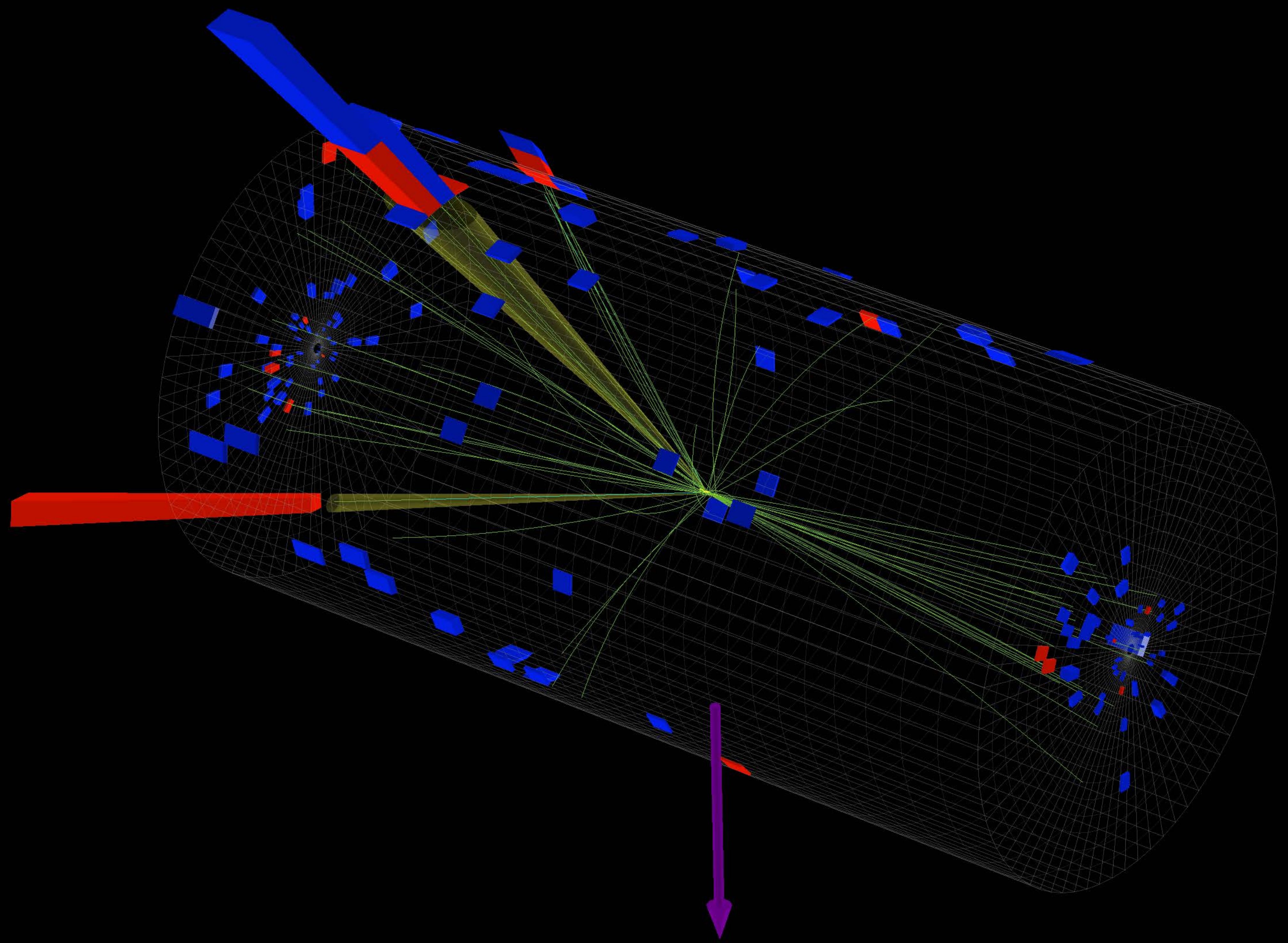
# Graph Neural Networks for Particle Physics

Huilin Qu (CERN)

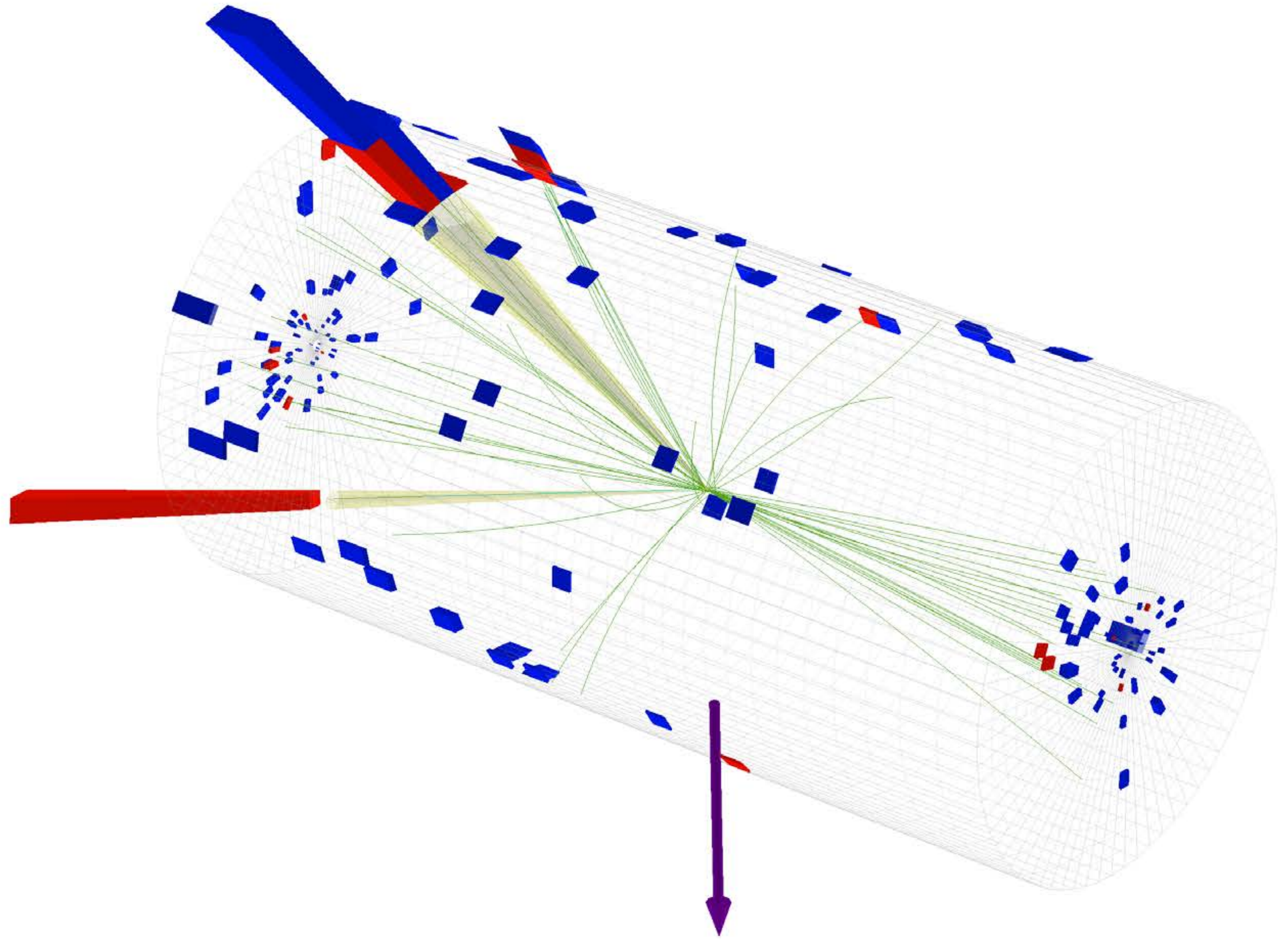
*IAS Program on High Energy Physics (HEP 2023)*

*February 12, 2023*



# Motivation

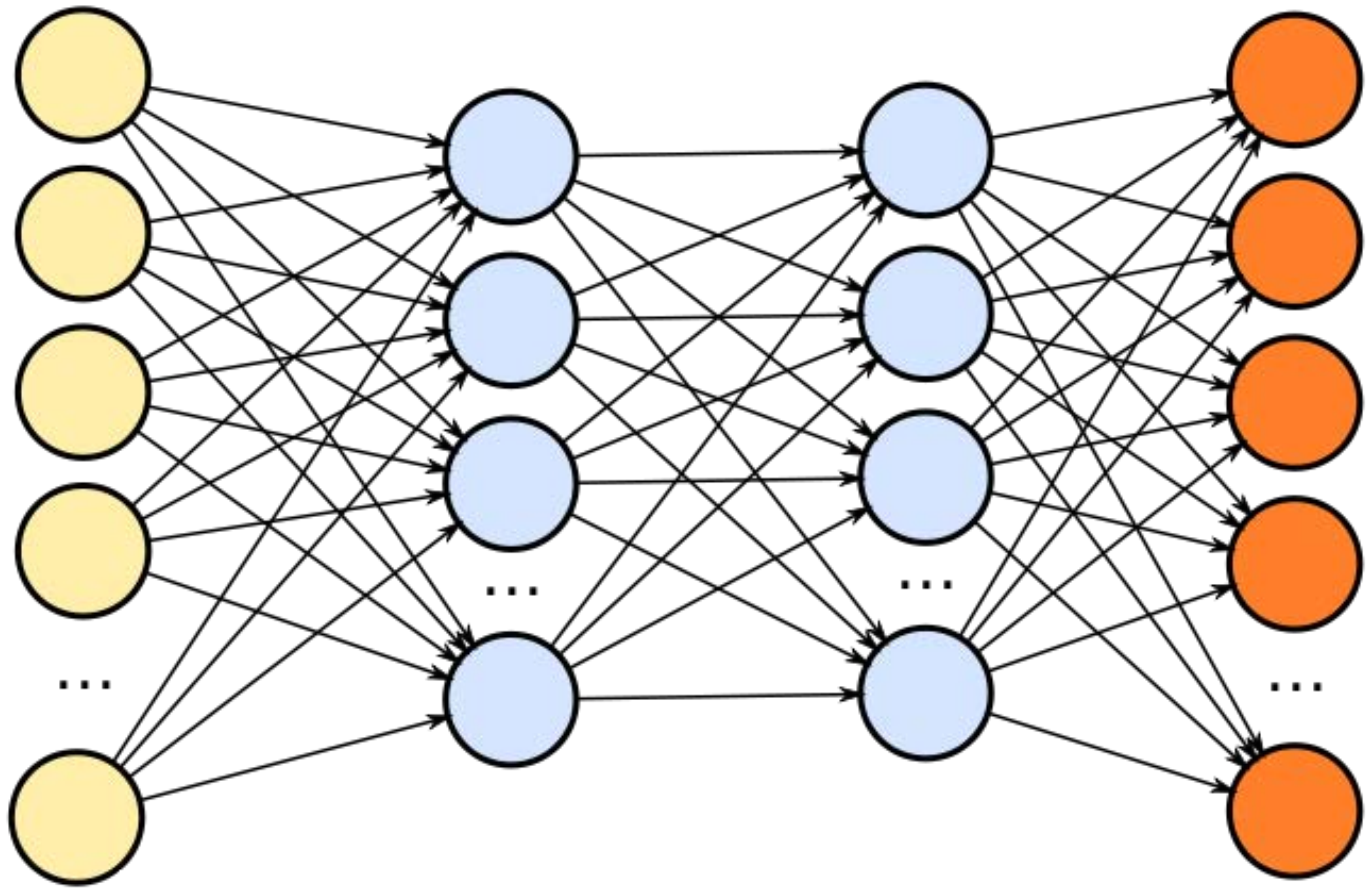
*HEP*



*Collision events, hadronic jets, tracker/calorimeter hits,…*

×

*ML*

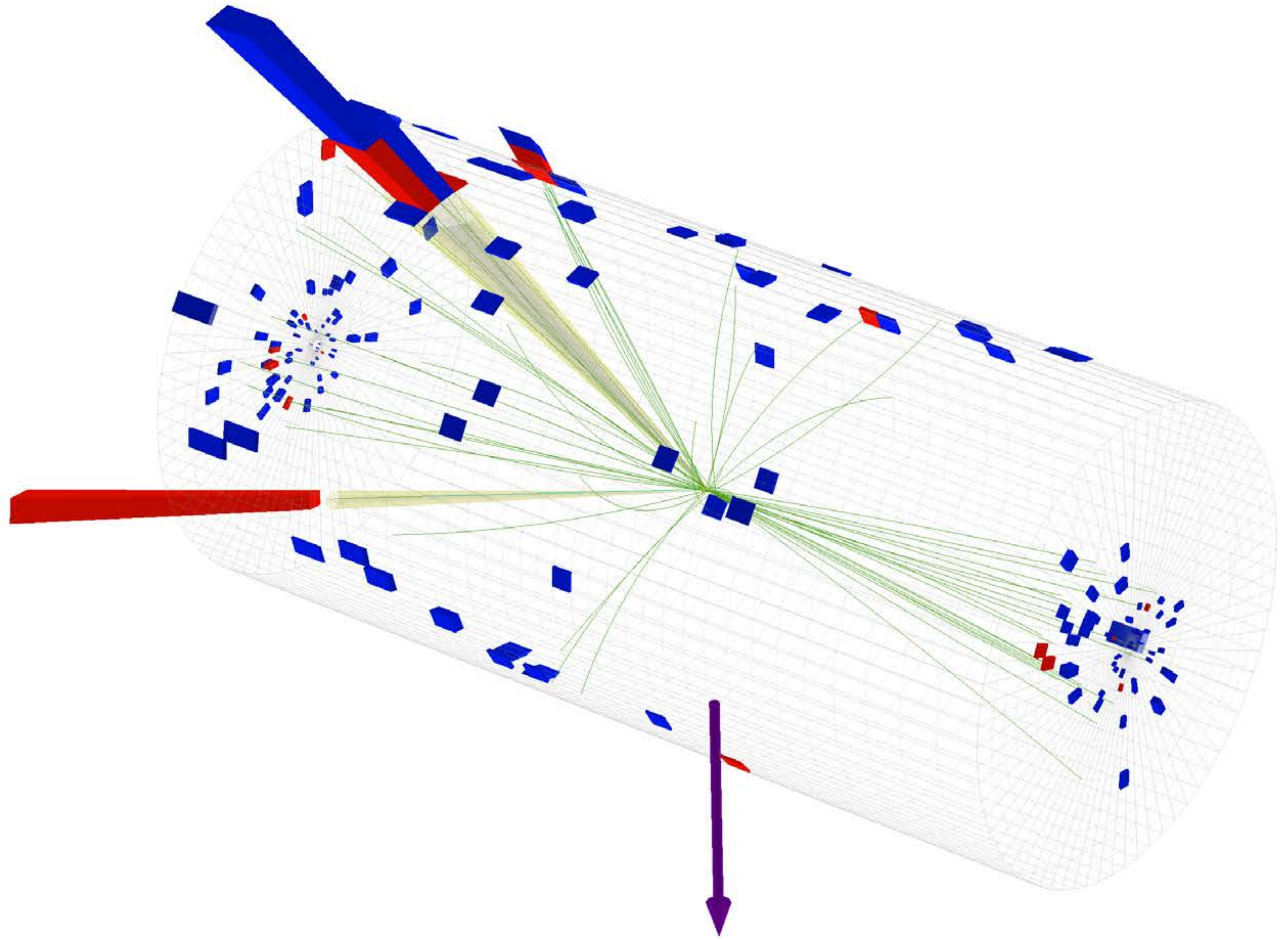



*First and foremost:*

***How to represent the data?***

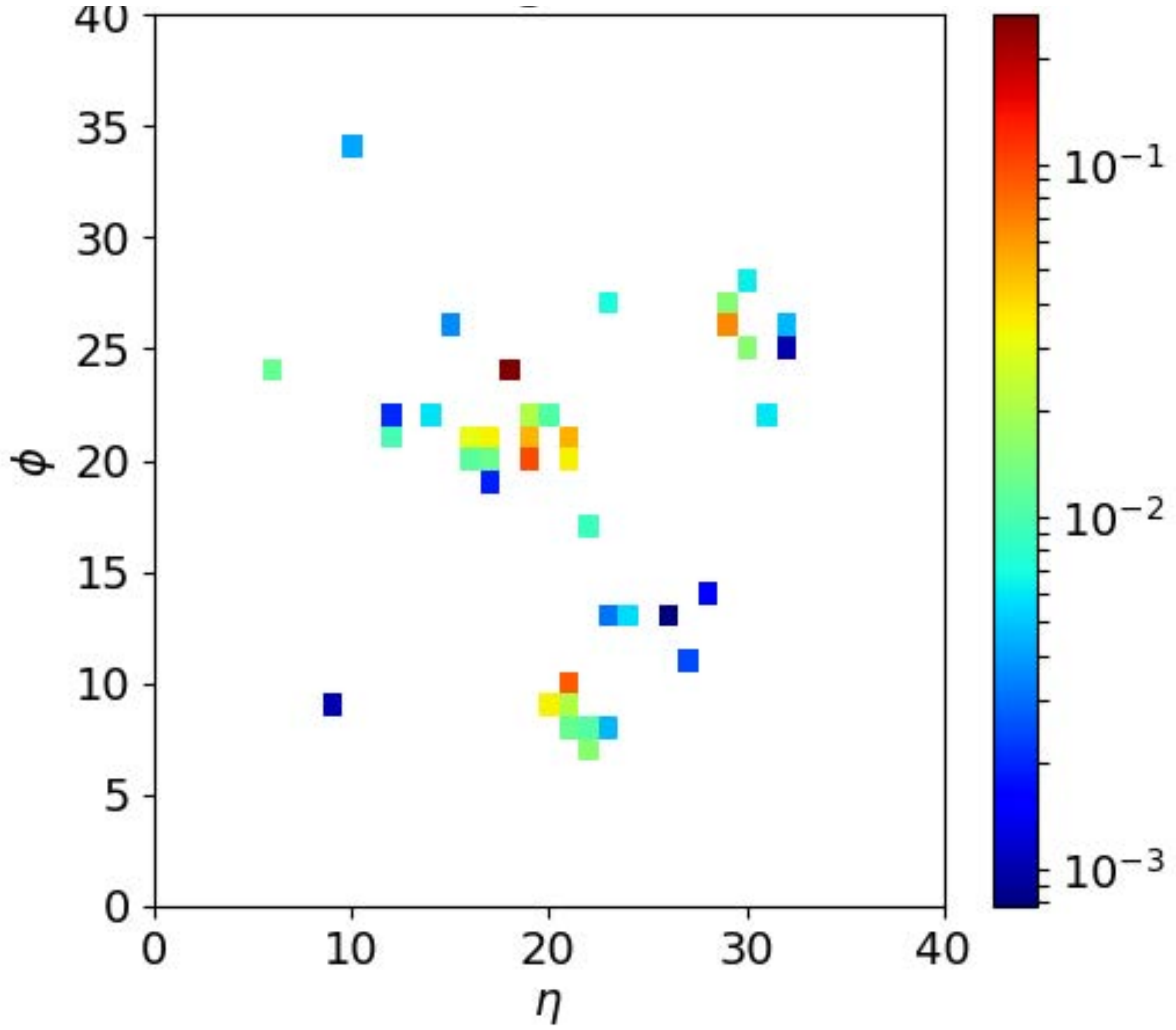
# DATA REPRESENTATION: IMAGE

*HEP*



*Collision events, hadronic jets, tracker/calorimeter hits,…*

*Image*

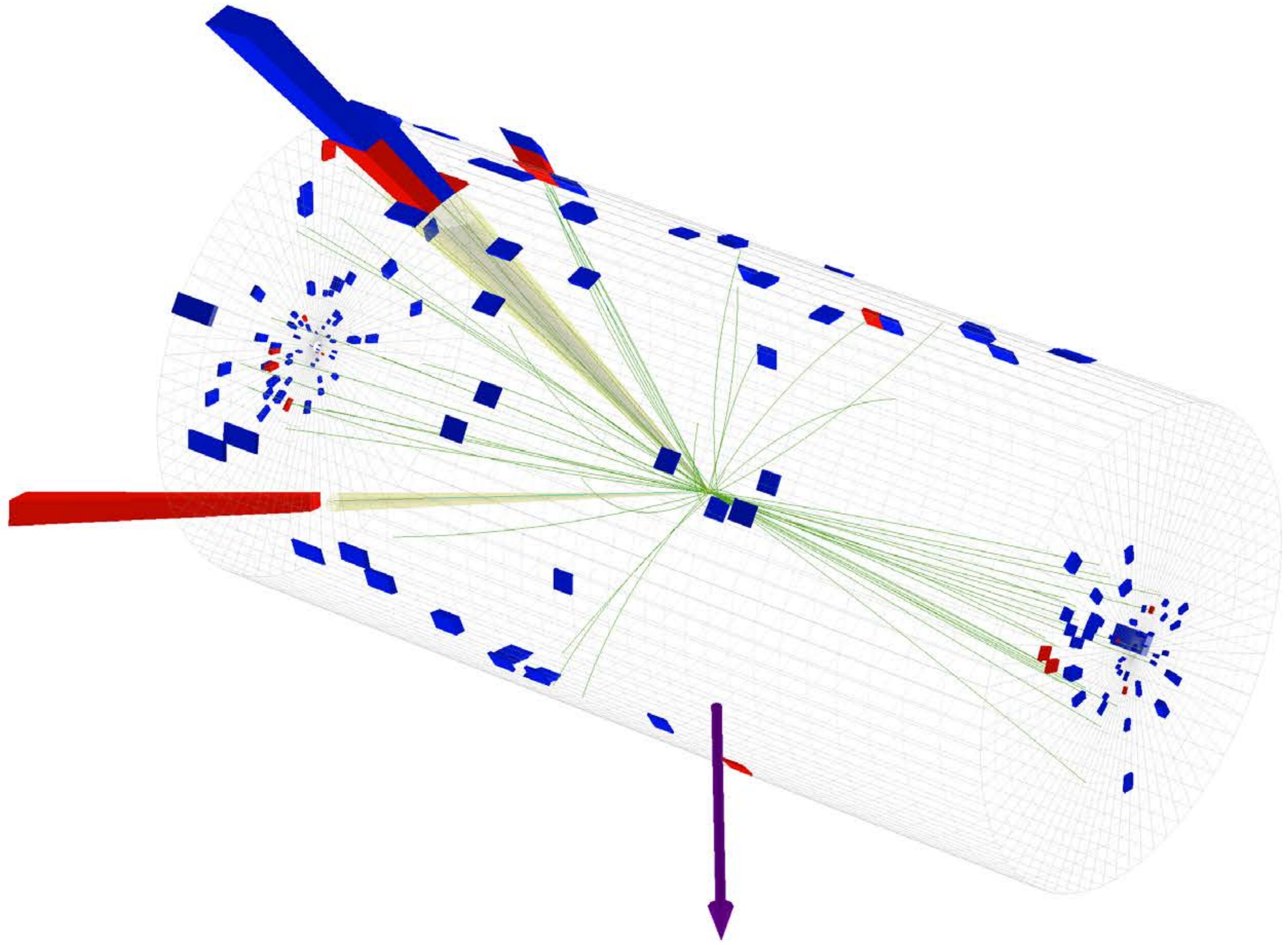



e.g., review in Kagan, arXiv:2012.09719

- Convert to 2D/3D image => **Computer vision**
  - then use convolutional neural networks (CNNs)
  - but:
    - inhomogeneous geometry, high sparsity, …

