Graph Neural Networks for Particle Physics

Part I

曲慧麟 (CERN)

第九届华大QCD讲习班

2021-10-13

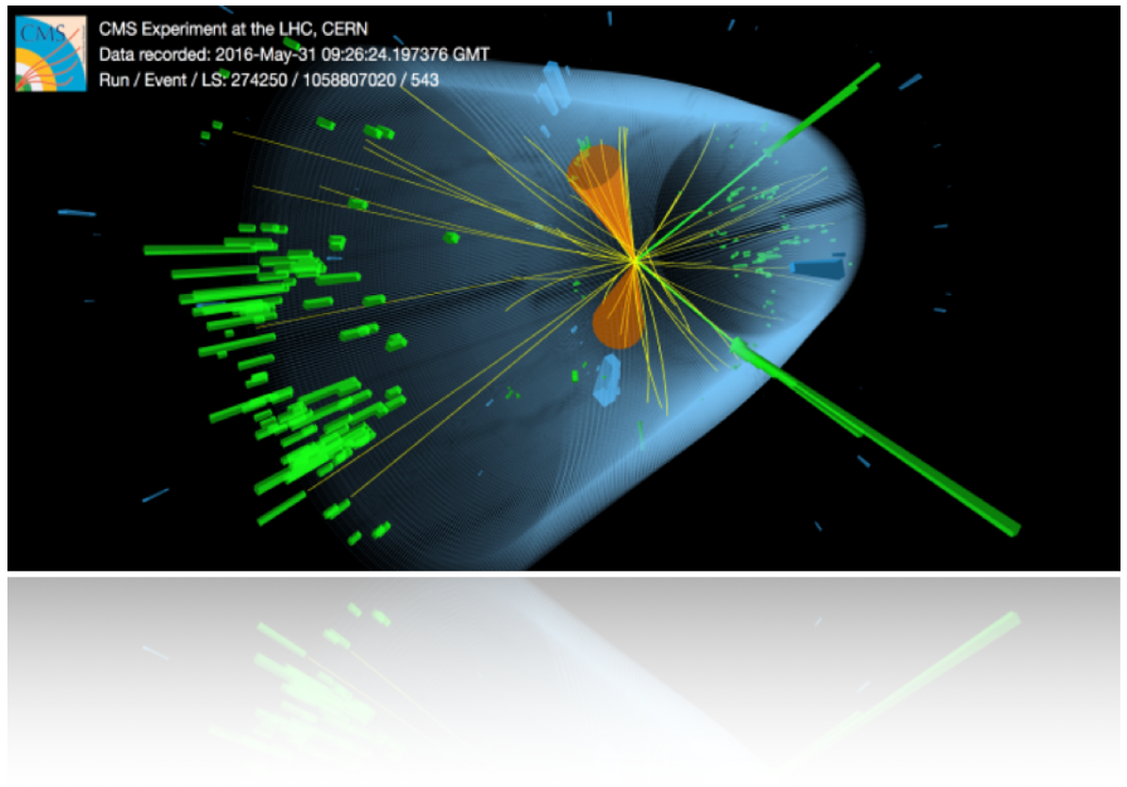
OUTLINE

- Part I (today)
  - Motivation: How to represent HEP data for machine learning?
  - Graph neural networks
  - Example applications in HEP

- Part II (tomorrow)
  - hands-on tutorial: jet tagging with GNNs
  - practicalities

