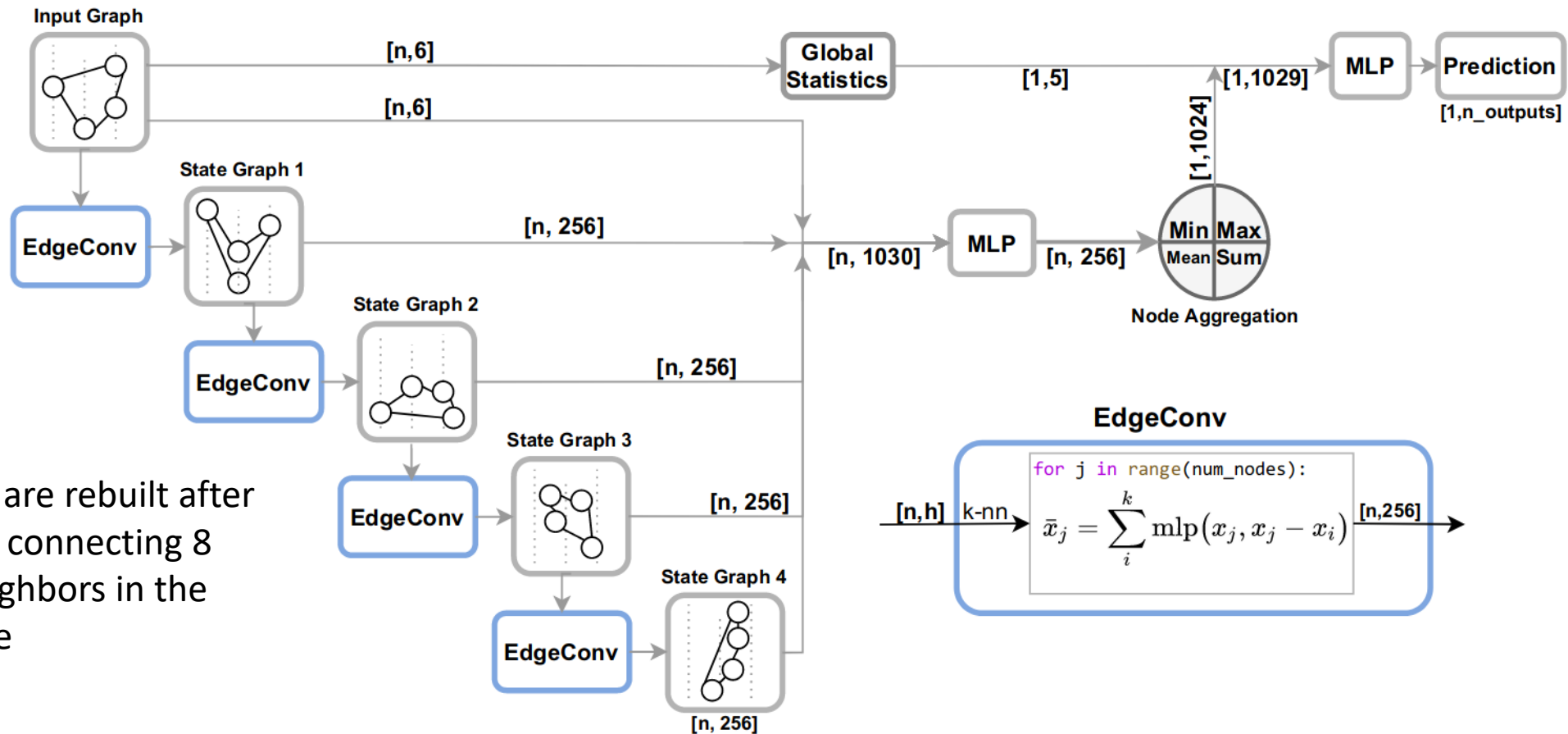
Normalized by $\bar{x} = \frac{x - P_{50th}(x)}{P_{75th}(x) - P_{25th}(x)}$

Recap: EdgeConv

The diagram illustrates the EdgeConv operation formula:
$$\tilde{x}_j = \sum_{i=1}^{N_{\text{neighbors}}} \text{MLP}(x_j, x_j - x_i)$$
 Annotations with blue arrows:

- new node j features** points to \tilde{x}_j .
- concatenation** points to the comma in the MLP function.
- old node j features** points to x_j .
- old node i features** points to x_i in the subtraction term.