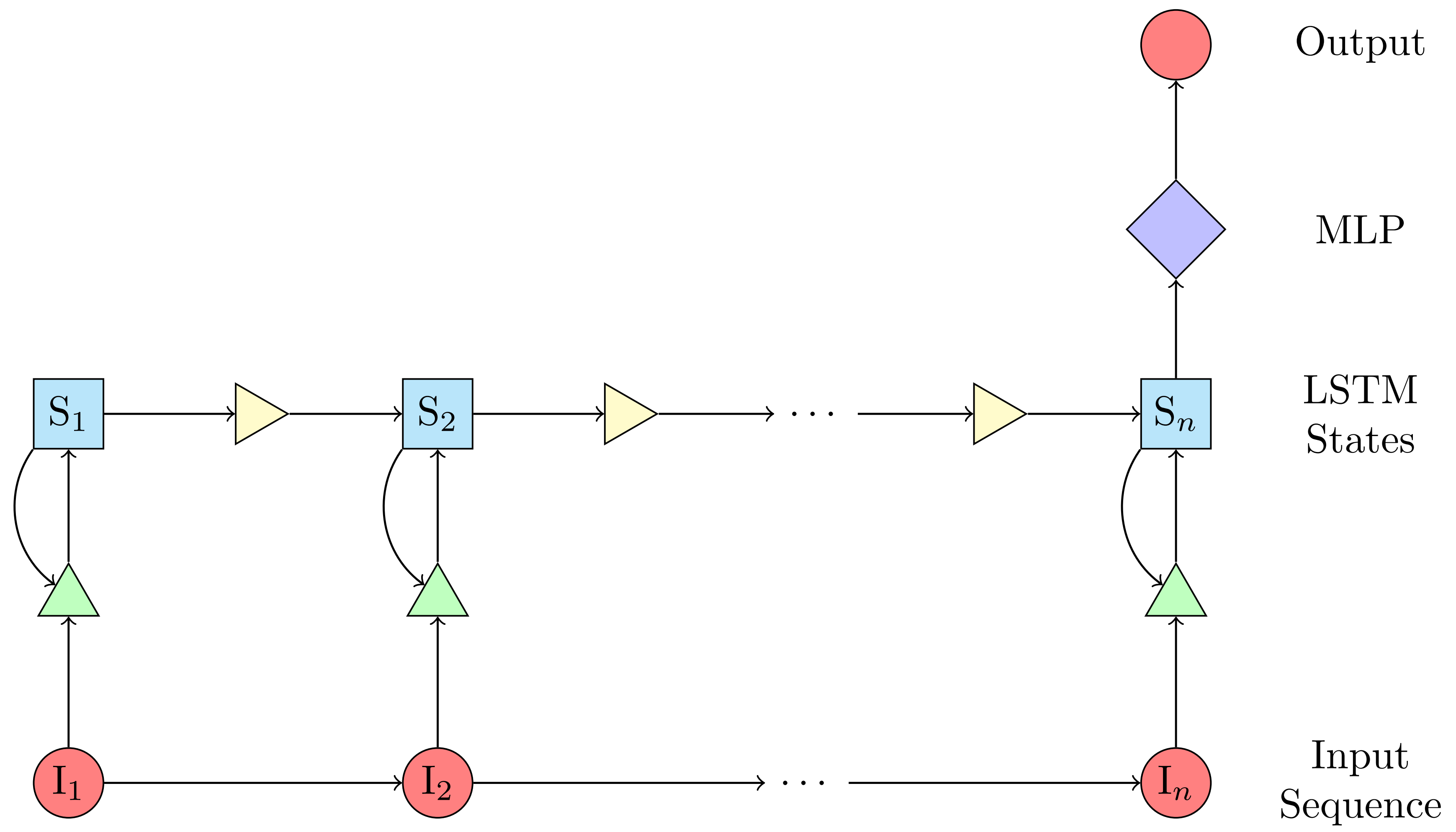
# Data Representation: Sequence

*HEP*



*Collision events, hadronic jets, tracker/calorimeter hits,…*

*Sequence*

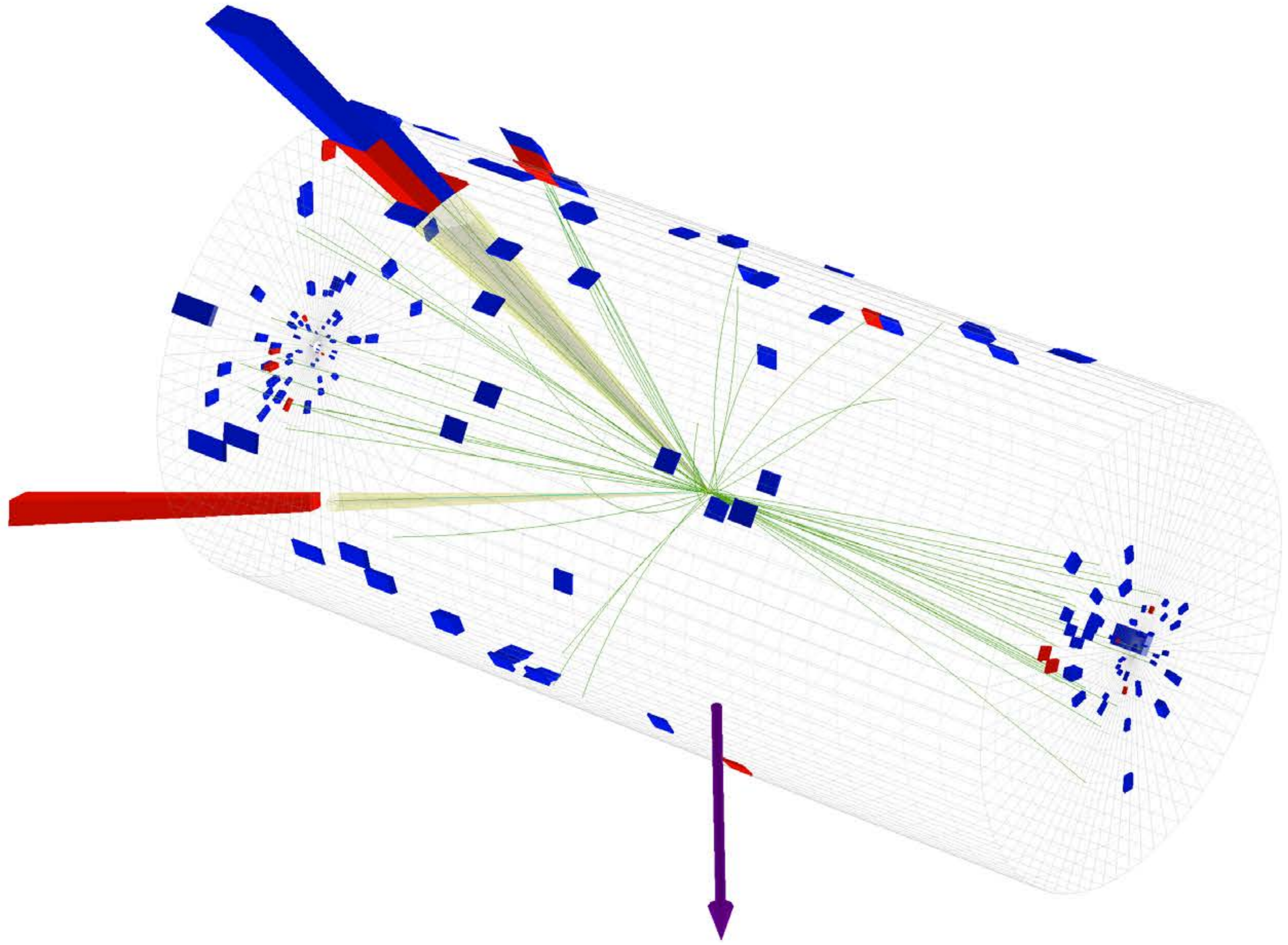



e.g., Guest, Collado, Baldi, Hsu, Urban, Whiteson
arXiv: 1607.08633

- Convert to a sequence => **Natural language processing (NLP)**
  - recurrent neural network (RNN), e.g., GRU/LSTM; 1D CNNs; etc.
  - but:
    - must impose an ***ordering*** on the particles/hits, which can limit the learning performance

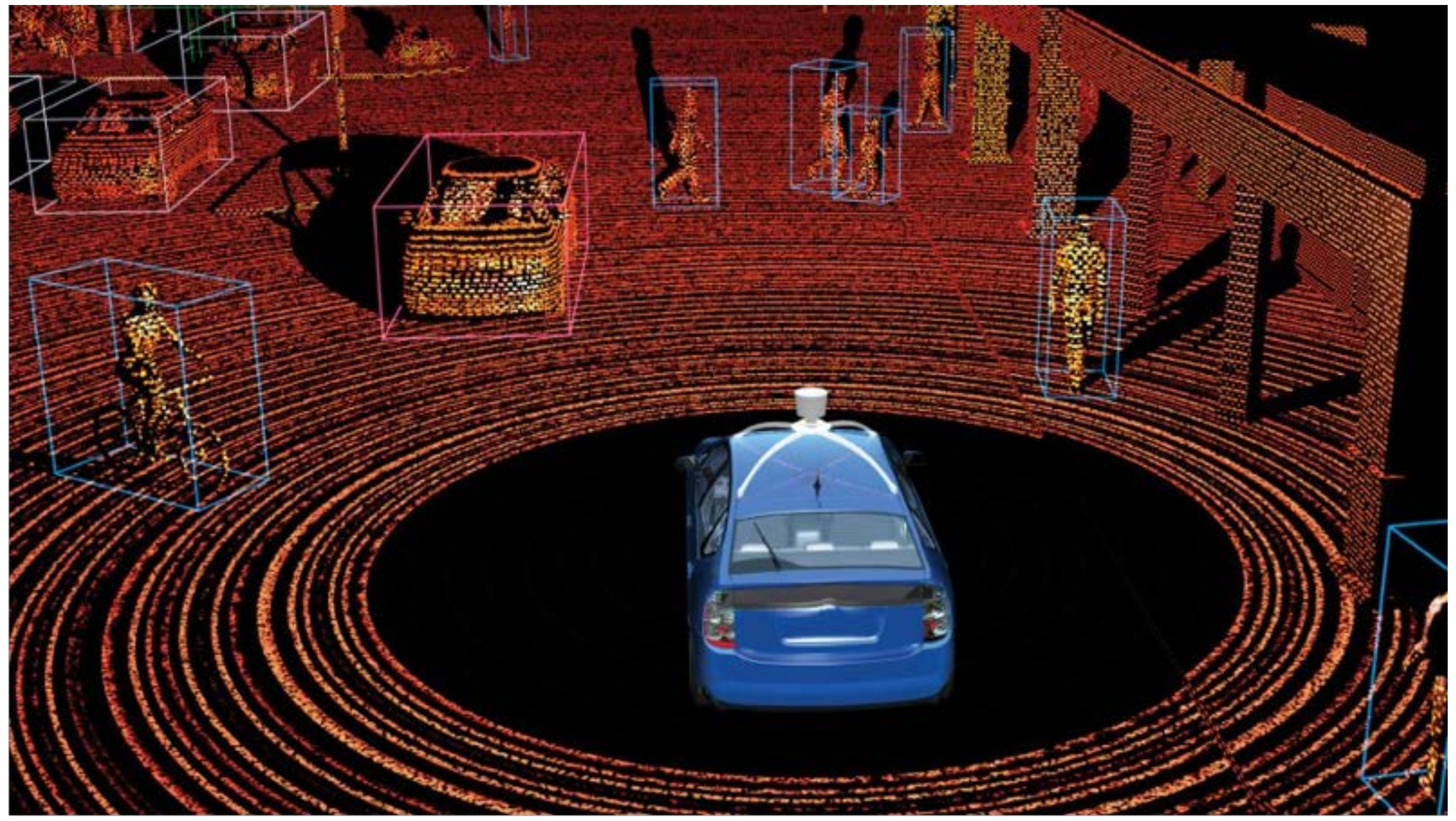
# DATA REPRESENTATION: POINT CLOUD

*HEP*



*Collision events, hadronic jets, tracker/calorimeter hits,…*

*Point cloud*

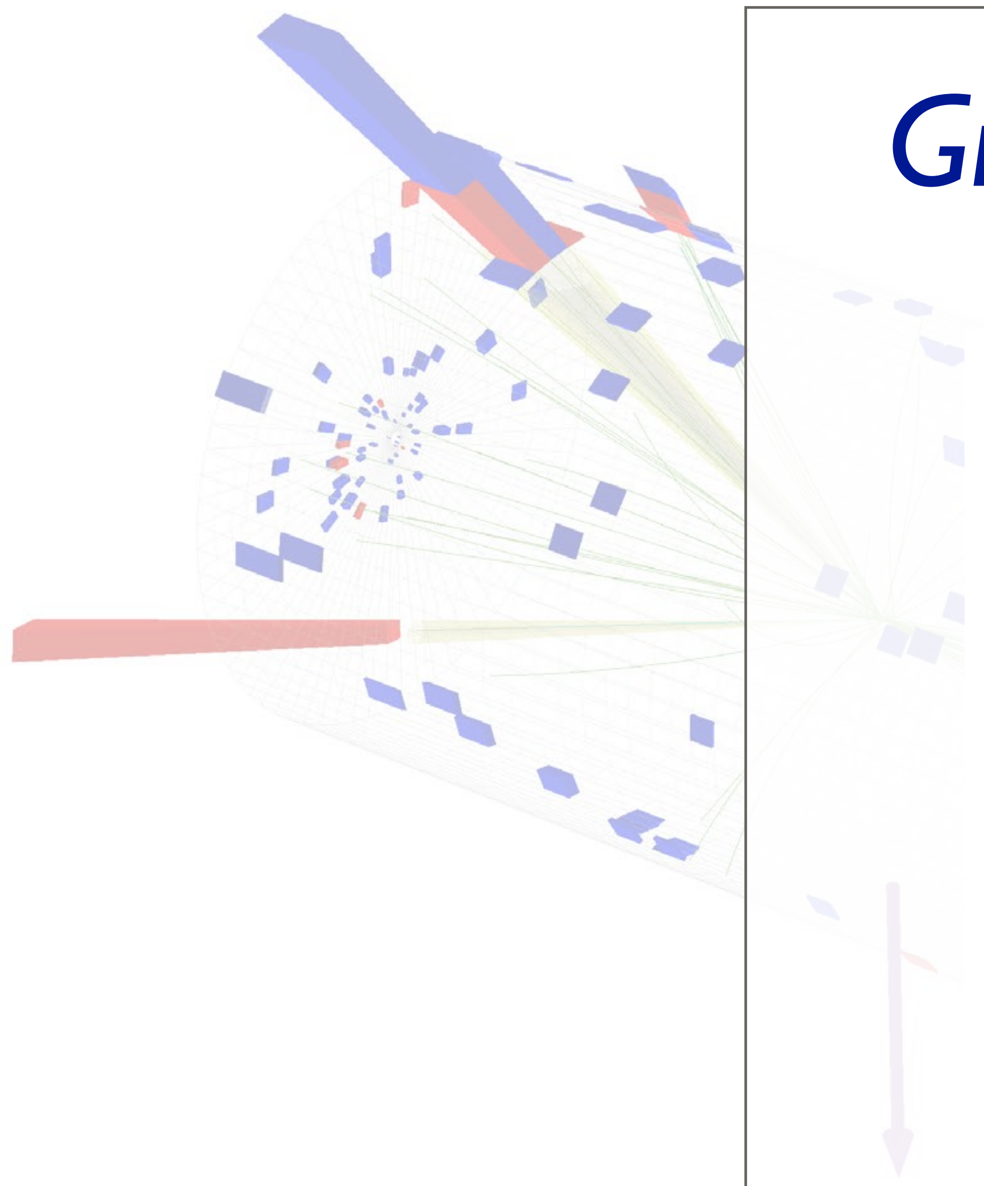


*An unordered set of points in space*
*(e.g., produced by a LiDAR on self-driving cars)*

- HEP data as a point cloud
  - each particle / hit / cell is a point in the cloud
    - for each point: (spatial) coordinates + any additional properties (energy/momentum, detector response, …)
  - key feature: ***permutation invariance***

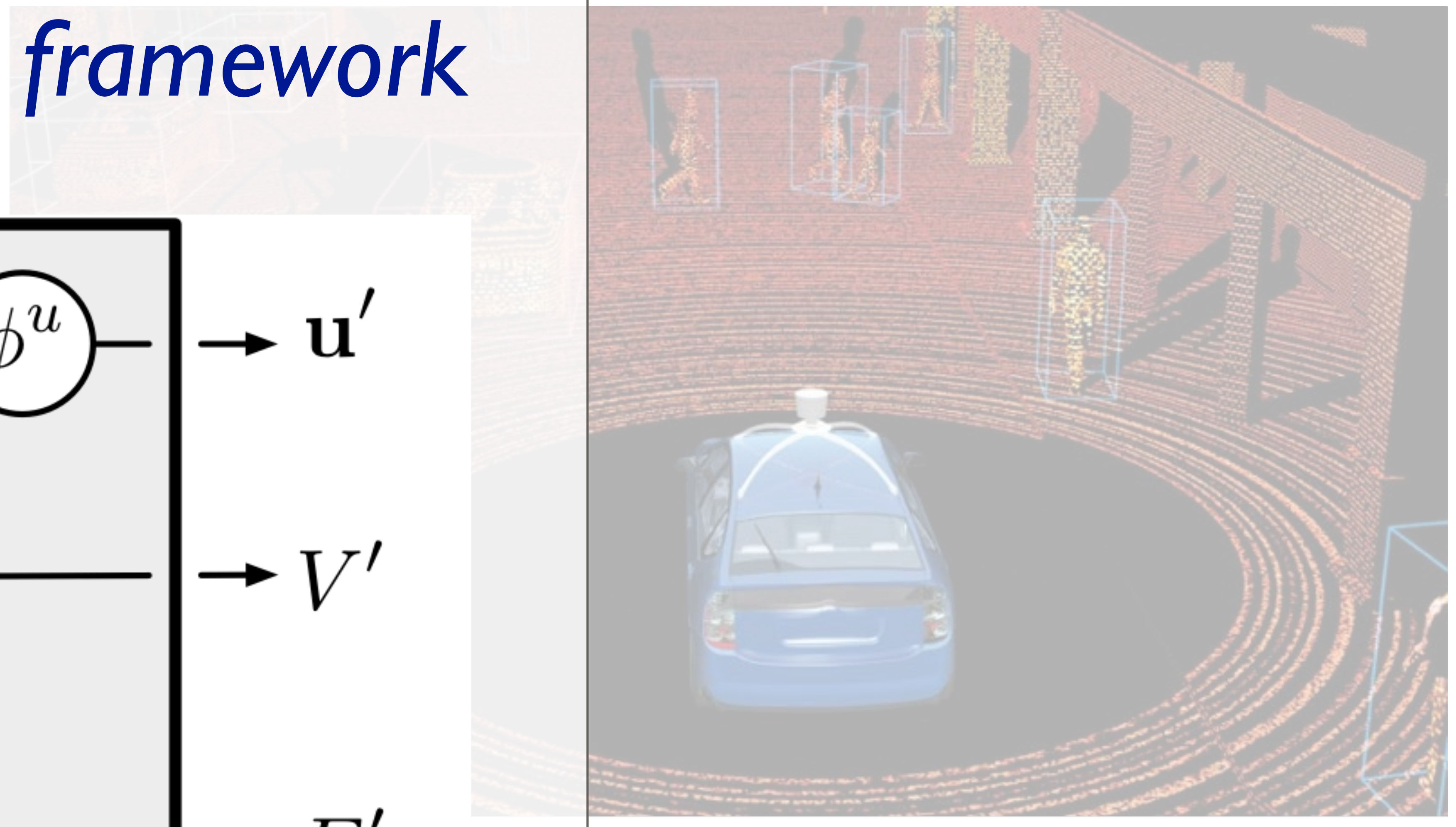
# Learning on Point Clouds

*HEP*



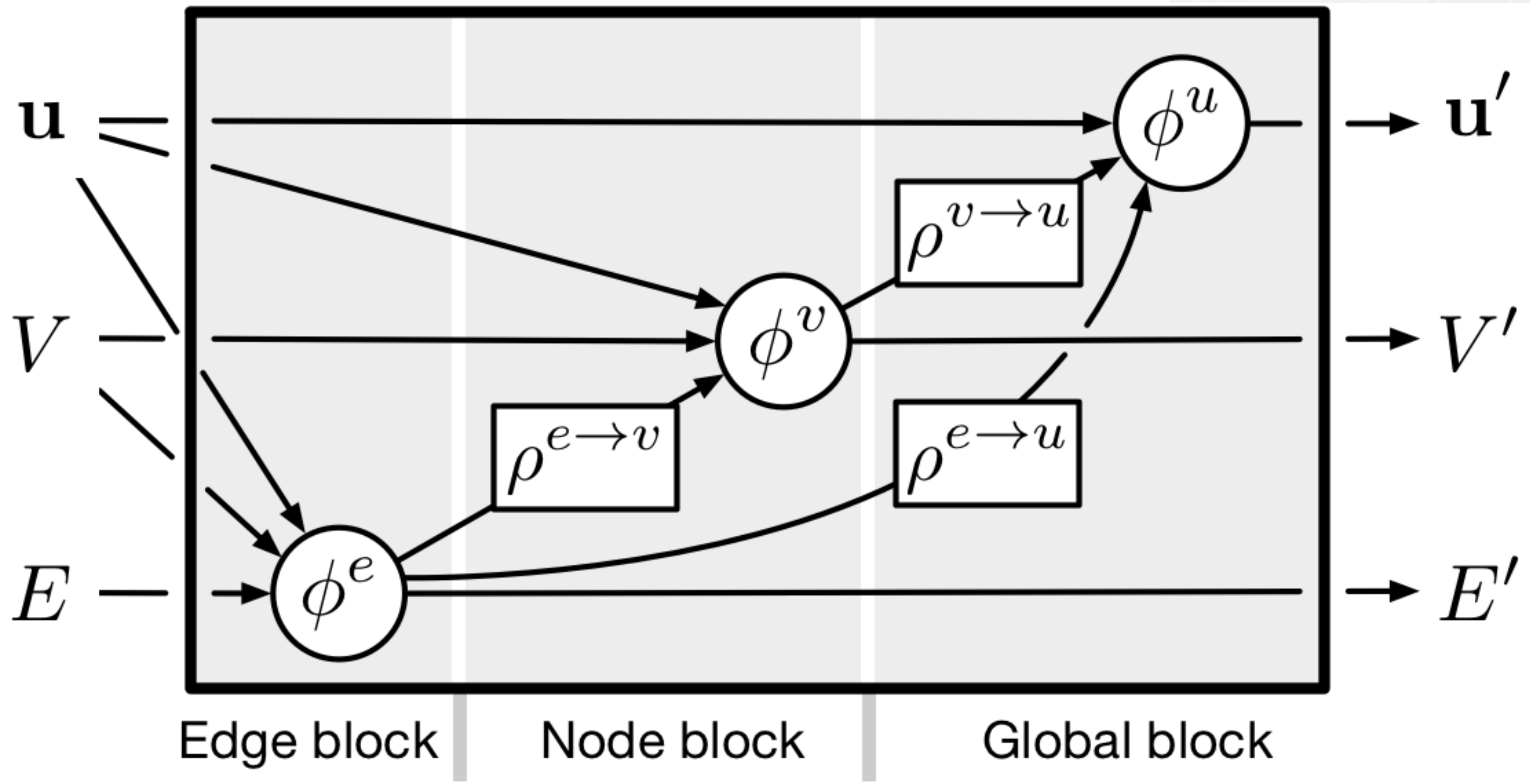
*Collision events, hadronic jets, tr*

*Point cloud*



*dered set of points in space*
*d by a LiDAR on self-driving cars)*

## *Graph neural network - A unified framework*



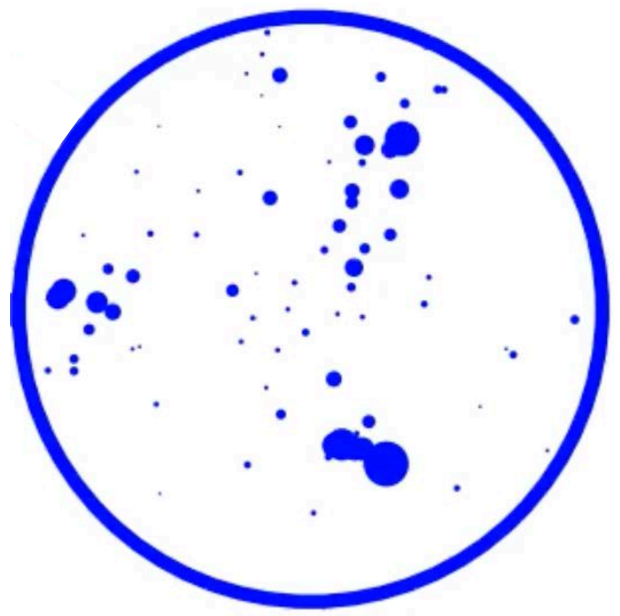

Review in Shlomi, Battaglia, Vlimant, arXiv:2007.13681

- HEP data as a point cloud
  - each particle / hit / cell is a point in the cloud
    - for each point: (spatial) coordinates + any additional properties (energy/momentum, detector response, ...)
  - key feature: ***permutation invariance***