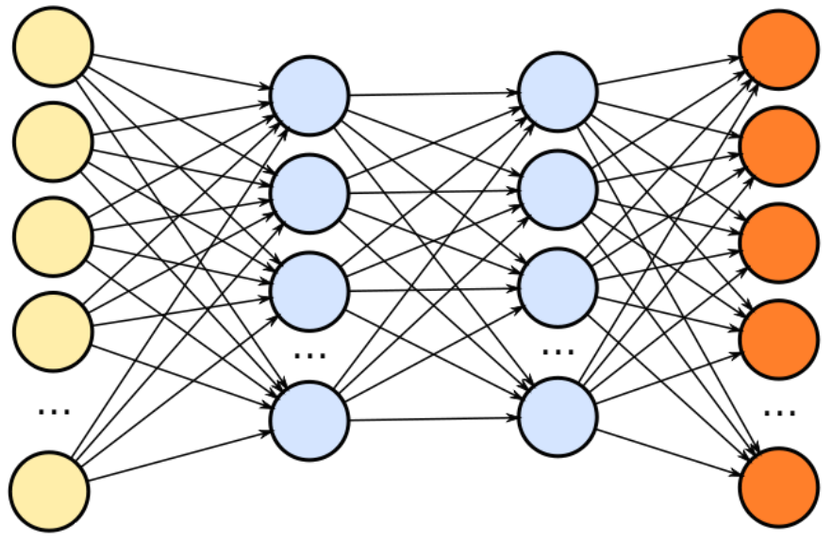
Motivation

HEP



×

ML

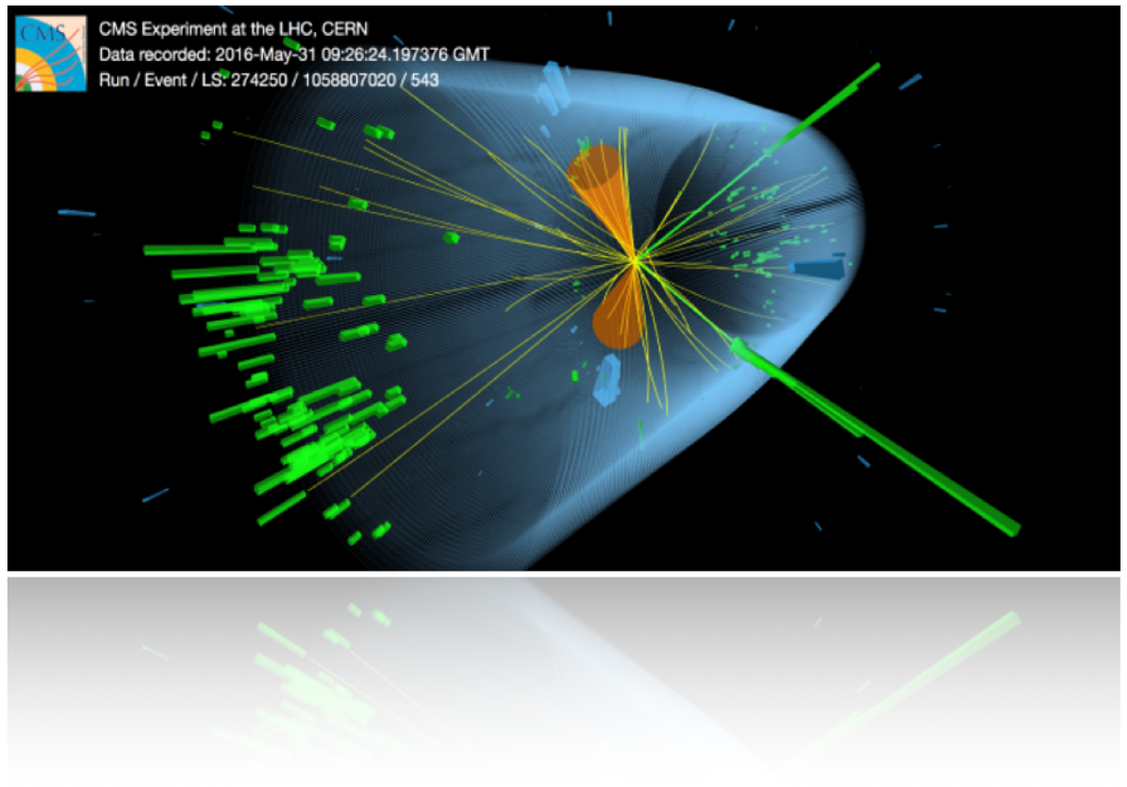


First and foremost:

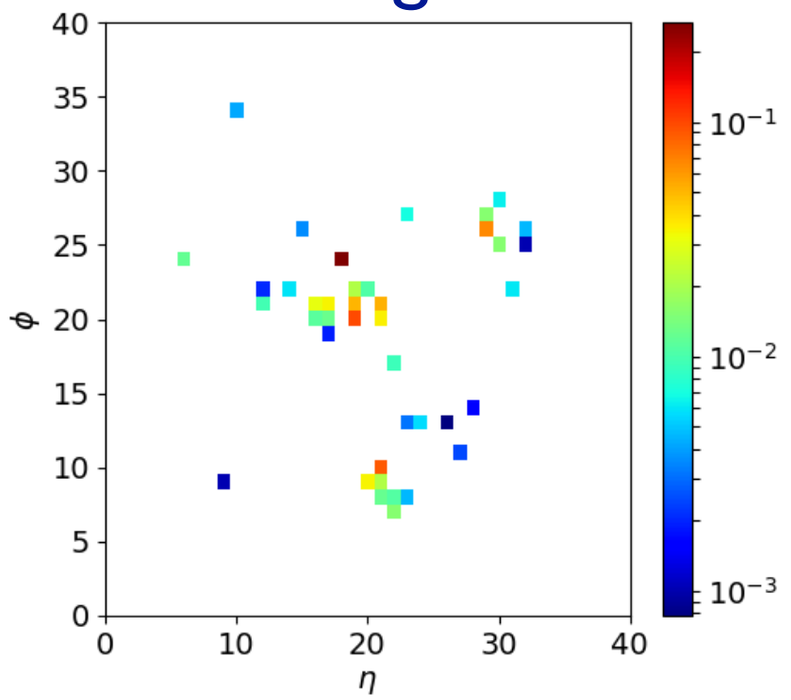
How to represent the data?