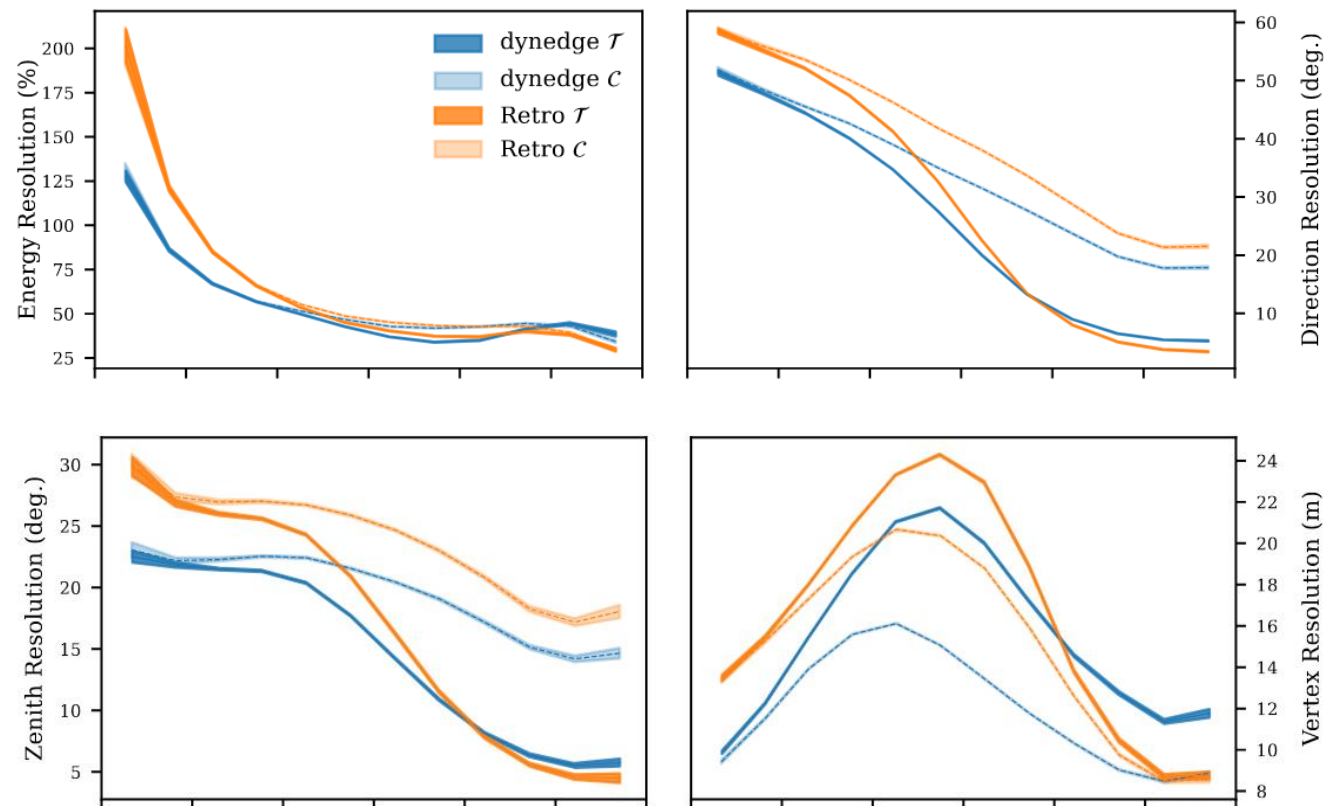
DYNEDGE architecture







The graphs are rebuilt after each stage, connecting 8 nearest neighbors in the latent space

Results

Lower is better



354	Samuel Timothy Chakwera		1.018	2	1mo
355	GraphNeT Baseline		1.018	3	2mo
356	rickmatter		1.018	3	1mo
					

Summary

Graph neural networks

- Offer a natural inductive bias for a lot of physical systems
 - Permutation invariance
 - Locality of interaction
- Depending on construction, can offer more (in/eq)variances
 - Translation, rotation
 - Lorenz transformation
- Variable input size
- Inconvenient to code, but there are libraries

Learn more

- [A good textbook-like description](#)
- [Another good textbook-like description](#)
- [Battaglia, Peter W., et al. "Relational inductive biases, deep learning, and graph networks." *arXiv preprint arXiv:1806.01261* \(2018\).](#) - a thorough and math-heavy derivation and description
- [Zhou, Jie, et al. "Graph neural networks: A review of methods and applications." *AI Open* 1 \(2020\): 57-81.](#)
- [Shlomi, Jonathan, Peter Battaglia, and Jean-Roch Vlimant. "Graph neural networks in particle physics." *Machine Learning: Science and Technology* 2.2 \(2020\): 021001.](#)

Thanks, and looking forward to
your questions!

Backup

HEP

- Graph-level target
 - Jet classification (LHC)
 - Whole event classification (IceCube)
- Node-level target
 - Pileup (noise) identification
 - Calorimeter reconstruction
- Edge-level target
 - Charged-particle tracking
 - Secondary vertex reconstruction