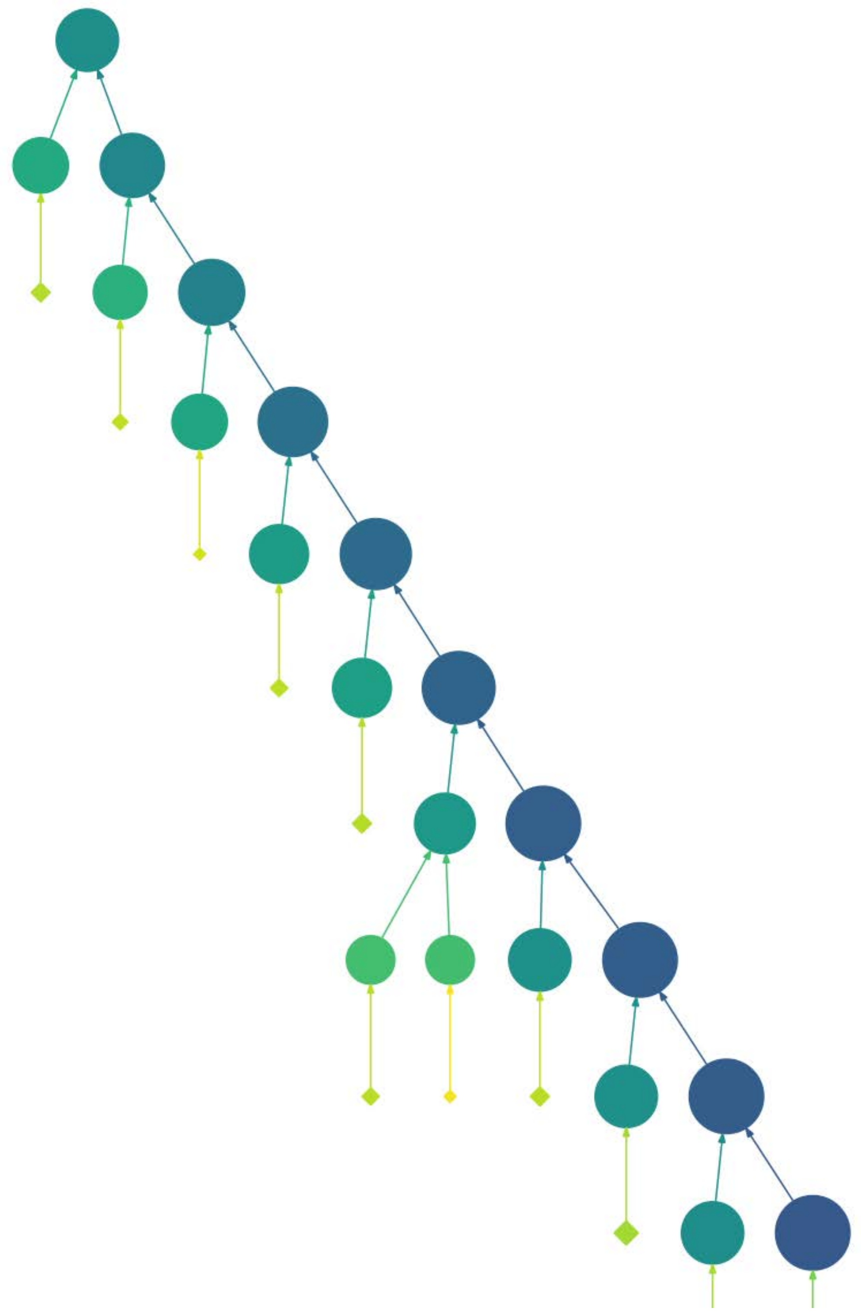
# Constructing the Graph

- From point clouds to graphs:
  - points (particles/hits/cells) naturally become the **nodes** of the graph
  - but how to define the ***edges***?

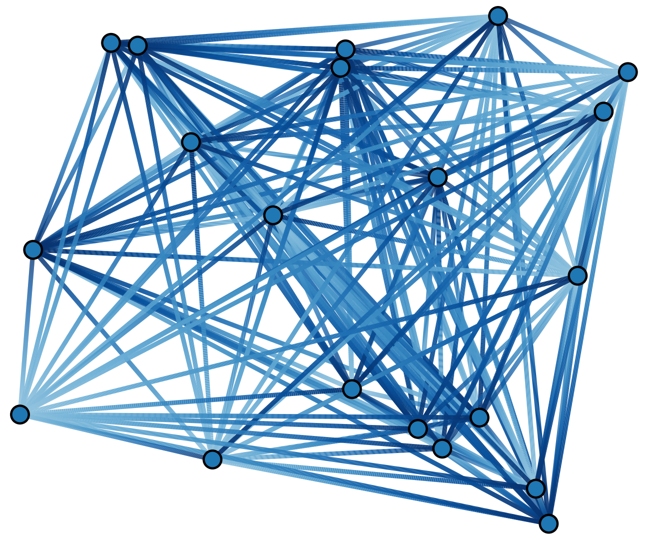
*Set: no edges*



*Hierarchical trees:*
- *decay chain*
- *jet clustering history*

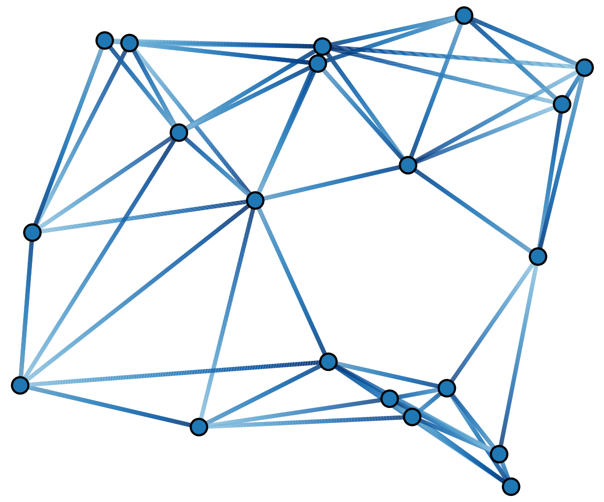


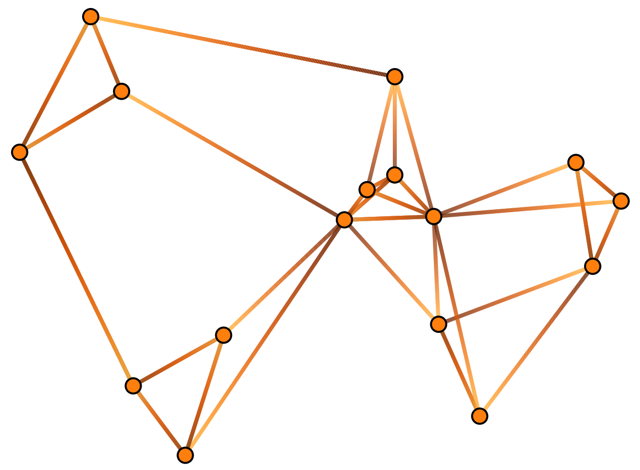
*Fully connected graph*
- *i.e., connect each node to all other nodes*



*Locally connected graph*
- *i.e., connect each node only to neighbor nodes*
  - *k-nearest neighbors*
  - *fixed radius*

*static*





*(dynamically) learned*

# Constructing the Graph

- From point clouds to graphs:
  - points (particles/hits/cells) naturally become the **nodes** of the graph
  - but how to define the ***edges***?

- Why we need the edges?
  - edges <==> interactions
  - edges control **information flows** in the graph
  - input edge features can encode **inter-relationship** between nodes and can incorporate **physics motivated variables** (e.g., ΔR between particles, invariant mass of the particle pair, etc.)
  - latent edge features store **learned relational information** — crucial for the ML task

# Graph Network Formalism

- Typical GNN architectures can be described in the "Message Passing" framework

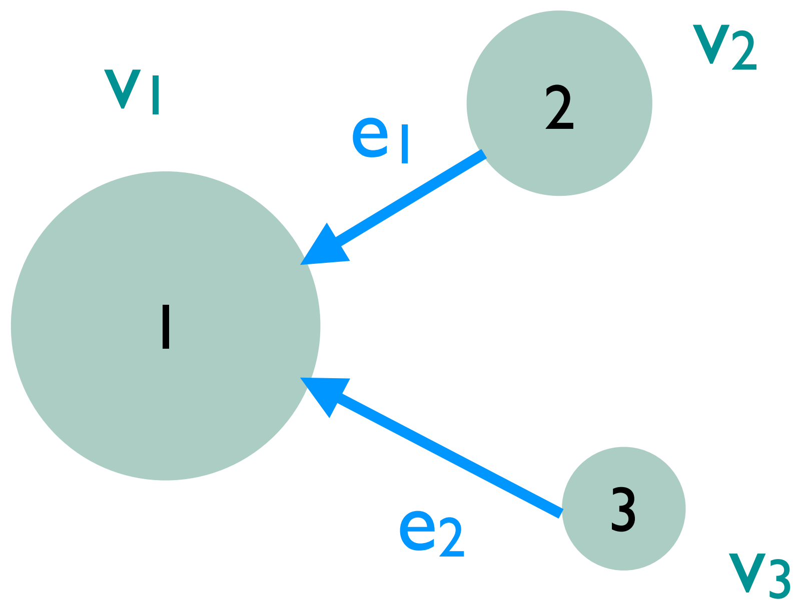
***Graph:*** $G = (\mathbf{u}, V, E)$ with $N_v$ vertices and $N_e$ edges

*Vertices (nodes)*

$$V = \{\mathbf{v}_i\}_{i=1:N_v}$$

*Edges (links)*

$$E = \{(\mathbf{e}_k, r_k, s_k)\}_{k=1:N_e}$$




$e'_k$: message computed for edge $k$ connecting nodes $r_k$, $s_k$

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

Shlomi, Battaglia, Vlimant, arXiv:2007.13681


Edge block

# Graph Network Formalism

- Typical GNN architectures can be described in the "Message Passing" framework

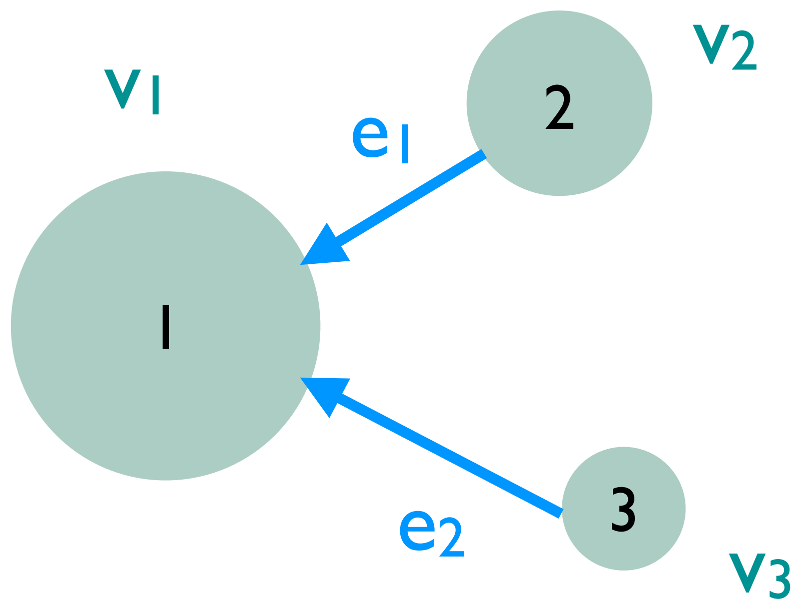
*Graph:* $G = (\mathbf{u}, V, E)$ with $N_v$ vertices and $N_e$ edges

*Vertices (nodes)*

$$V = \{\mathbf{v}_i\}_{i=1:N_v}$$

*Edges (links)*

$$E = \{(\mathbf{e}_k, r_k, s_k)\}_{k=1:N_e}$$



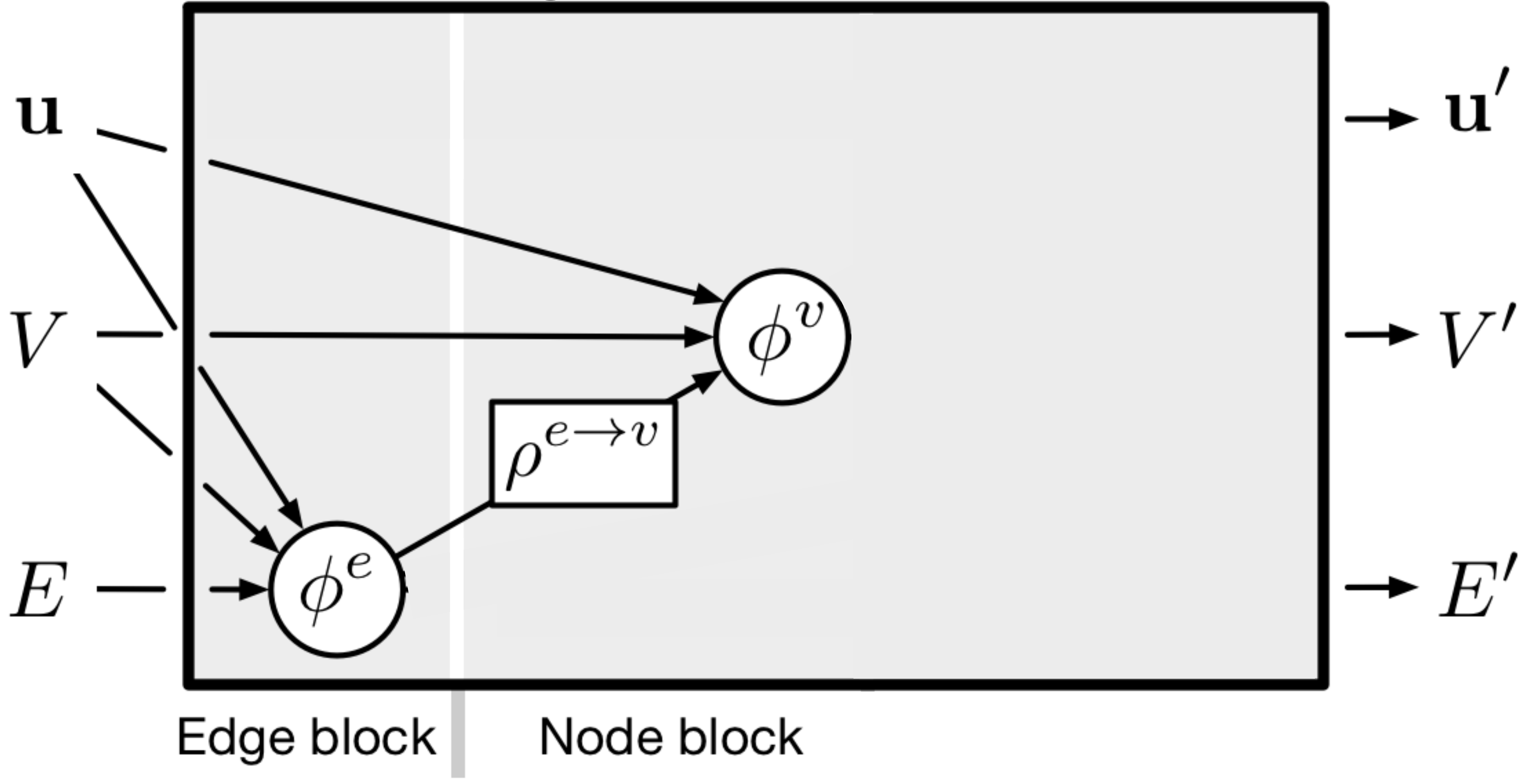

$e'_k$: message computed for edge $k$ connecting nodes $r_k$, $s_k$

$v'_i$: node feature update based on aggregated messages and previous features

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

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

Shlomi, Battaglia, Vlimant, arXiv:2007.13681



# Graph Network Formalism

- Typical GNN architectures can be described in the "Message Passing" framework

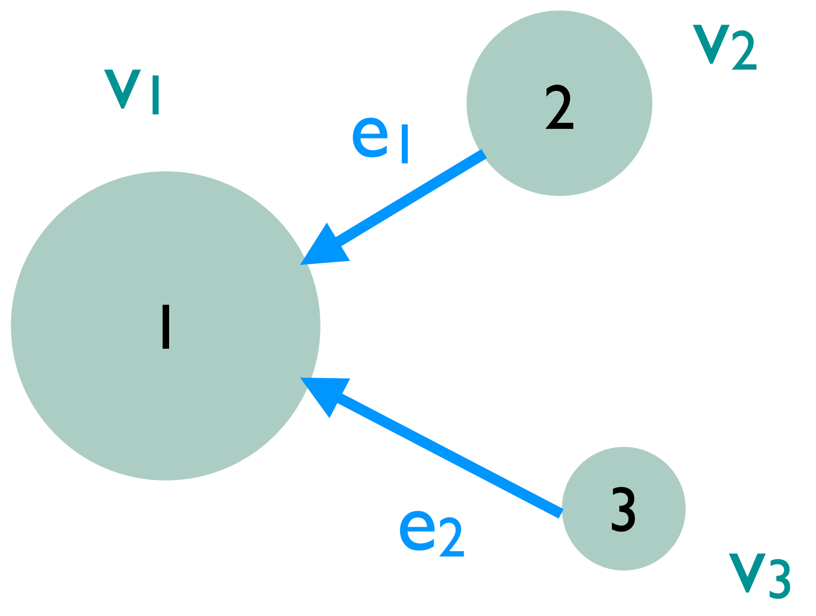
**Graph:** $G = (\mathbf{u}, V, E)$ with $N_v$ vertices and $N_e$ edges