Data Representation: Image

HEP



Image

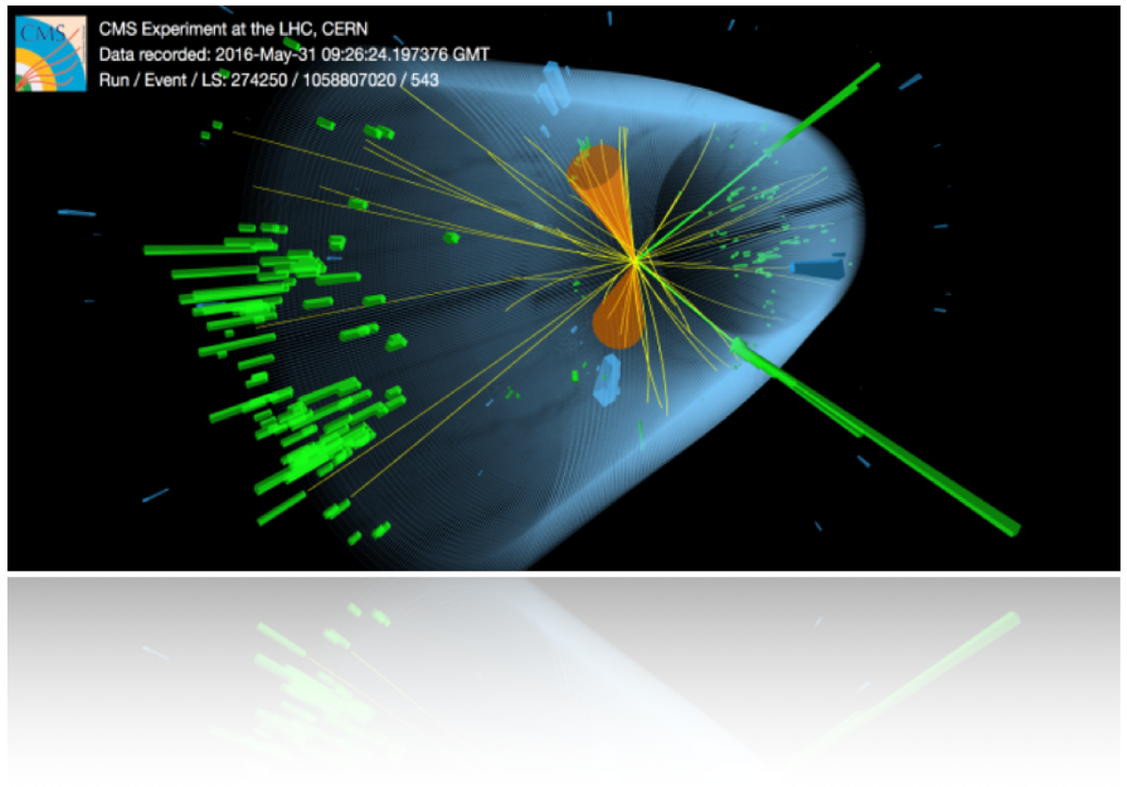


e.g., review in Kagan, arXiv:2012.09719

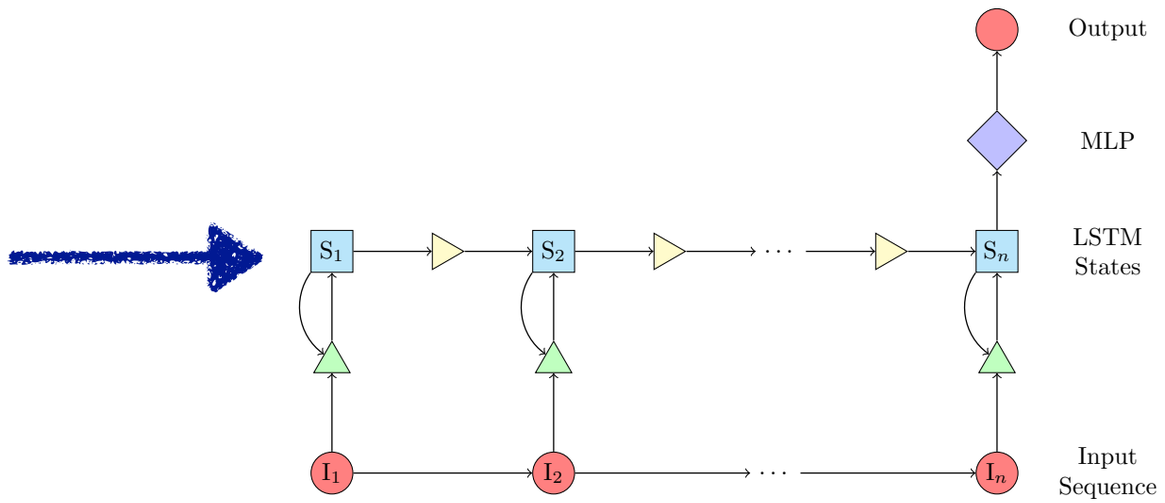
- Convert to 2D/3D image => *Computer vision*
  - then use convolutional neural networks