*Vertices (nodes)*

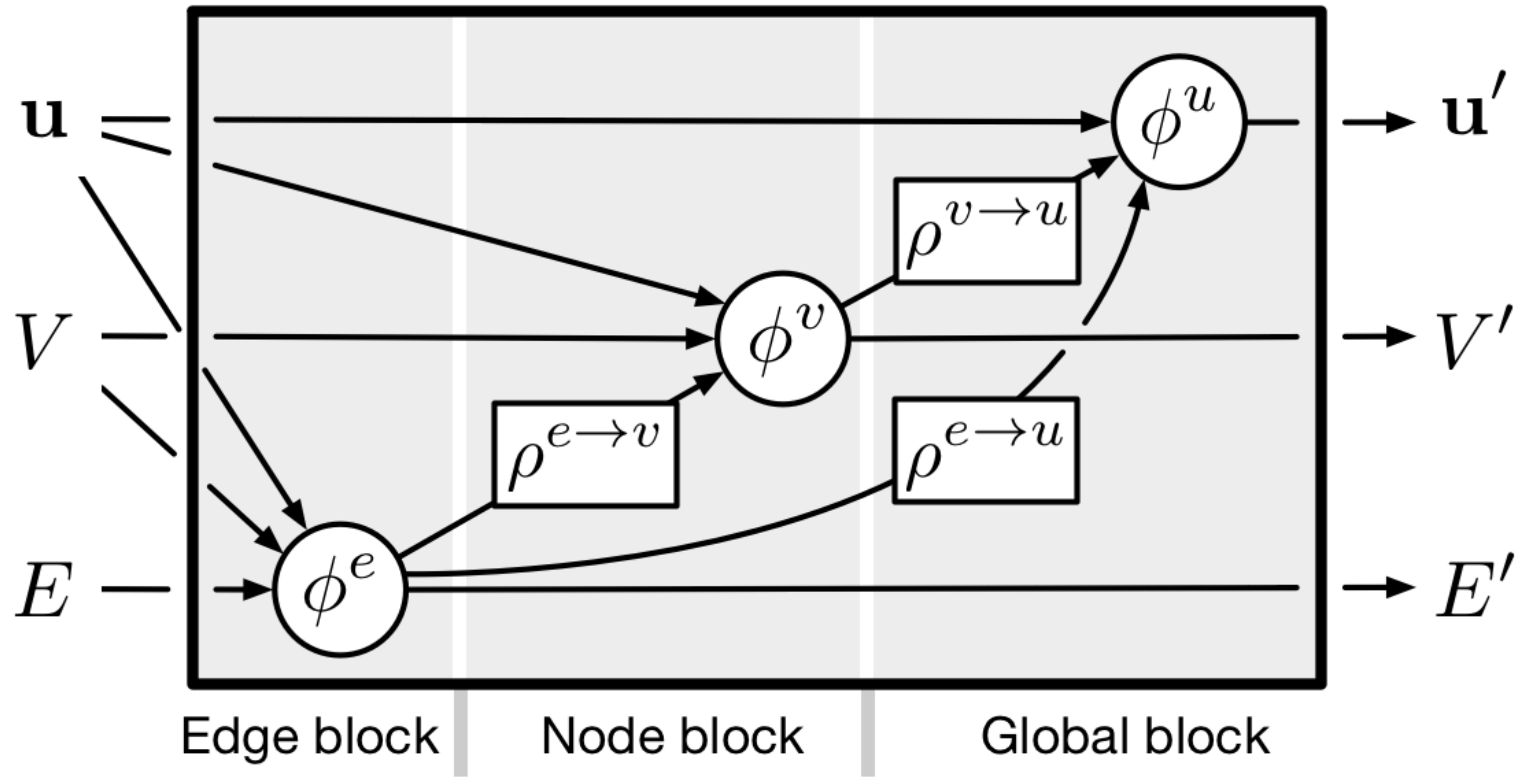
$$V = \{\mathbf{v}_i\}_{i=1:N_v}$$

*Edges (links)*

$$E = \{(\mathbf{e}_k, r_k, s_k)\}_{k=1:N_e}$$




Shlomi, Battaglia, Vlimant, arXiv:2007.13681




$e'_k$: message computed for edge $k$ connecting nodes $r_k$, $s_k$

$v'_i$: node feature update based on aggregated messages and previous features

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

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

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

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

# Graph Network Formalism

- Typical GNN architectures can be described in the "Message Passing" framework

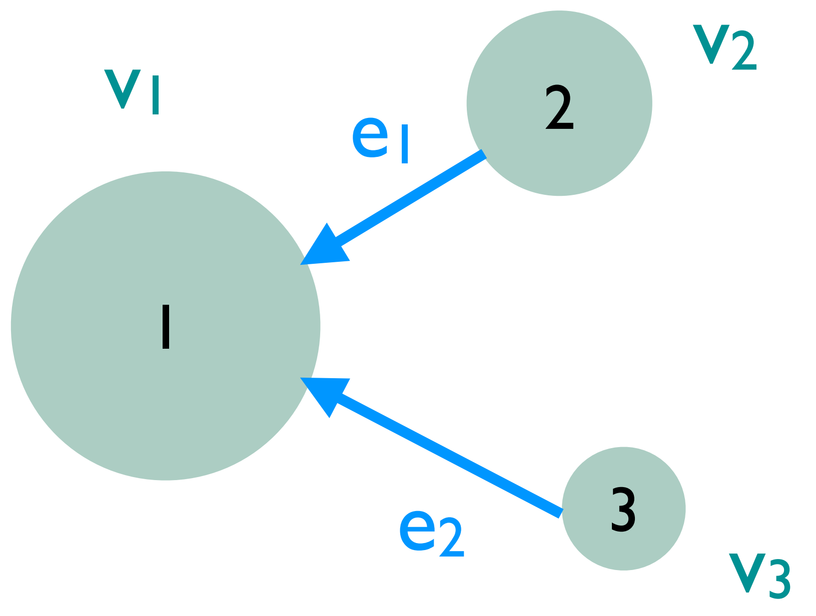
**Graph:** $G = (\mathbf{u}, V, E)$ with $N_v$ vertices and $N_e$ edges

*Vertices (nodes)*

$$V = \{\mathbf{v}_i\}_{i=1:N_v}$$

*Edges (links)*

$$E = \{(\mathbf{e}_k, r_k, s_k)\}_{k=1:N_e}$$




$e_k'$: message computed for edge $k$ connecting nodes $r_k$, $s_k$

$v_i'$: node feature update based on aggregated messages and previous features

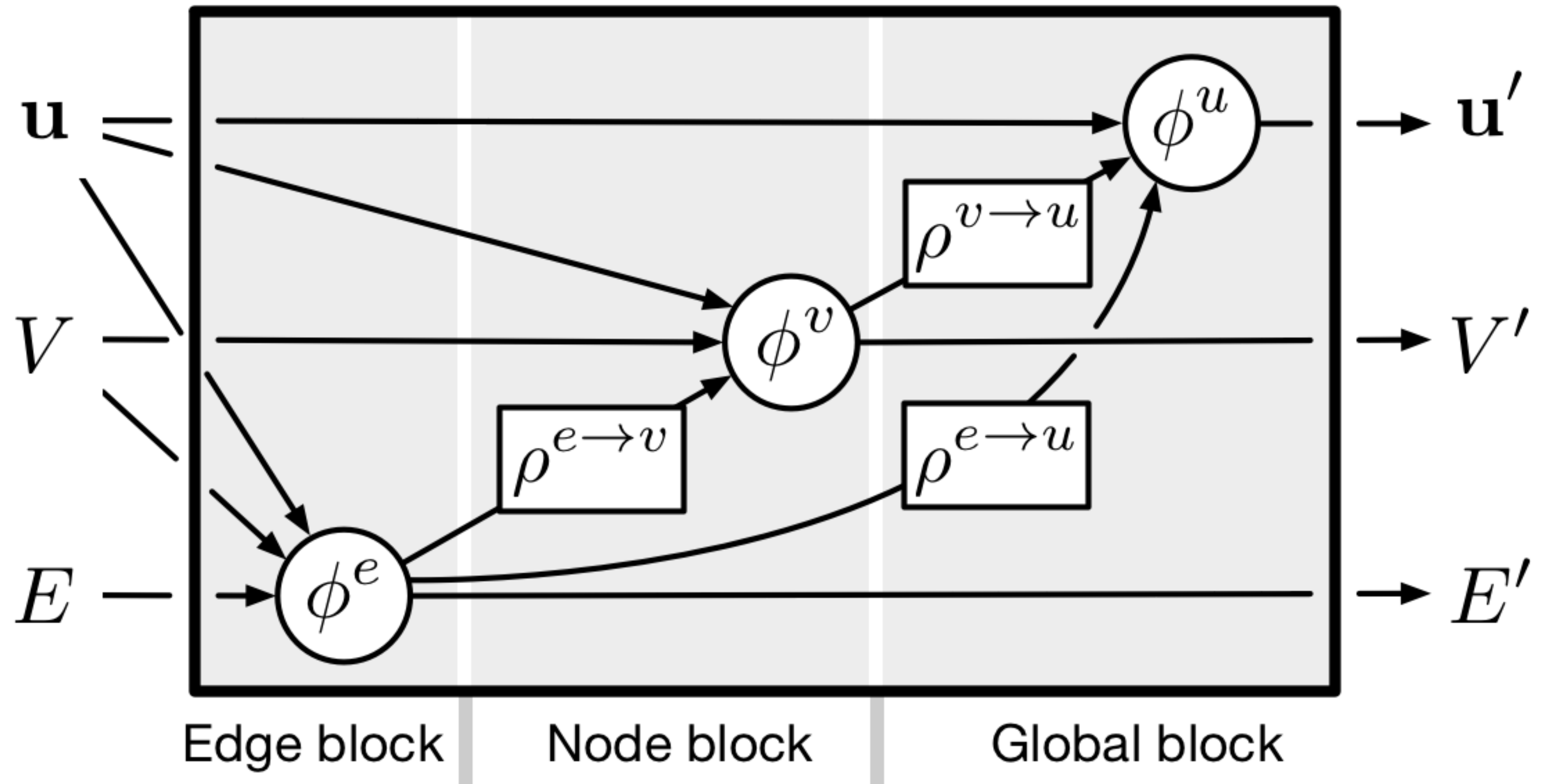
$u'$: global feature update based on aggregated, updated node and edge features

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

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

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

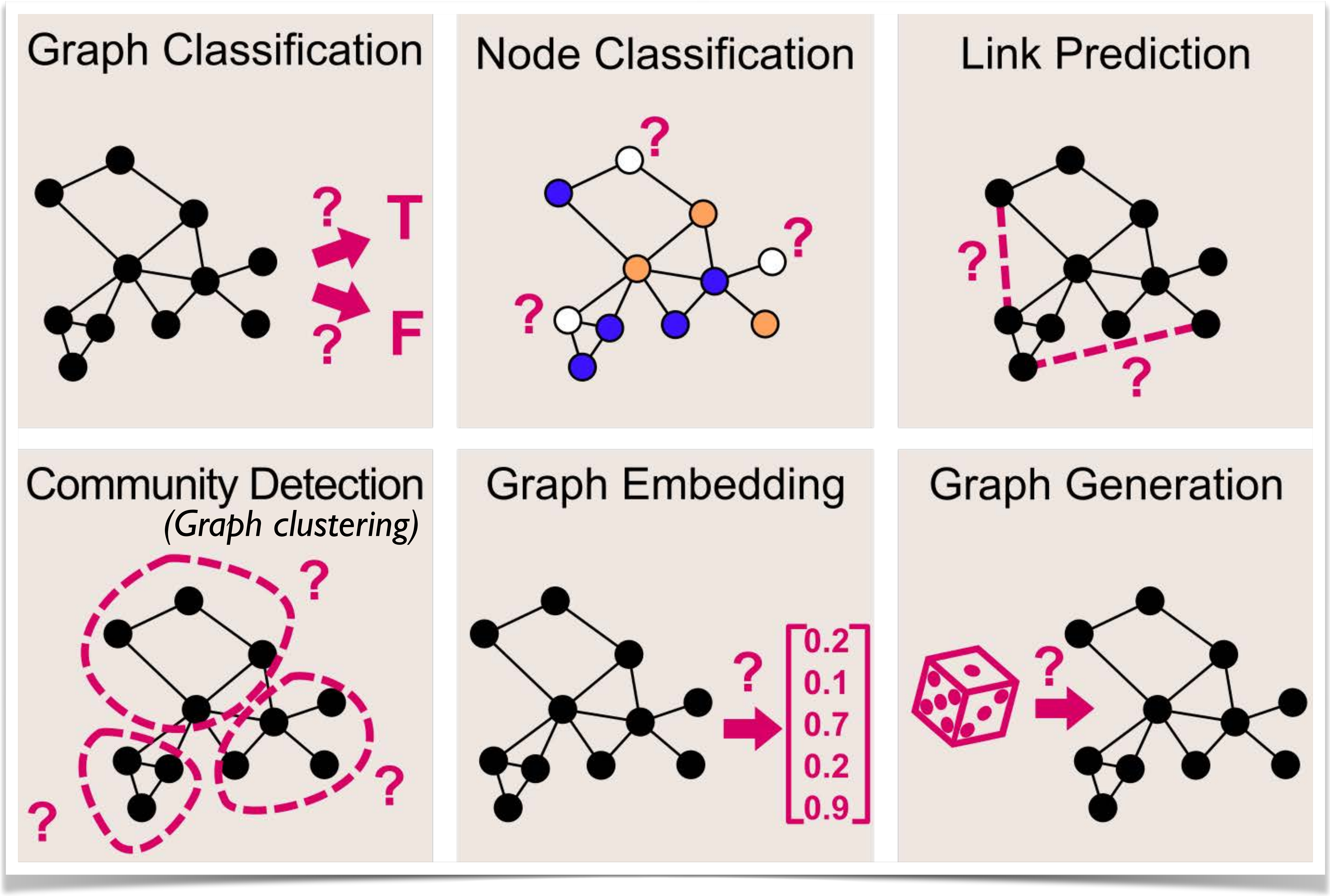
Shlomi, Battaglia, Vlimant, arXiv:2007.13681




*Shared-weight NN* *Symmetric functions (e.g., sum, mean, max, etc.)* → *Permutation invariance*

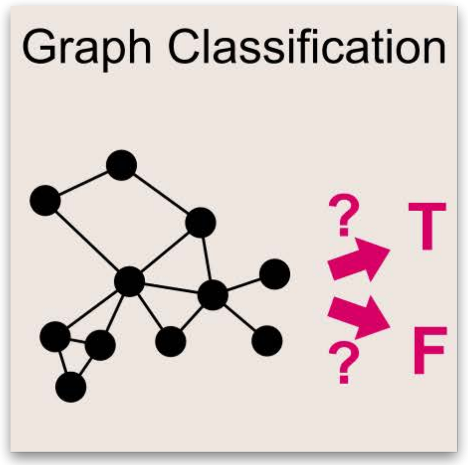
# Graph ML Tasks






https://towardsdatascience.com/graph-convolutional-networks-deep-99d7fee5706f

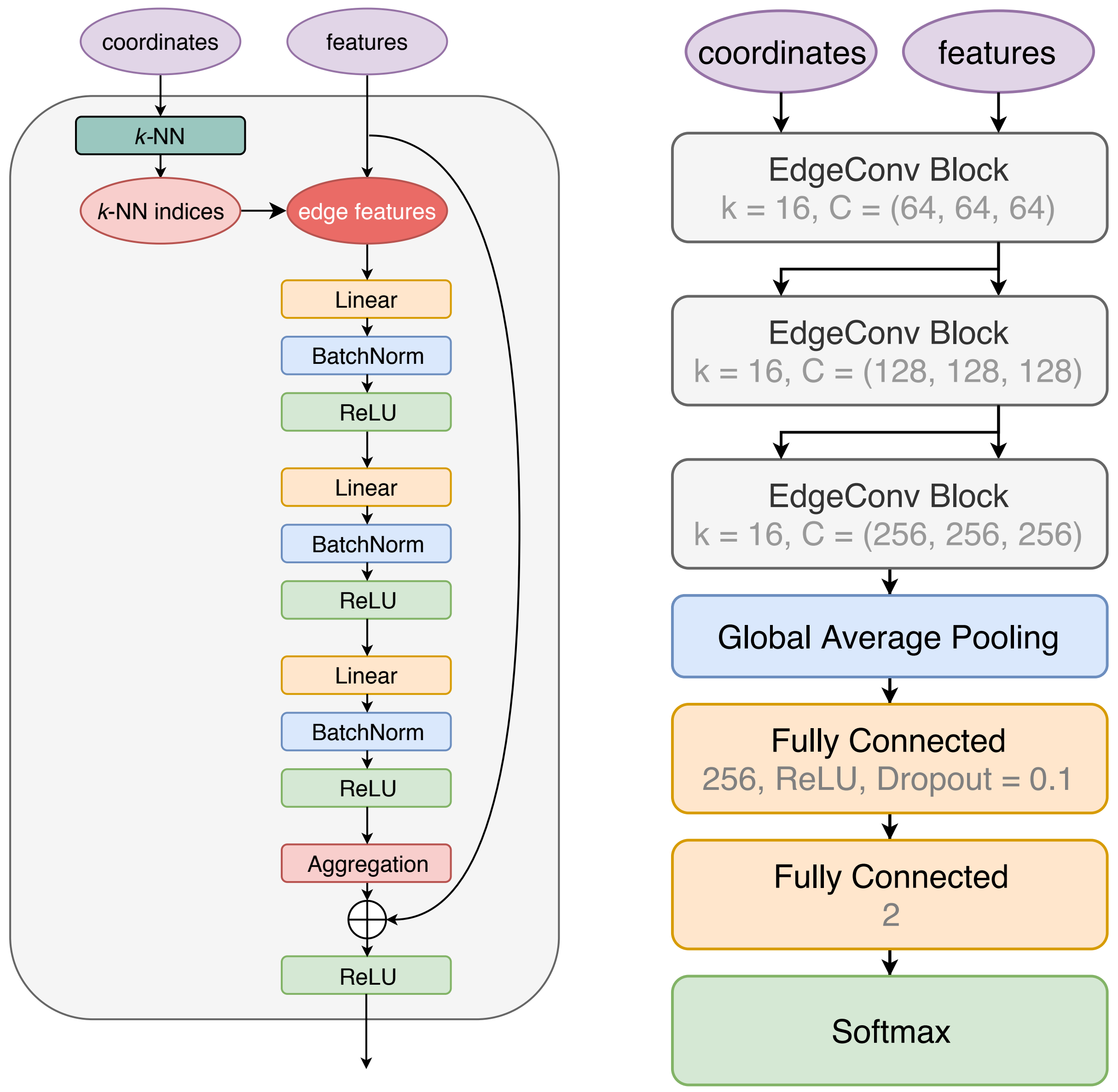
# GNNs for Jet Tagging: ParticleNet

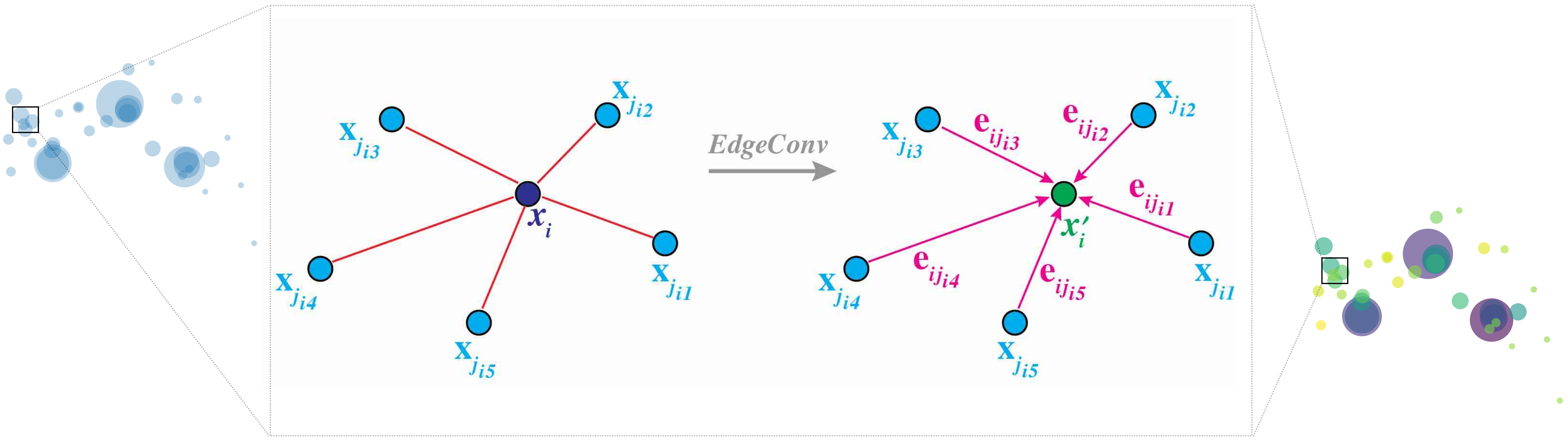



- ParticleNet: jet tagging via particle clouds
  - treating a jet as an **unordered set of particles**, distributed in the $\eta - \phi$ space
  - **graph neural network architecture**, adapted from Dynamic Graph CNN [arXiv:1801.07829]
    - treating a point cloud as a graph: each point is a vertex
      - for each point, a local patch is defined by finding its k-nearest neighbors
    - designing a permutation-invariant "convolution" function
      - define "edge feature" for each center-neighbor pair: $e_{ij} = h_\Theta(x_i, x_j)$
      - aggregate the edge features in a symmetric way: $x_i' = \mathrm{mean}_j\, e_{ij}$

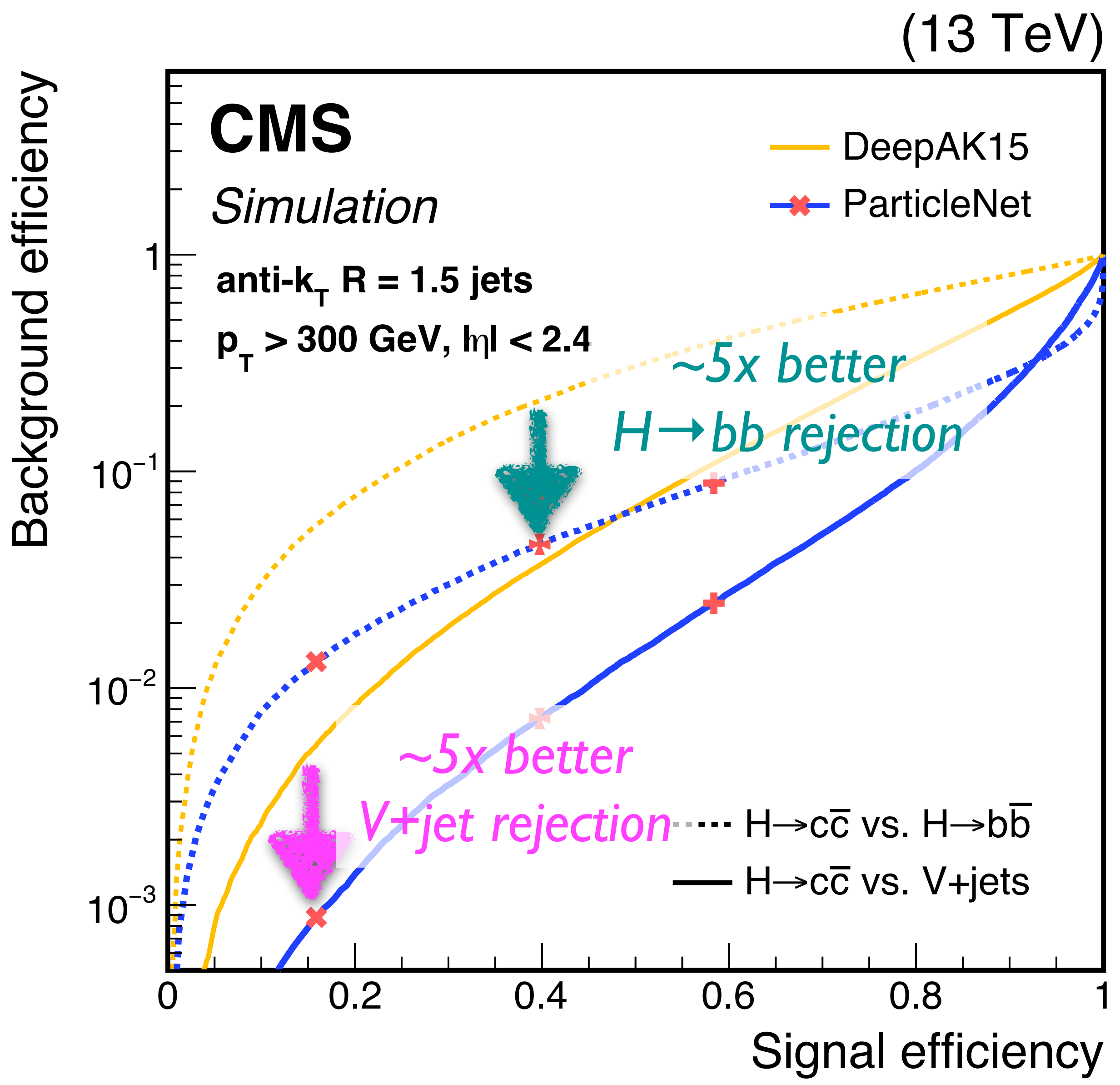
*HQ and L. Gouskos [arXiv: 1902.08570]*

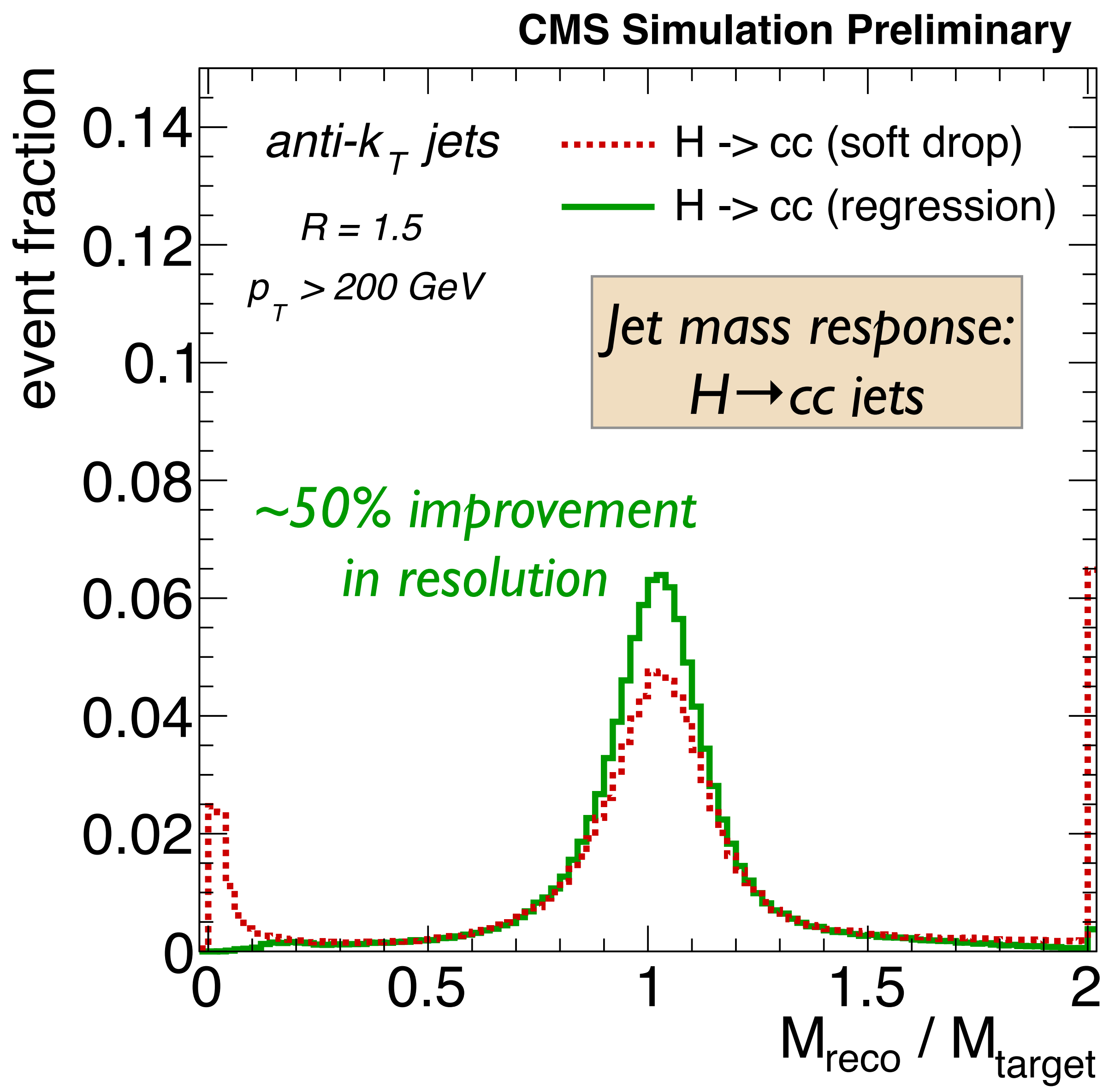
*ParticleNet architecture*



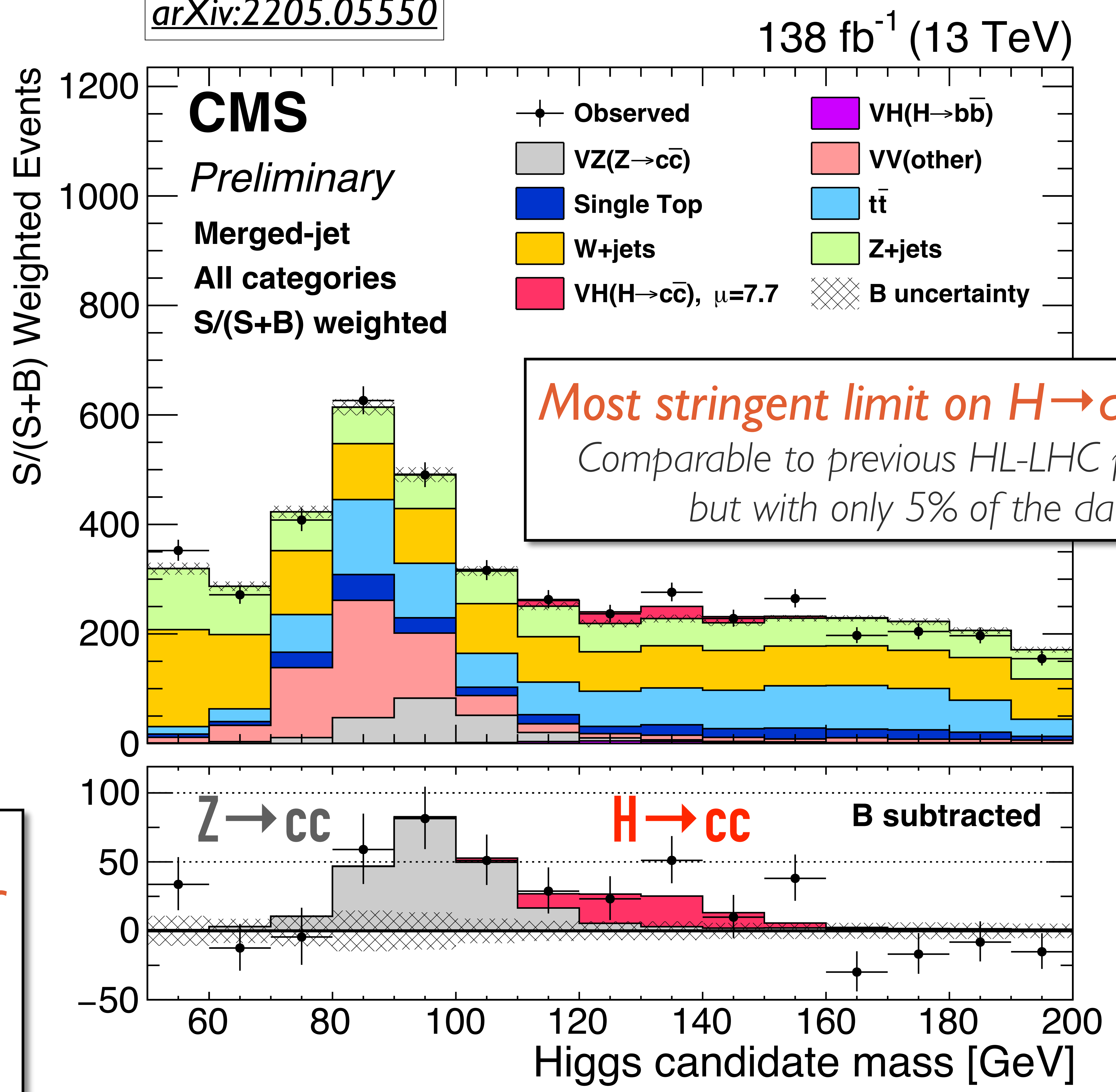



# ParticleNet in Action: H→cc Search




arXiv:2205.05550




*Most stringent limit on H→cc to date.*

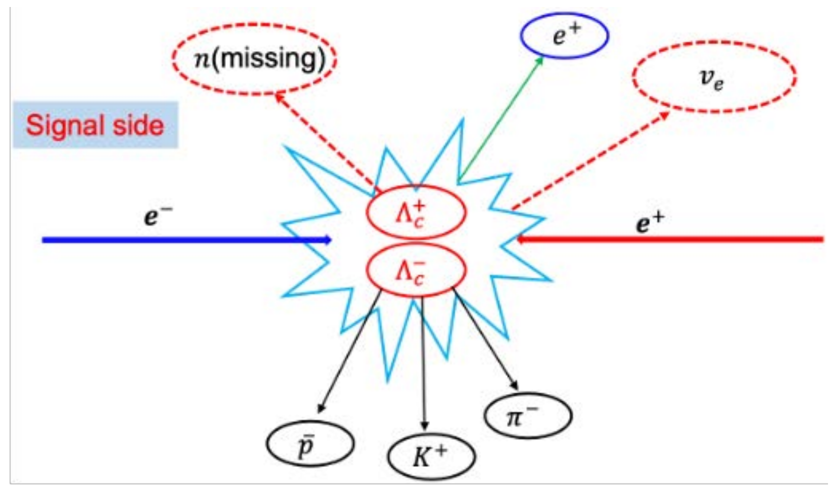
*Comparable to previous HL-LHC projection but with only 5% of the data.*

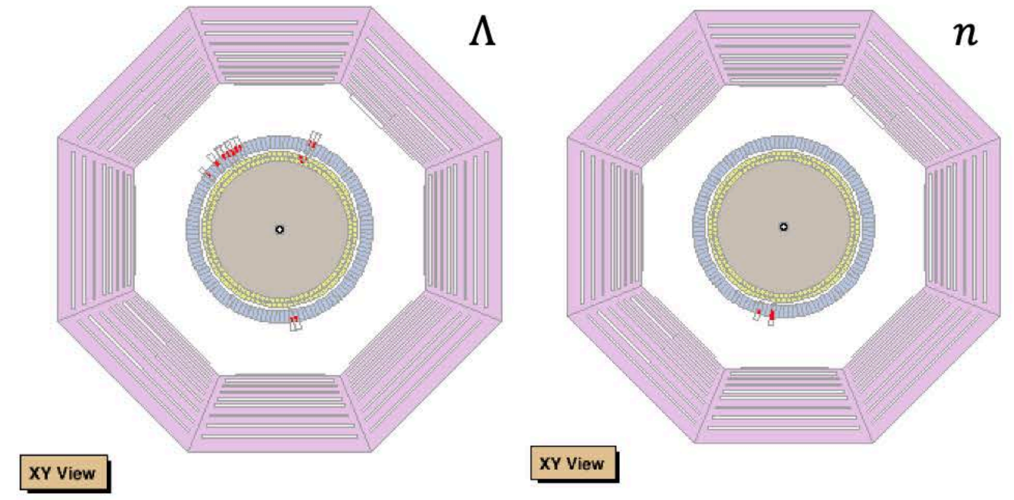
*First observation of Z→cc at a hadron collider!*

# ParticleNet in Action: Beyond Jets

## $\Lambda_c^+ \to n e^+ \nu$ measurement

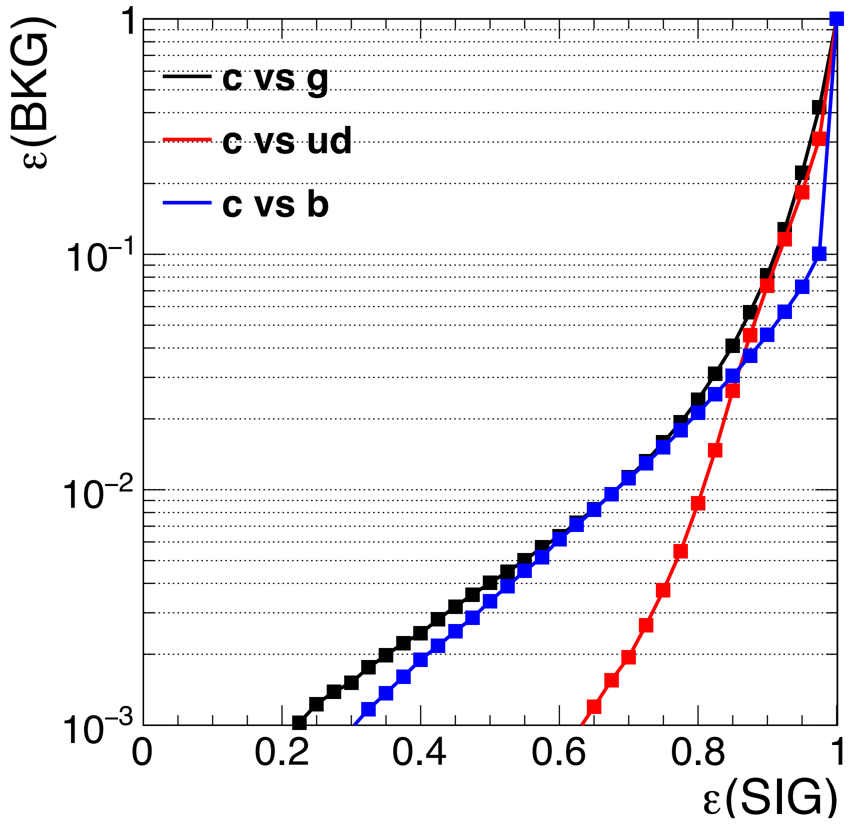
*Yunxuan Song, Yangu Li et al., BAM-00632*

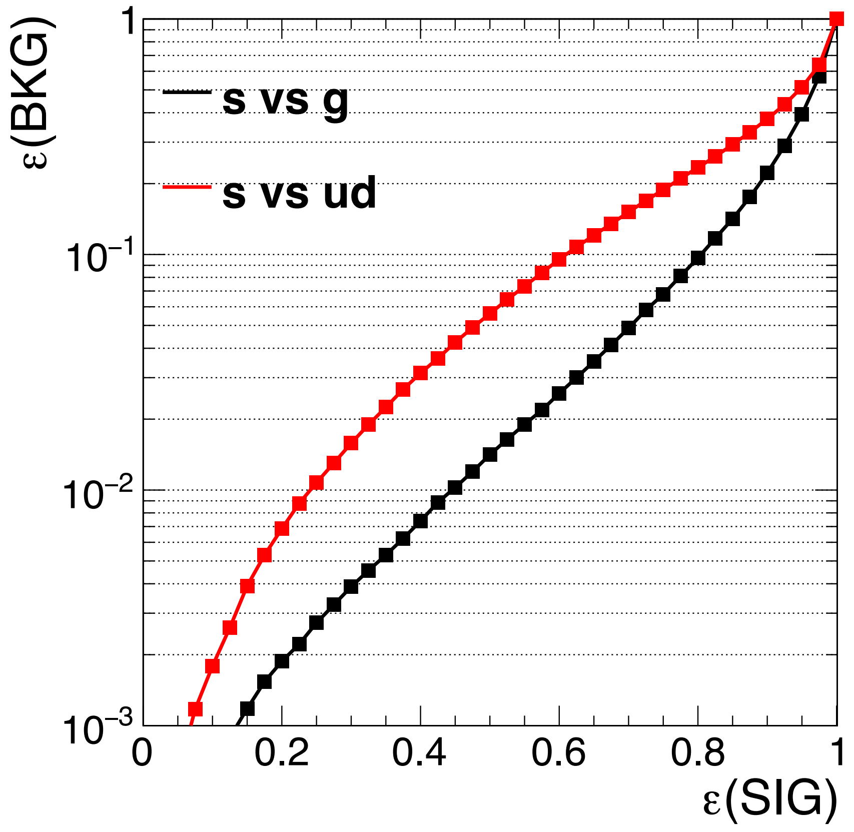



## Particle identification

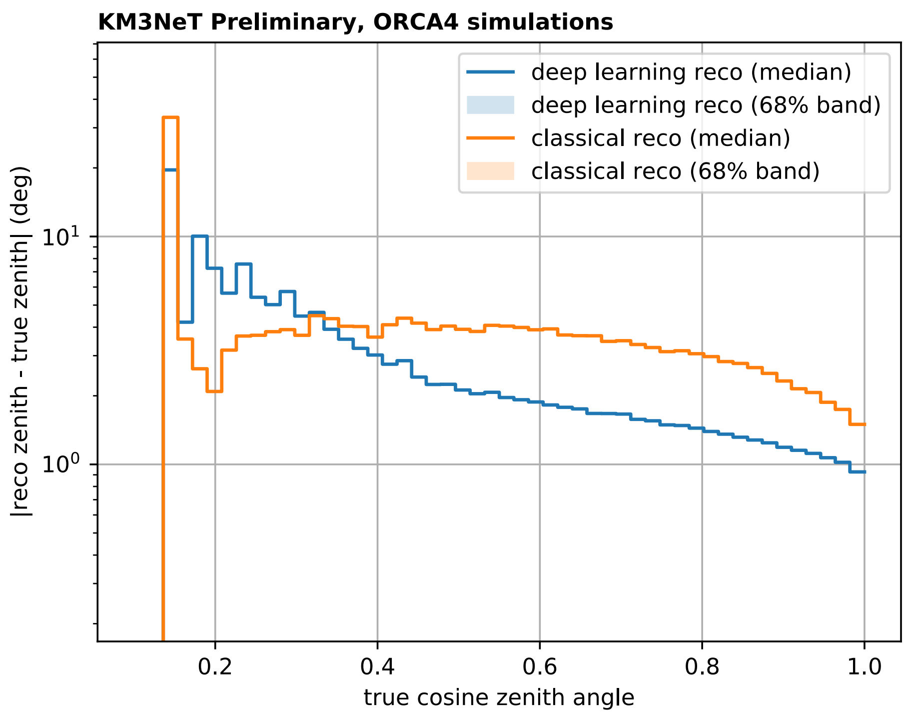
Eur.Phys.J.Plus 137 (2022) 1, 39
Eur.Phys.J.C 82 (2022) 7, 646



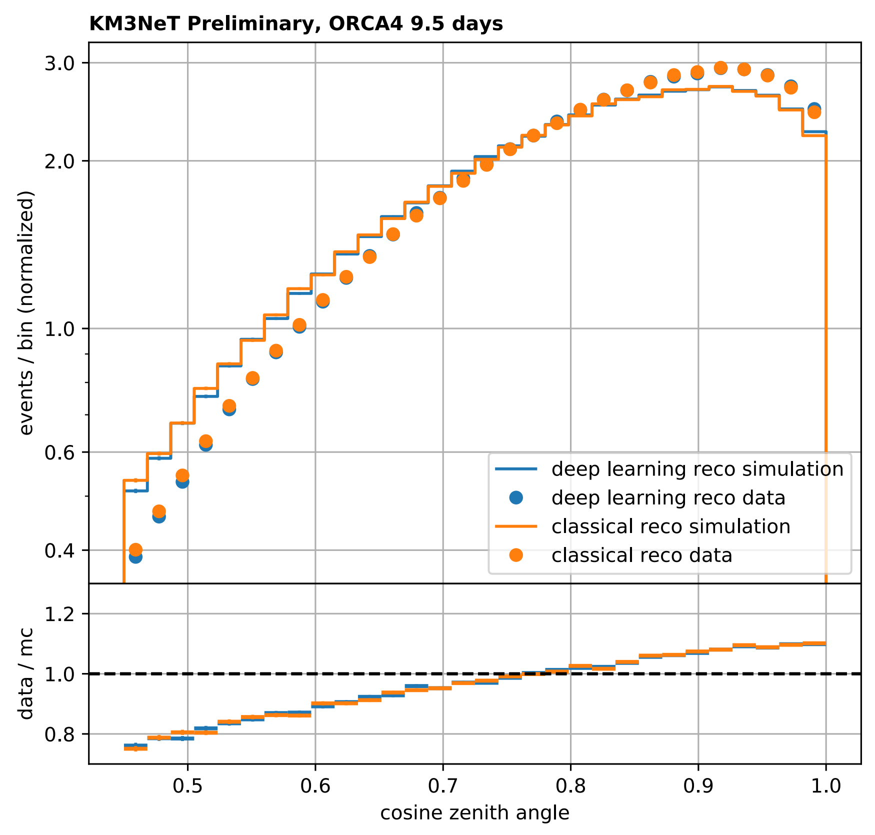




## Muon bundle reconstruction

JINST 16 (2021) 10, C10011,
PoS ICRC2021 (2021) 1048

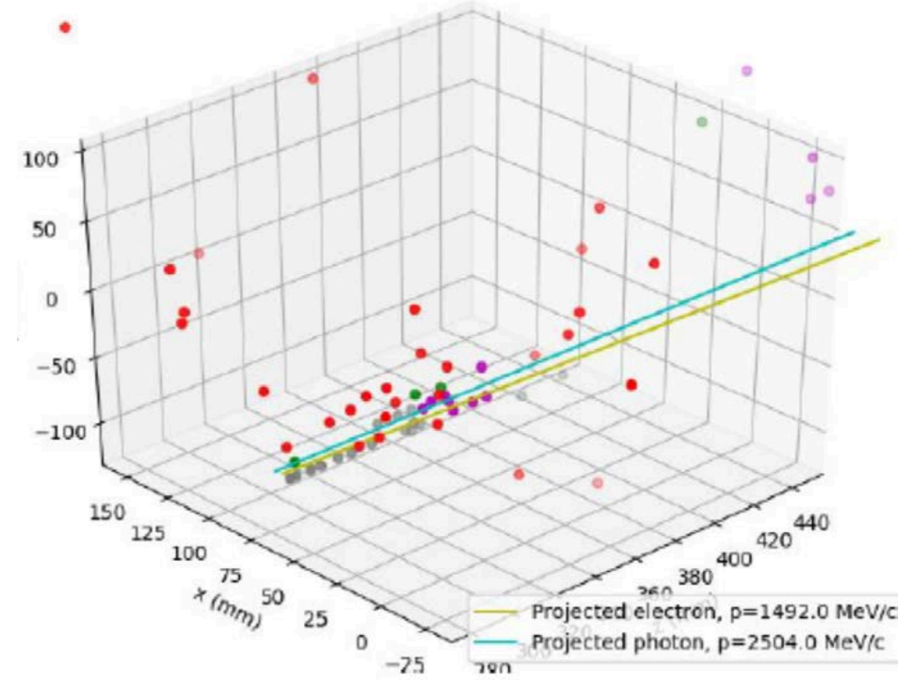


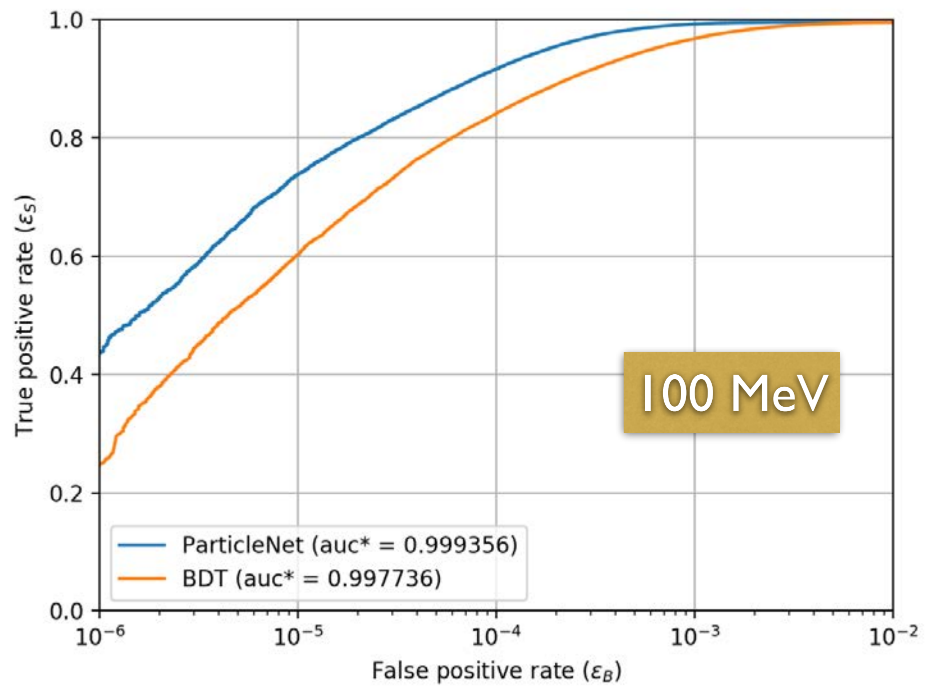

**(b)** events with two or more muons




## Photo-nuclear background rejection

*Work in progress*



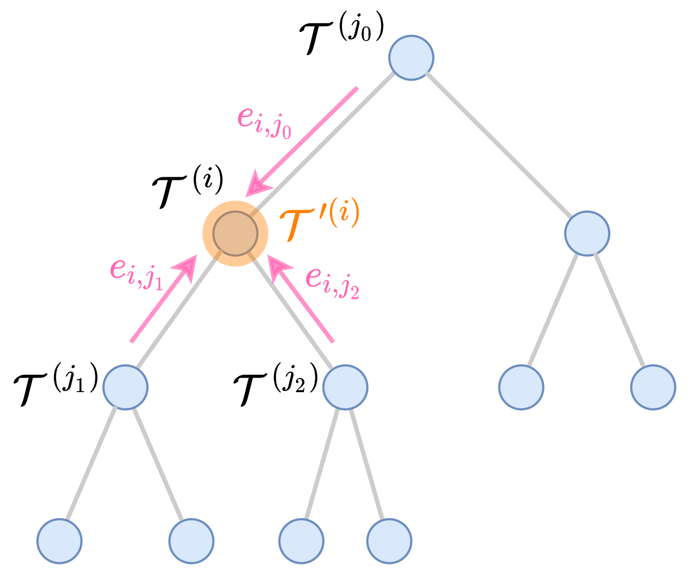



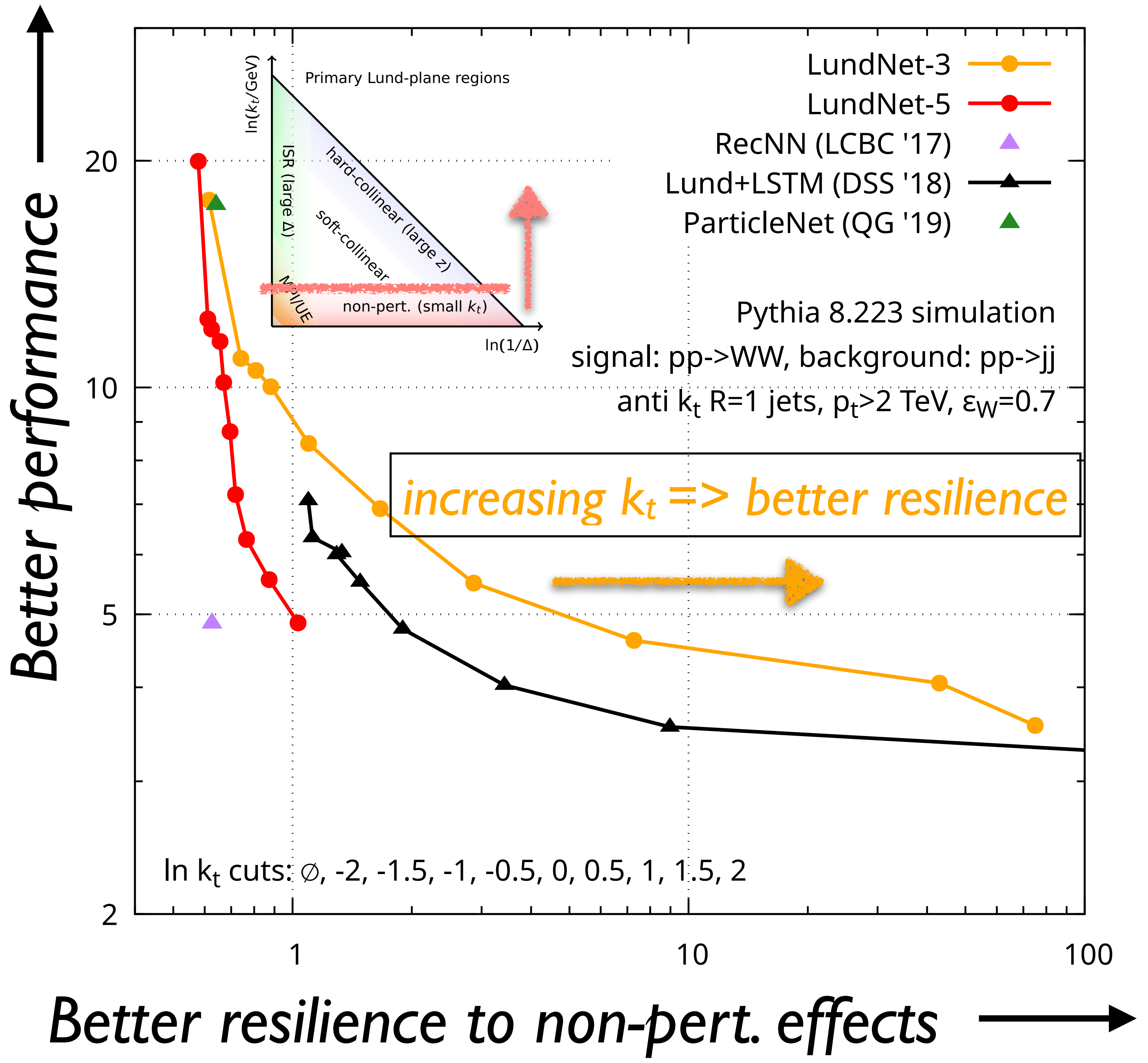
# Beyond ParticleNet

## *LundNet*

*F. Dreyer & HQ*
*JHEP (2021)*

*GNN + Lund jet plane (jet tree)*

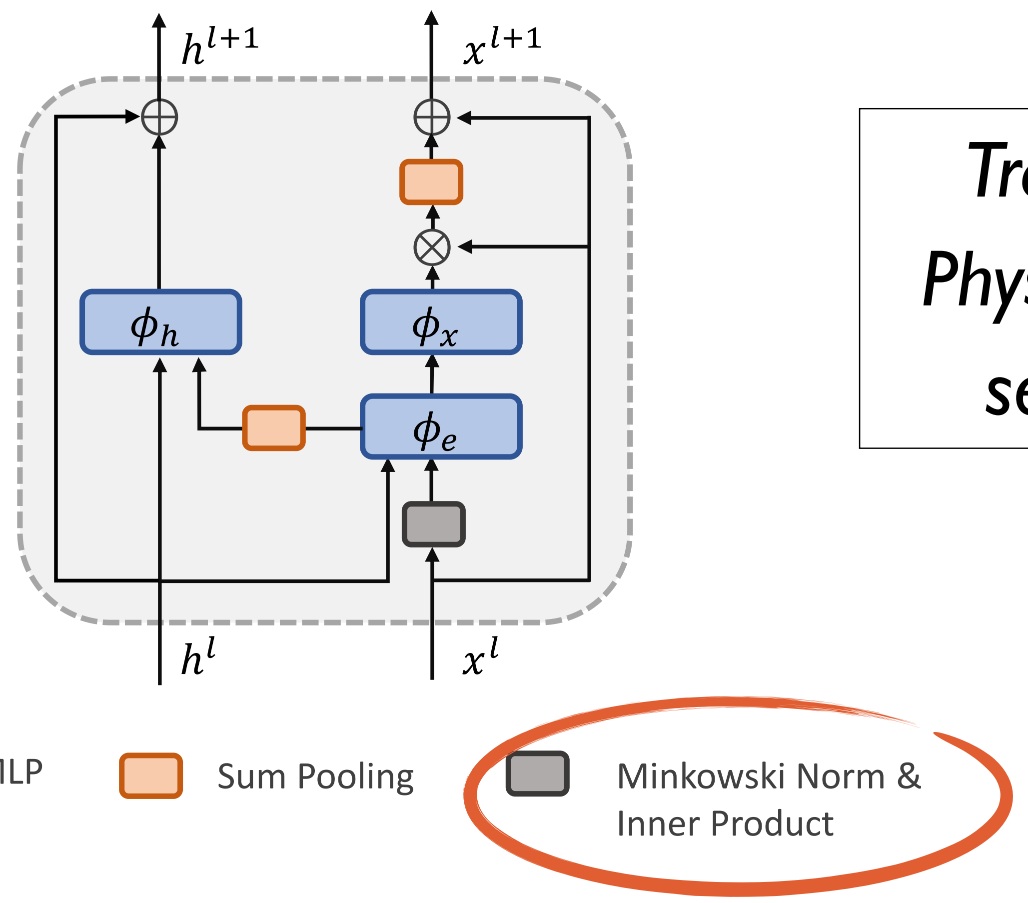





## *LorentzNet*

*S. Gong, Q. Meng, J. Zhang, HQ, C. Li, S. Qian et. al.*
*JHEP (2022)*

*GNN + Lorentz symmetry*



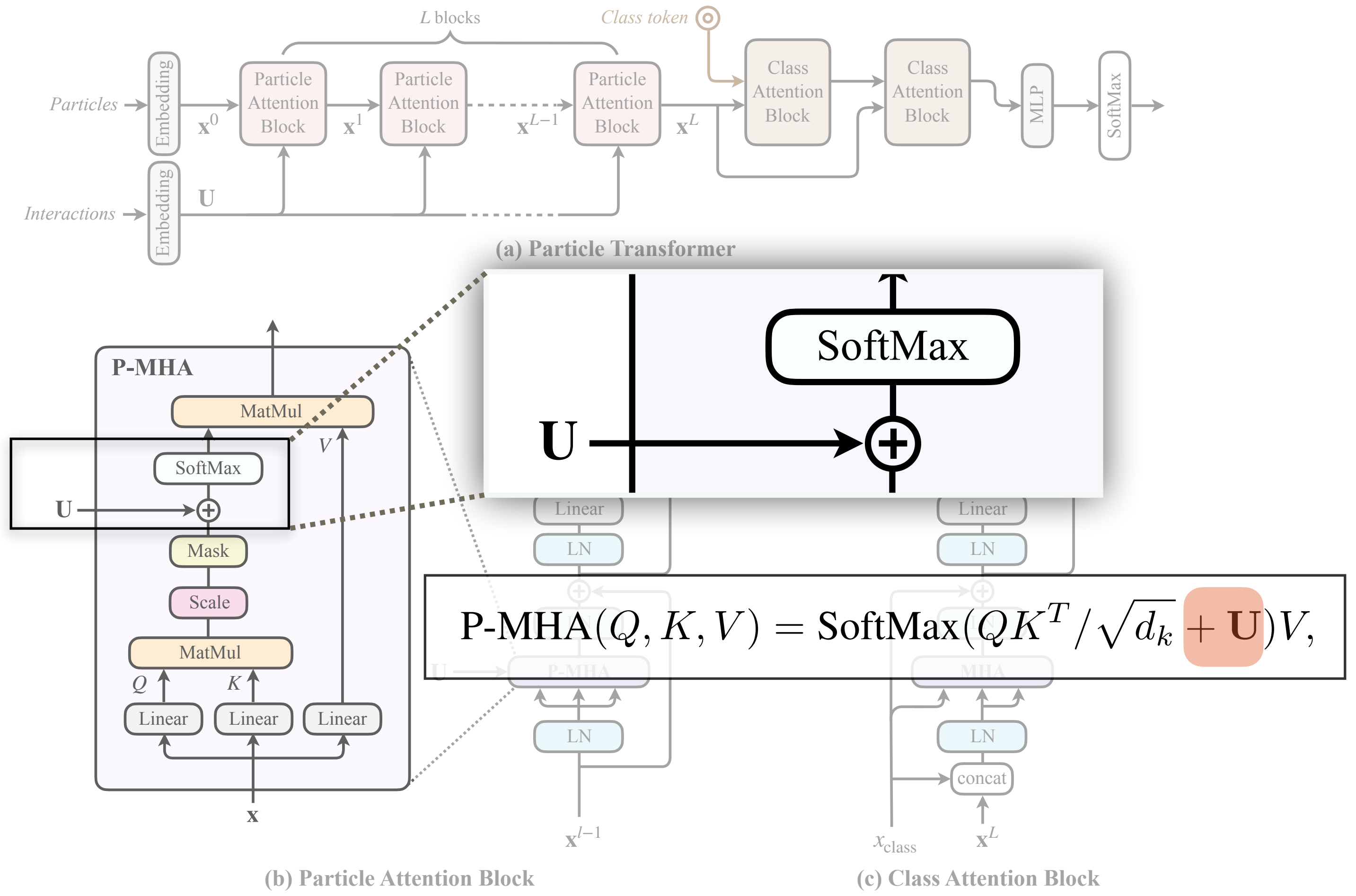

| Training Fraction | Model | Accuracy | AUC | $1/\varepsilon_B$ ($\varepsilon_S = 0.5$) | $1/\varepsilon_B$ ($\varepsilon_S = 0.3$) |
|---|---|---|---|---|---|
| 0.5% | ParticleNet | 0.913 | 0.9687 | $77 \pm 4$ | $199 \pm 14$ |
| | LorentzNet | **0.929** | **0.9793** | $\mathbf{176 \pm 14}$ | $\mathbf{562 \pm 72}$ |
| 1% | ParticleNet | 0.919 | 0.9734 | $103 \pm 5$ | $287 \pm 19$ |
| | LorentzNet | **0.932** | **0.9812** | $\mathbf{209 \pm 5}$ | $\mathbf{697 \pm 58}$ |
| 5% | ParticleNet | 0.931 | 0.9807 | $195 \pm 4$ | $609 \pm 35$ |
| | LorentzNet | **0.937** | **0.9839** | $\mathbf{293 \pm 12}$ | $\mathbf{1108 \pm 84}$ |

*Symmetry preservation greatly improves the generalization power.*

## *Particle Transformer*

*HQ, C. Li and S. Qian*
*ICML 2022*

*Transformer + Physics-enhanced self-attention*

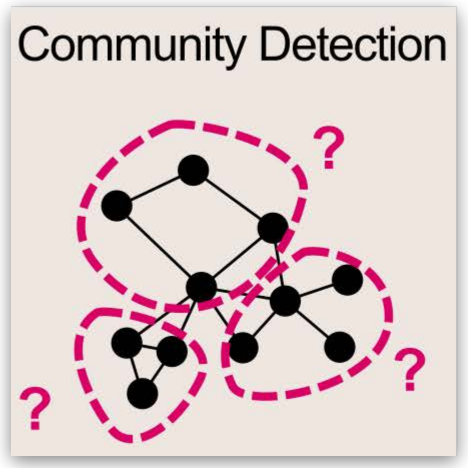



$$\text{P-MHA}(Q, K, V) = \text{SoftMax}(QK^T/\sqrt{d_k} + \mathbf{U})V,$$

| | All classes Accuracy | AUC | $H \to b\bar{b}$ Rej$_{50\%}$ | $H \to c\bar{c}$ Rej$_{50\%}$ | $H \to gg$ Rej$_{50\%}$ | $H \to 4q$ Rej$_{50\%}$ |
|---|---|---|---|---|---|---|
| PFN | 0.772 | 0.9714 | 2924 | 841 | 75 | 198 |
| P-CNN | 0.809 | 0.9789 | 4890 | 1276 | 88 | 474 |
| ParticleNet | 0.844 | 0.9849 | 7634 | 2475 | 104 | 954 |
| **ParT** | **0.861** | **0.9877** | **10638** | **4149** | **123** | **1864** |
| ParT (plain) | 0.849 | 0.9859 | 9569 | 2911 | 112 | 1185 |

*New state-of-the-art. Significantly outperforms ParticleNet.*

# GNNs for Reconstruction

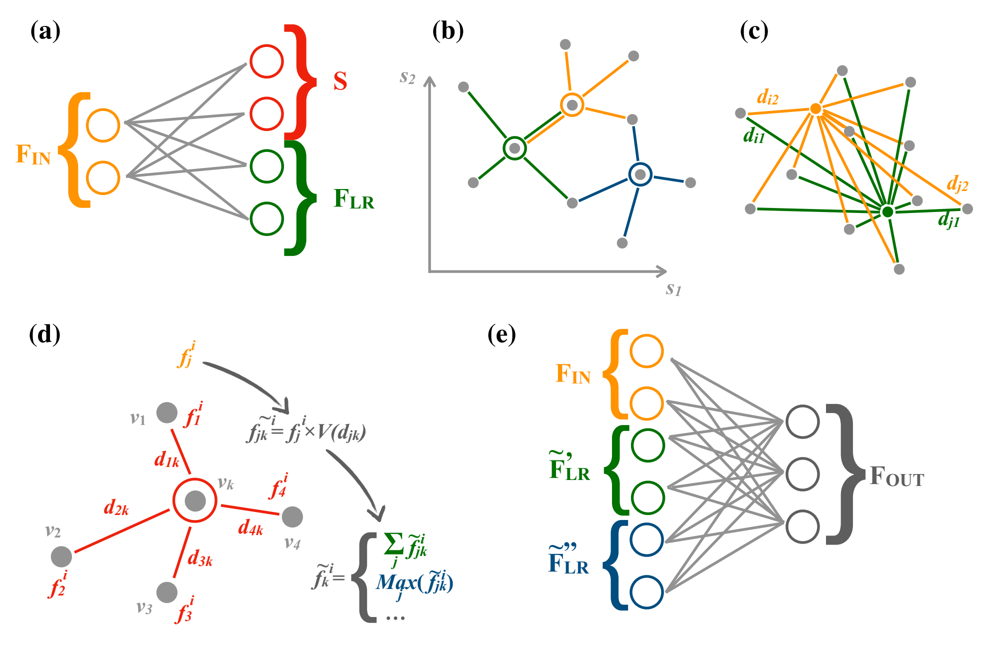



- GNNs also powerful tools for event reconstruction, particularly for non-uniform detector geometry

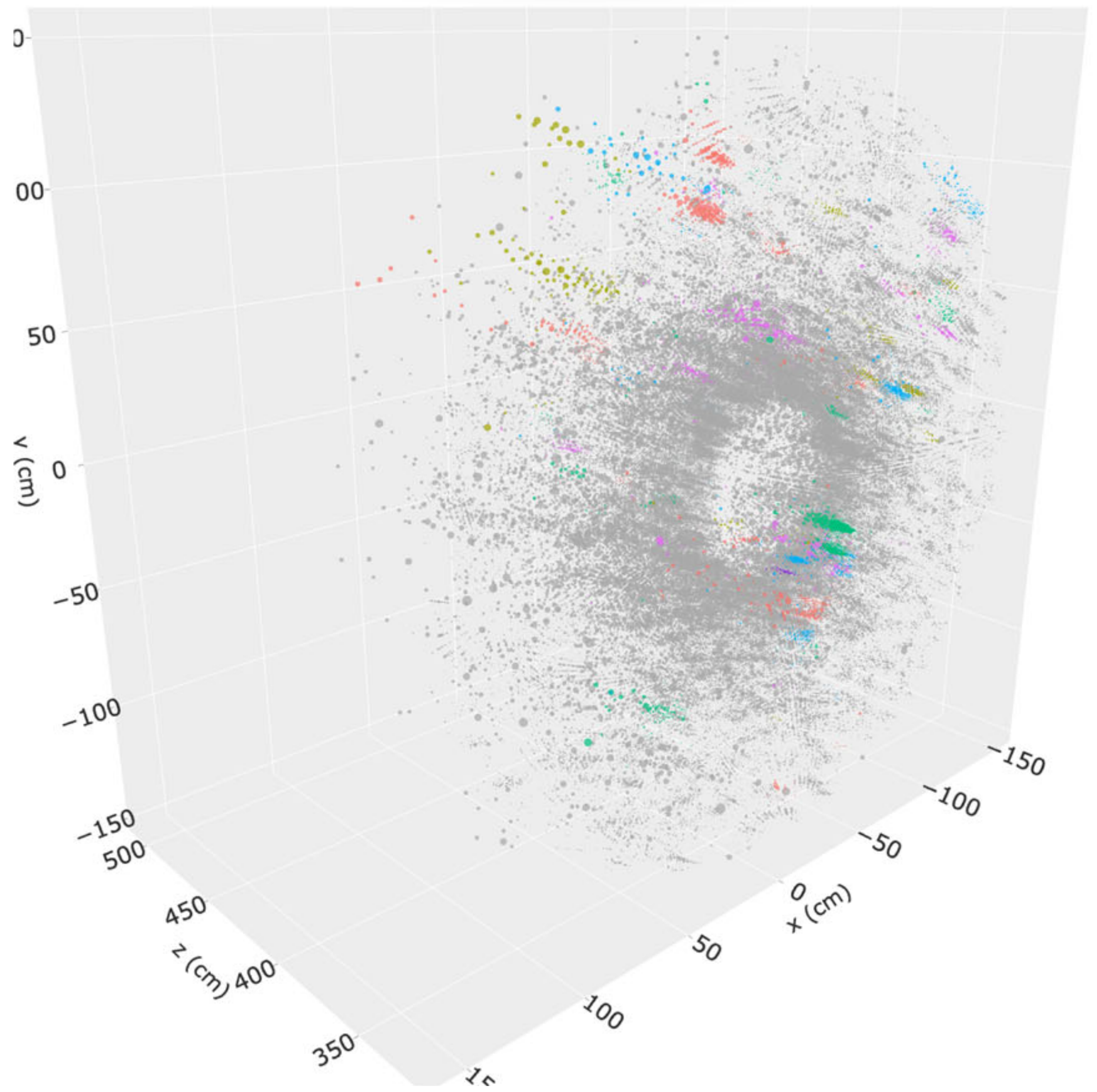
- Distance-weighted GNNs: GarNet/GravNet
  - much lower computational cost than DGCNN
  - GarNet: lightweight, can be implemented on FPGA for e.g., event triggering

- Object condensation: one-stage multi-object reconstruction
  - simultaneously predict the number of showers and their properties
  - in addition: cluster hits belonging to shower in a clustering space by using attractive/repulsive potentials in the loss

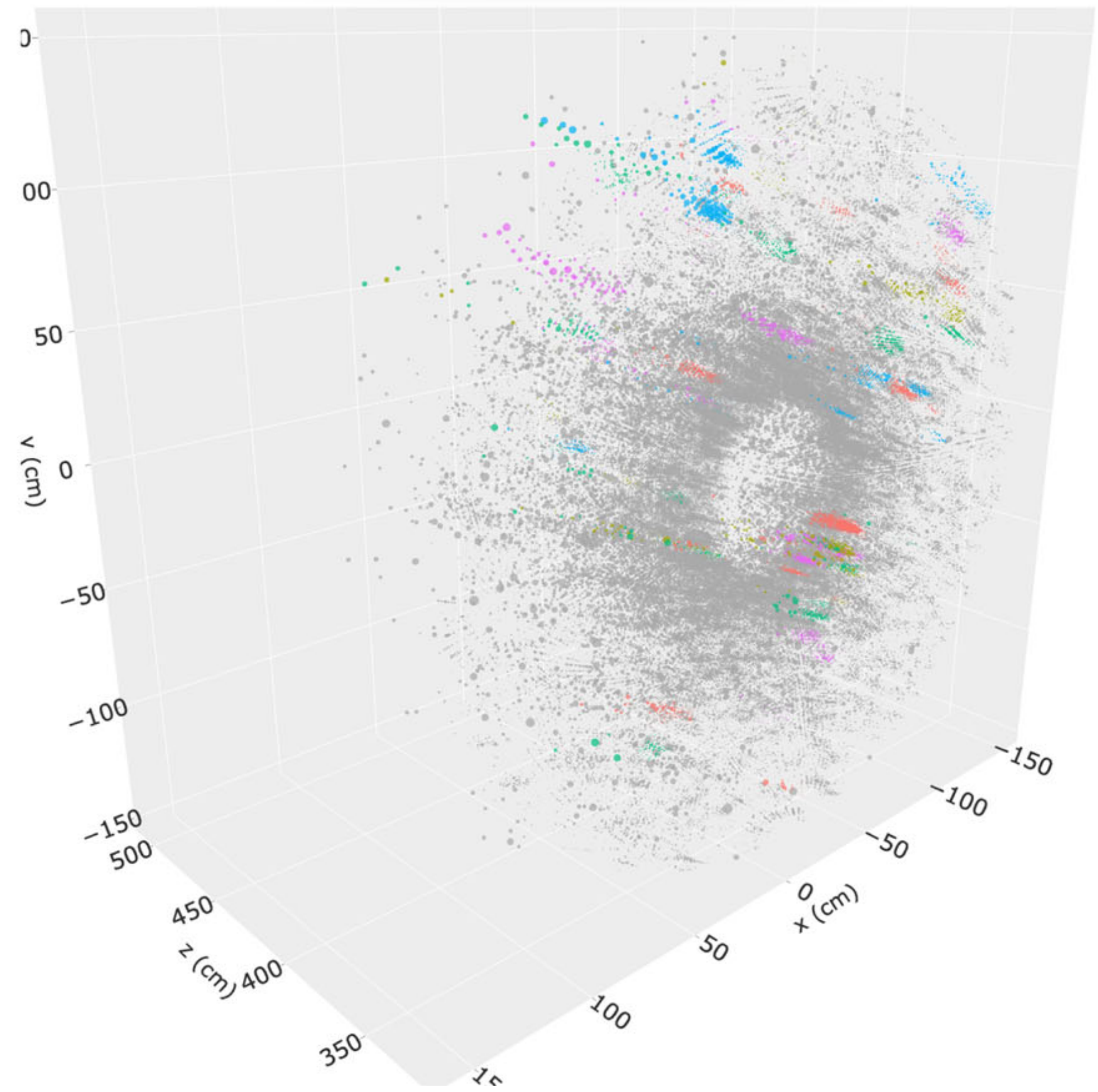
*GarNet/GravNet*



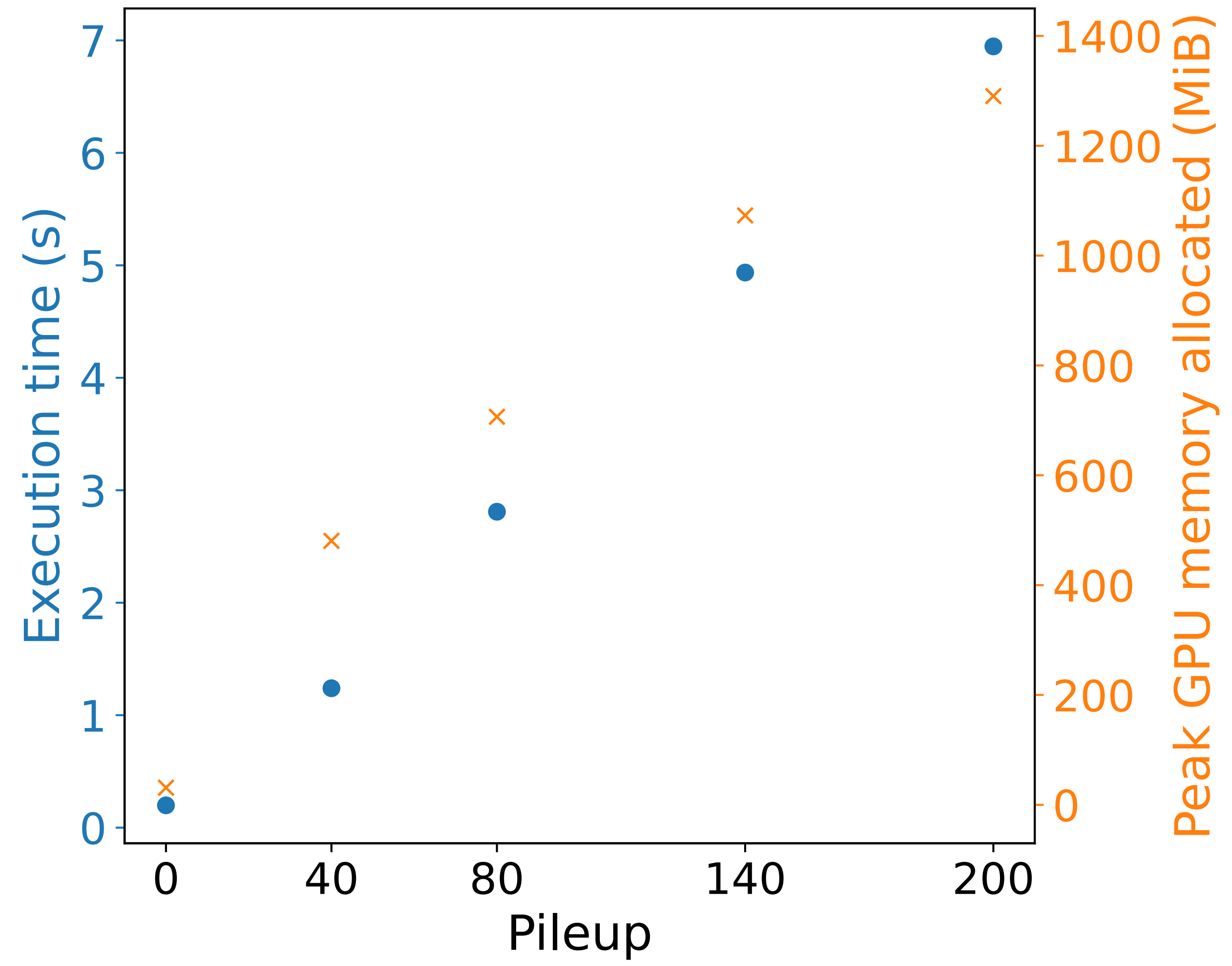
*Truth*



*Reconstructed*



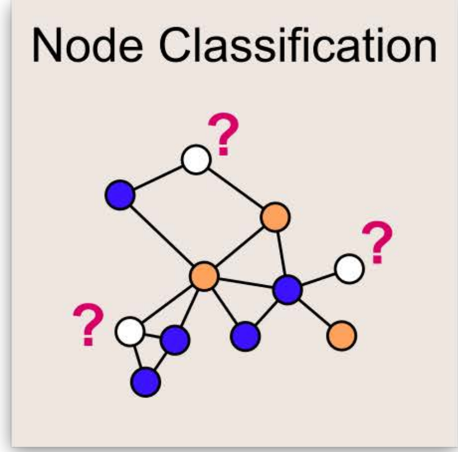
*Time and memory usage*



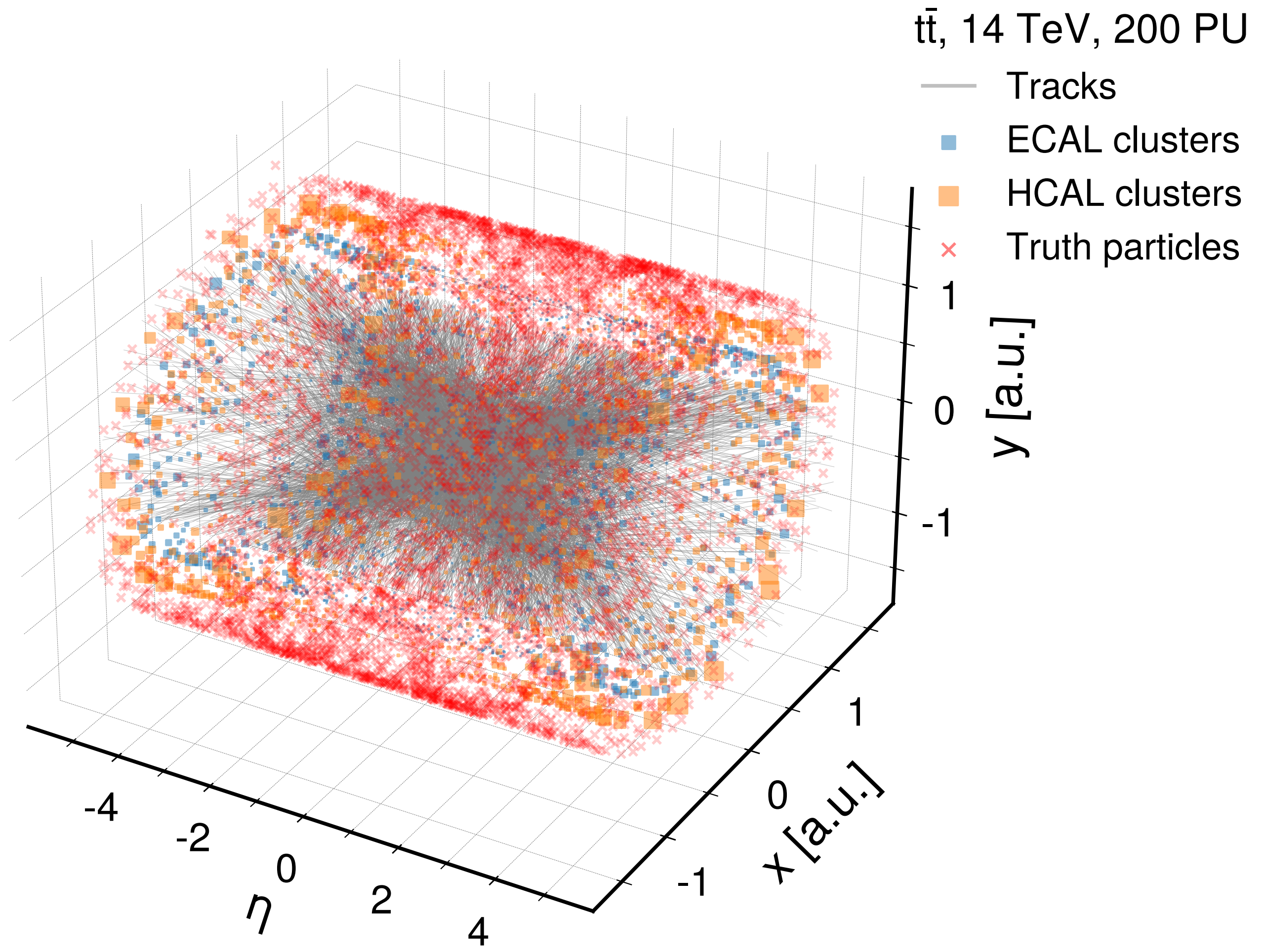

*S. R. Qasim, J. Kieseler, Y. Iiyama and M. Pierini [arXiv:1902.07987]; J. Kieseler [arXiv:2002.03605]; S. R. Qasim et. al., [arXiv:2204.01681]*

# GNNs for Particle Flow

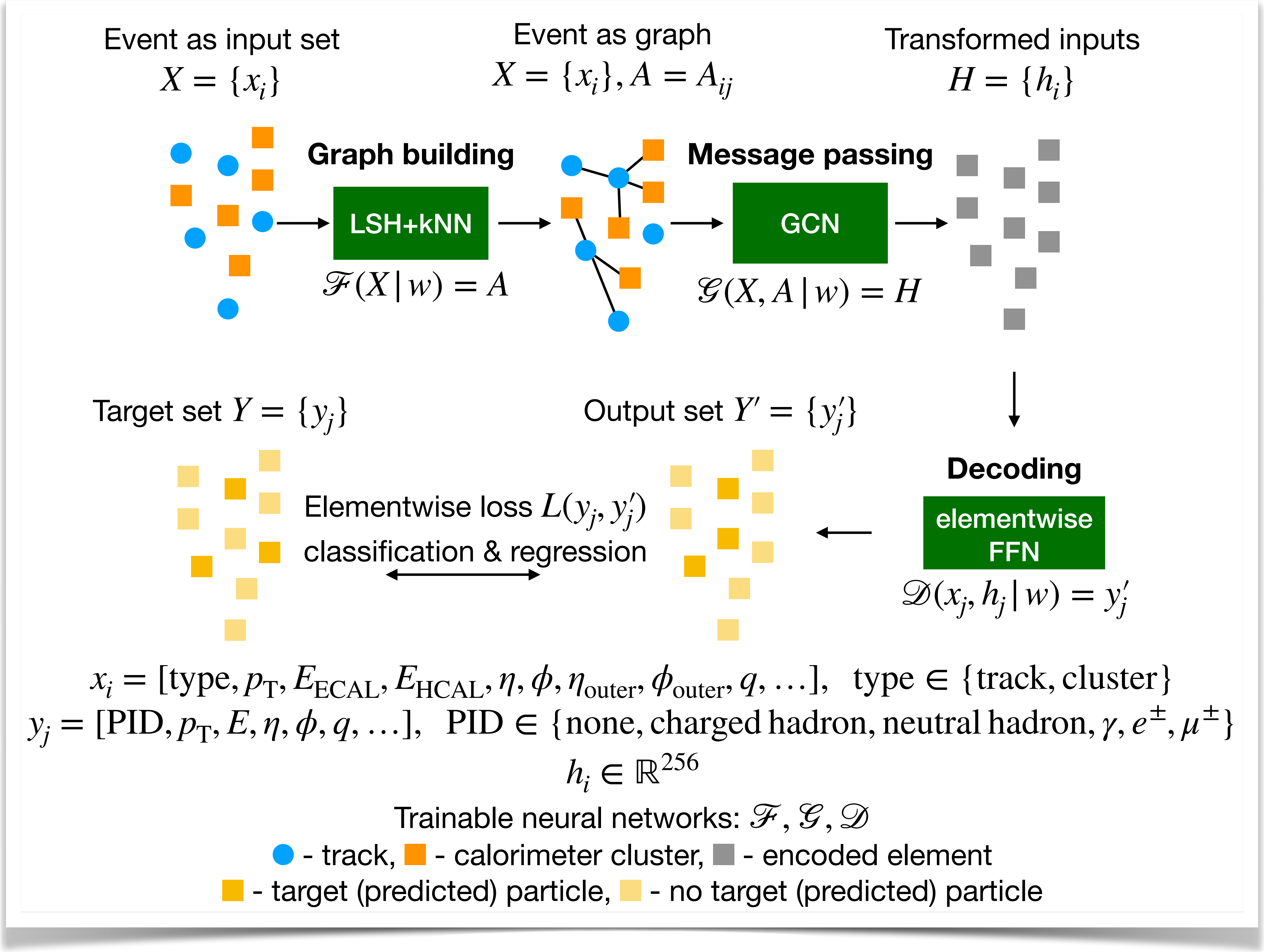
*J. Pata, J. Duarte, J. R. Vlimant, M. Pierini and M. Spiropulu [arXiv: 2101.08578]*



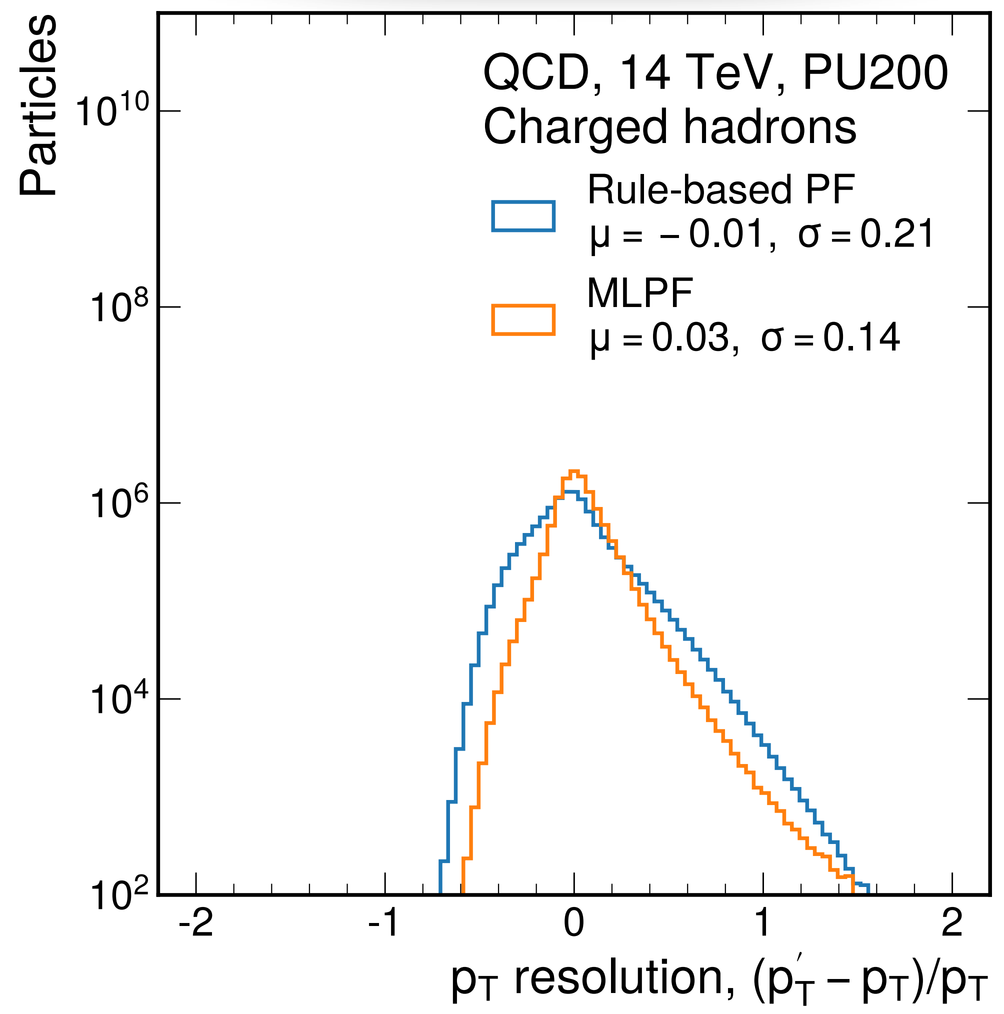

- Use GNNs to directly perform end-to-end particle flow reconstruction
  - comparable/better performance than rule-based PF on Delphes dataset
  - runtime scales linearly with input size, no quartic explosion



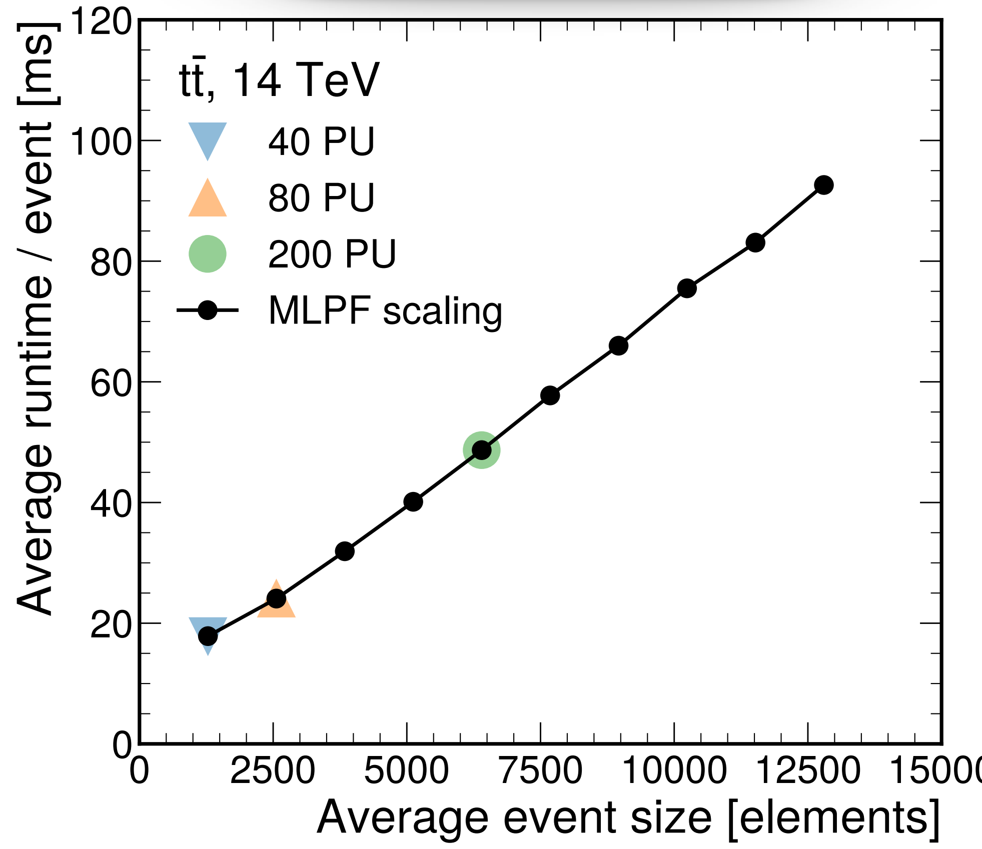

*Delphes simulation*



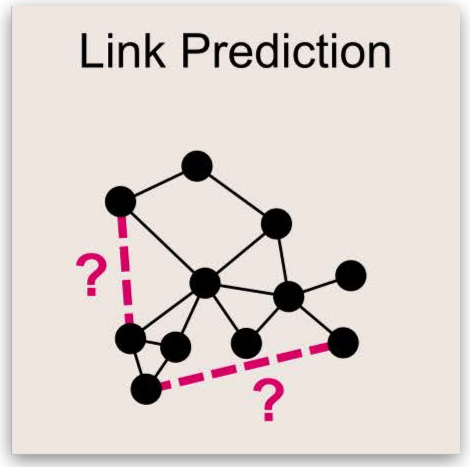

**Resolution**




**Inference time**

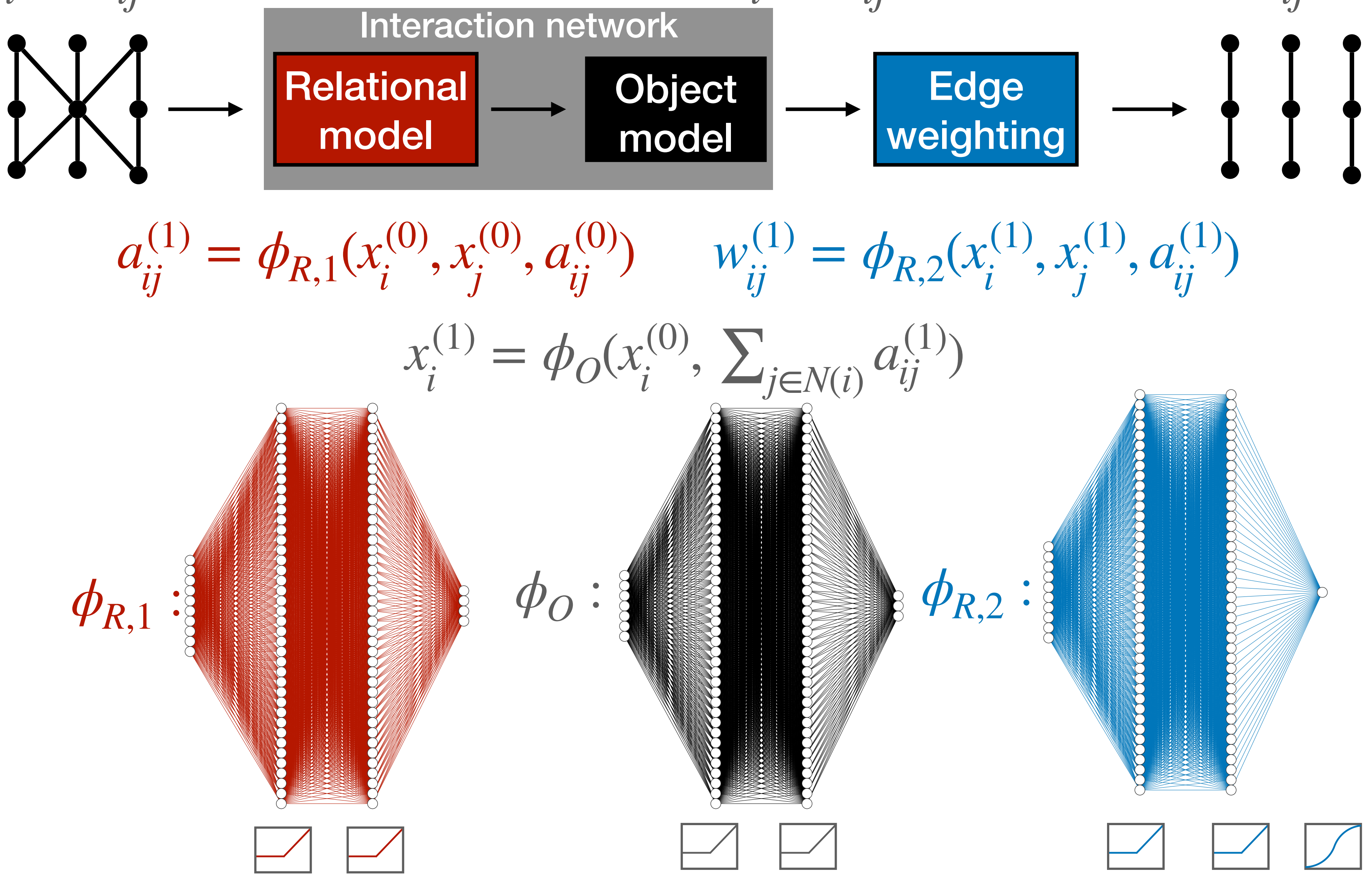


# GNNs for Tracking




G. DeZoort et al. arXiv:2103.16701

- Charged particle tracking as an edge prediction task within the GNN framework
  - each hit is a node of the graph
  - edges constructed between pairs of hits with geometrically plausible relations
  - classify whether each edge connects hits belonging to the same track or not

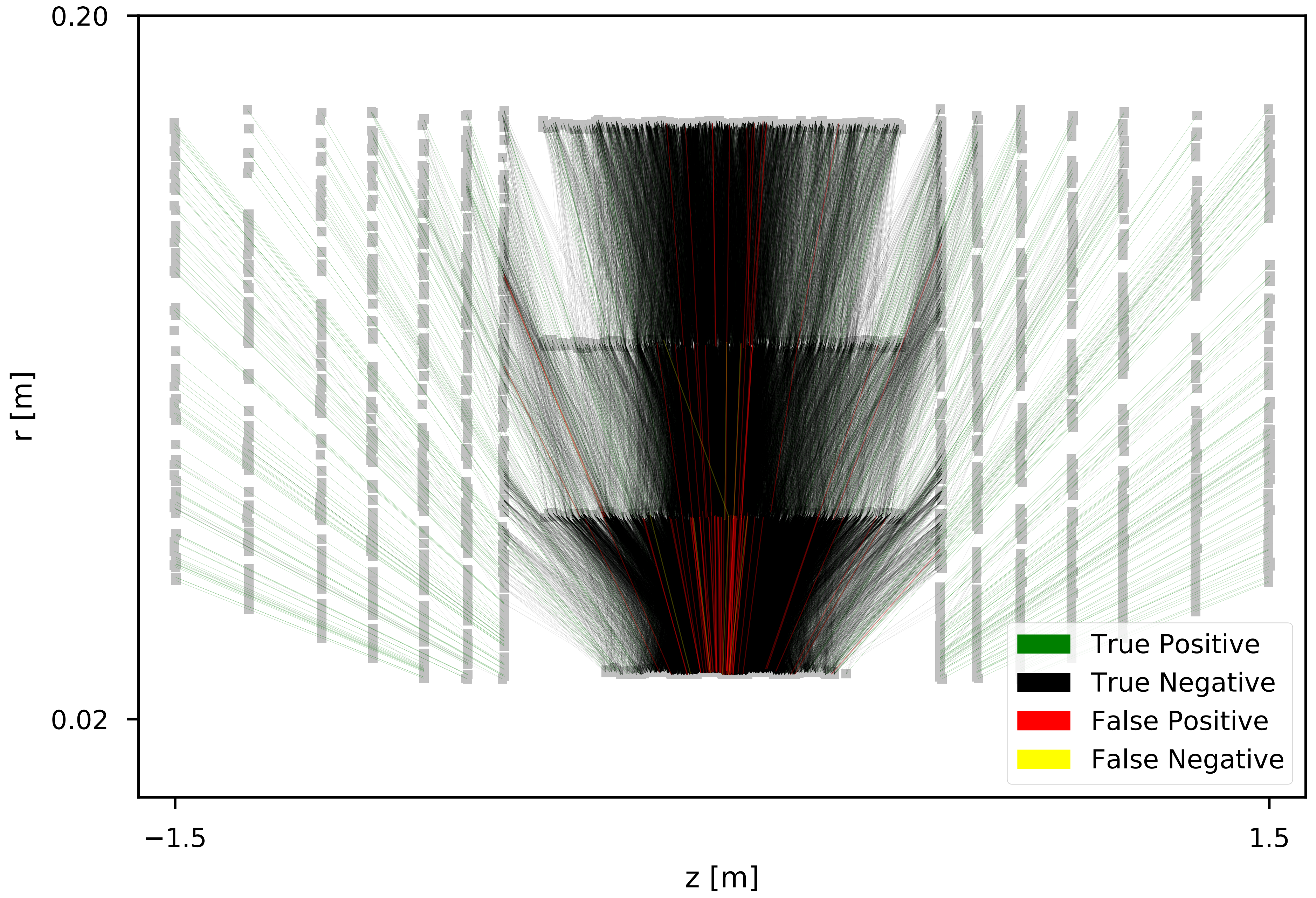



$(x_i^{(0)}, a_{ij}^{(0)})$ $(x_i^{(1)}, a_{ij}^{(1)})$ $(w_{ij}^{(1)})$

$a_{ij}^{(1)} = \phi_{R,1}(x_i^{(0)}, x_j^{(0)}, a_{ij}^{(0)})$ $w_{ij}^{(1)} = \phi_{R,2}(x_i^{(1)}, x_j^{(1)}, a_{ij}^{(1)})$

$x_i^{(1)} = \phi_O(x_i^{(0)}, \sum_{j \in N(i)} a_{ij}^{(1)})$

$\phi_{R,1}$ : $\phi_O$ : $\phi_{R,2}$ :




See also: S. Farrell et al. [1810.06111]; X. Ju et al. [2003.11603];
C. Biscarat, S. Caillou, C. Rougier, J. Stark and J. Zahreddine [2103.00916]; X. Ju et al. [2103.06995]; etc.

# Summary & Outlook

- Graph neural networks: a powerful and flexible framework with increasing adoption in HEP
  - state-of-the-art performance in jet tagging, particle identification, event classification, …
  - active R&D for event reconstruction, particle flow, tracking, …
  - moreover: generative models (e.g., for fast simulation), representation learning (e.g., for anomaly detection), …

- Outlook
  - more powerful architectures => better performance
  - more effective incorporation of physics knowledge => better robustness
  - improving computational efficiency (latency/throughput/memory/etc.)
  - and eventually:
    - increased sensitivity to new physics at various frontiers!

# BACKUPS

# LorentzNet

- Incorporating Lorentz symmetry into graph neural network architecture

S. Gong, Q. Meng, J. Zhang, H. Qu, C. Li, S. Qian, W. Du, Z. M. Ma and T.Y. Liu, arXiv: 2201.08187

*Coordinate input:* $x^0$

*Feature input:* $h_i^0$

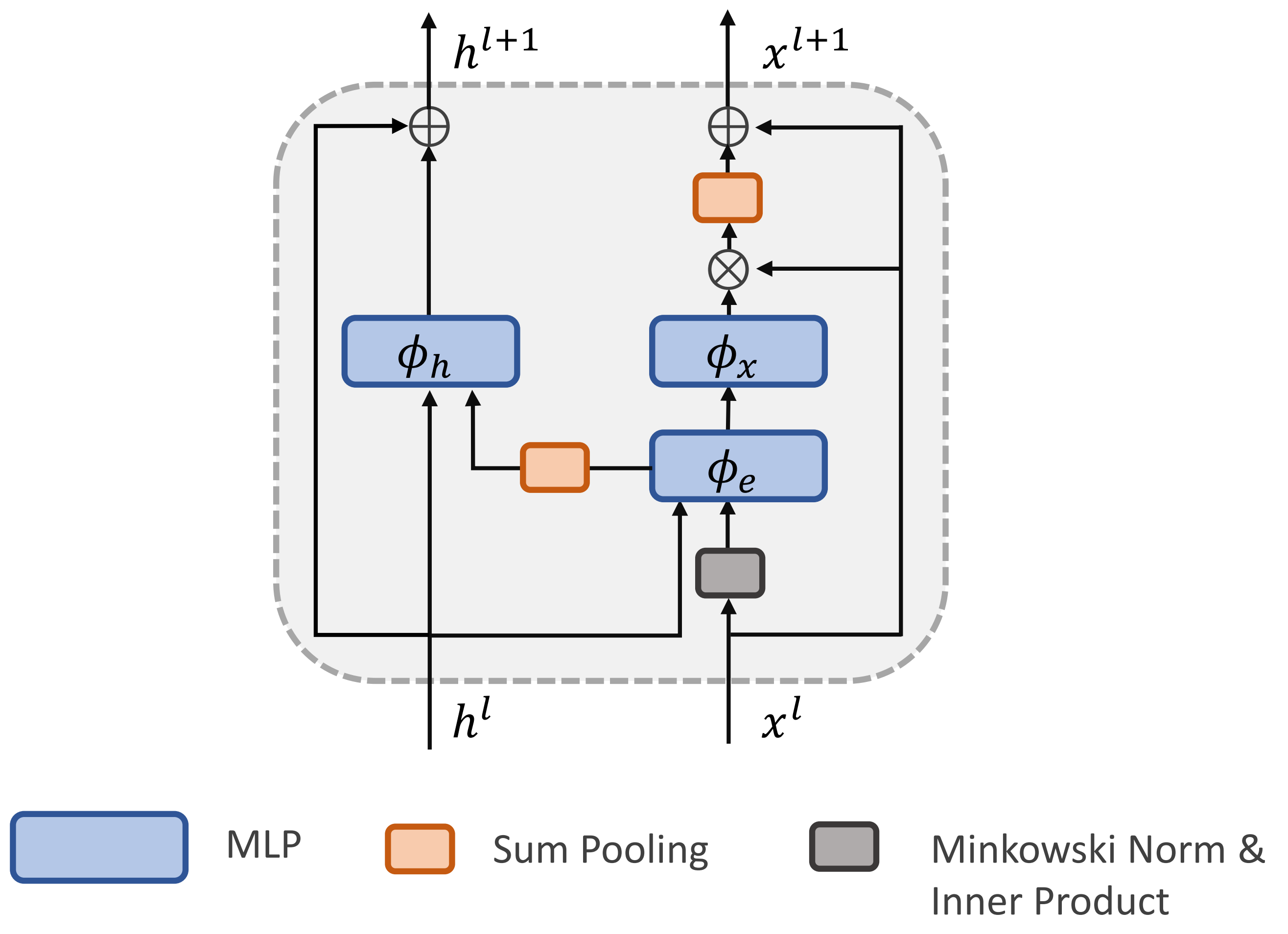
Lorentz 4-vector
Lorentz scalar

*Message:* $m_{ij}^l = \phi_e\left(h_i^l, h_j^l, \psi(\|x_i^l - x_j^l\|^2), \psi(\langle x_i^l, x_j^l\rangle)\right)$

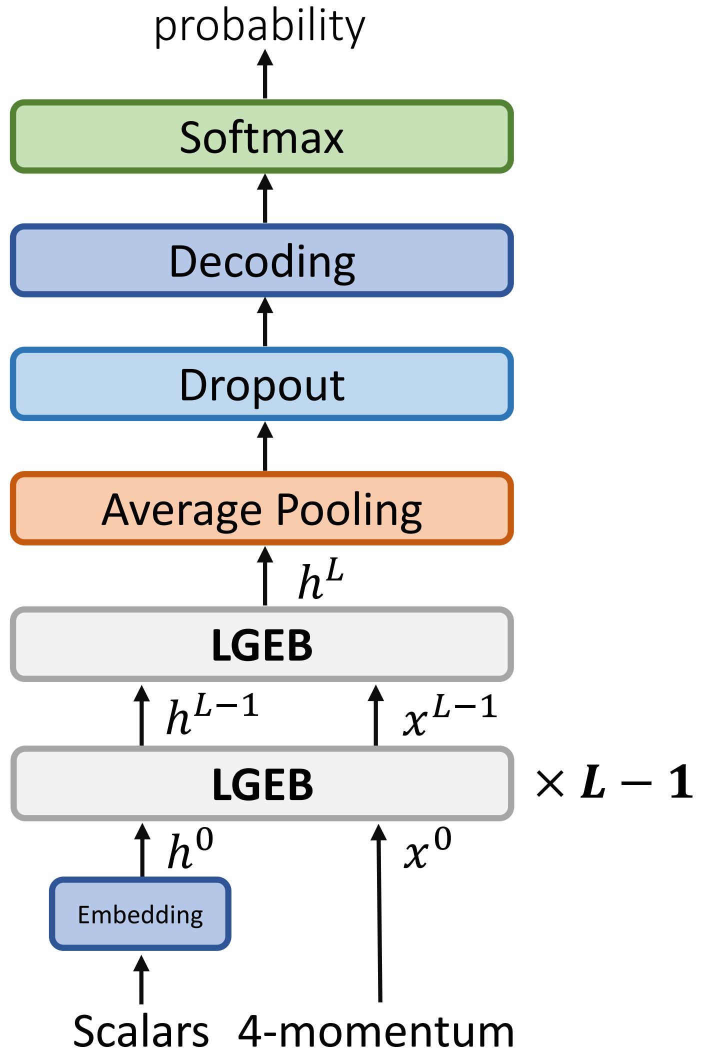
Scalars — Pairwise Lorentz invariants

*Coordinate update:* $x_i^{l+1} = x_i^l + c \sum_{j\in[N]} \phi_x(m_{ij}^l)\cdot x_j^l$

*Feature update:* $h_i^{l+1} = h_i^l + \phi_h(h_i^l, \sum_{j\in[N]} w_{ij} m_{ij}^l)$



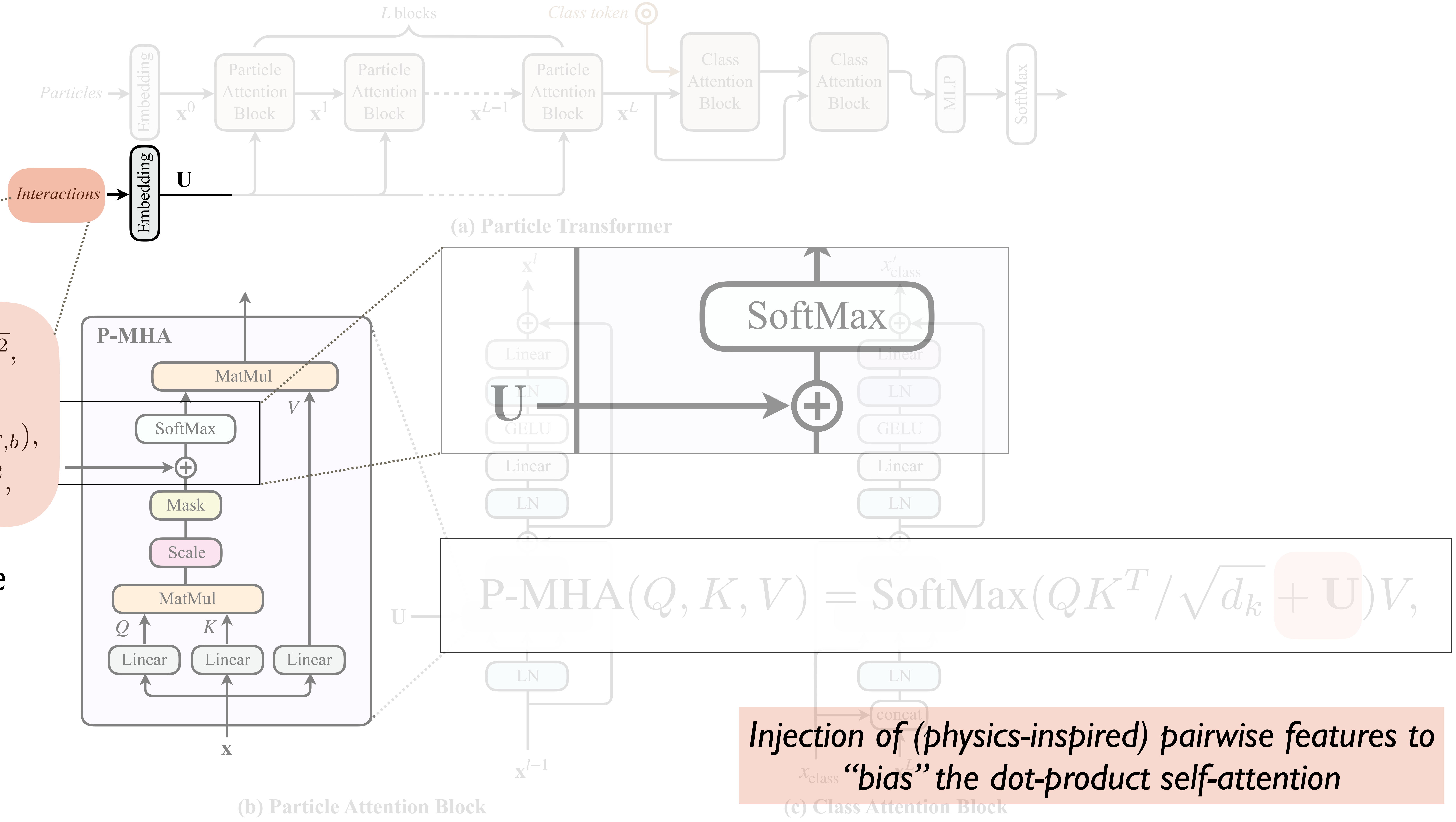
**Lorentz Group Equivariant Block (LGEB)**



**LorentzNet**

cf. A. Bogatskiy, B. Anderson, J. Offermann, M. Roussi, D. Miller and R. Kondor, arXiv: 2006.04780

# Particle Attention Block




$$\Delta = \sqrt{(y_a - y_b)^2 + (\phi_a - \phi_b)^2},$$
$$k_{\mathrm{T}} = \min(p_{\mathrm{T},a}, p_{\mathrm{T},b})\Delta,$$
$$z = \min(p_{\mathrm{T},a}, p_{\mathrm{T},b})/(p_{\mathrm{T},a} + p_{\mathrm{T},b}),$$
$$m^2 = (E_a + E_b)^2 - \|\mathbf{p}_a + \mathbf{p}_b\|^2,$$

*and many other possible pairwise features…*

*Injection of (physics-inspired) pairwise features to "bias" the dot-product self-attention*