  - but:
    - inhomogeneous geometry
    - high sparsity

Data Representation: Sequence

HEP



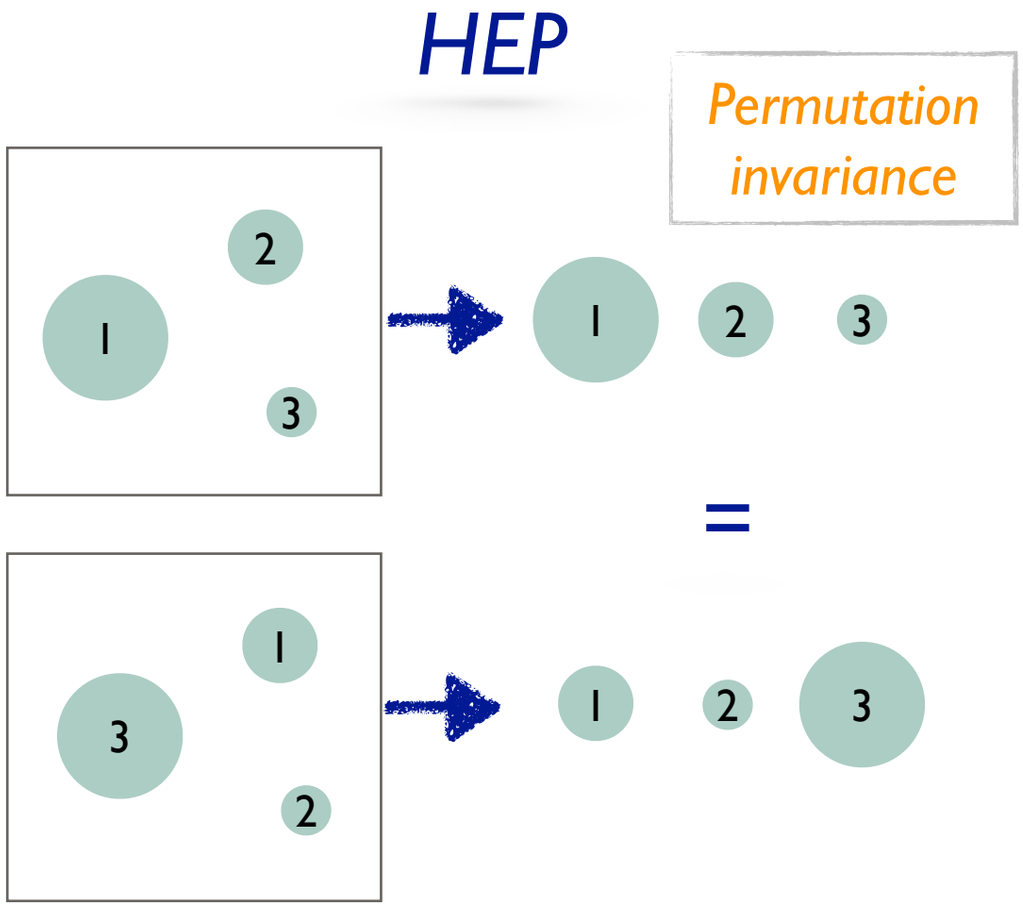
Sequence



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; Transformer

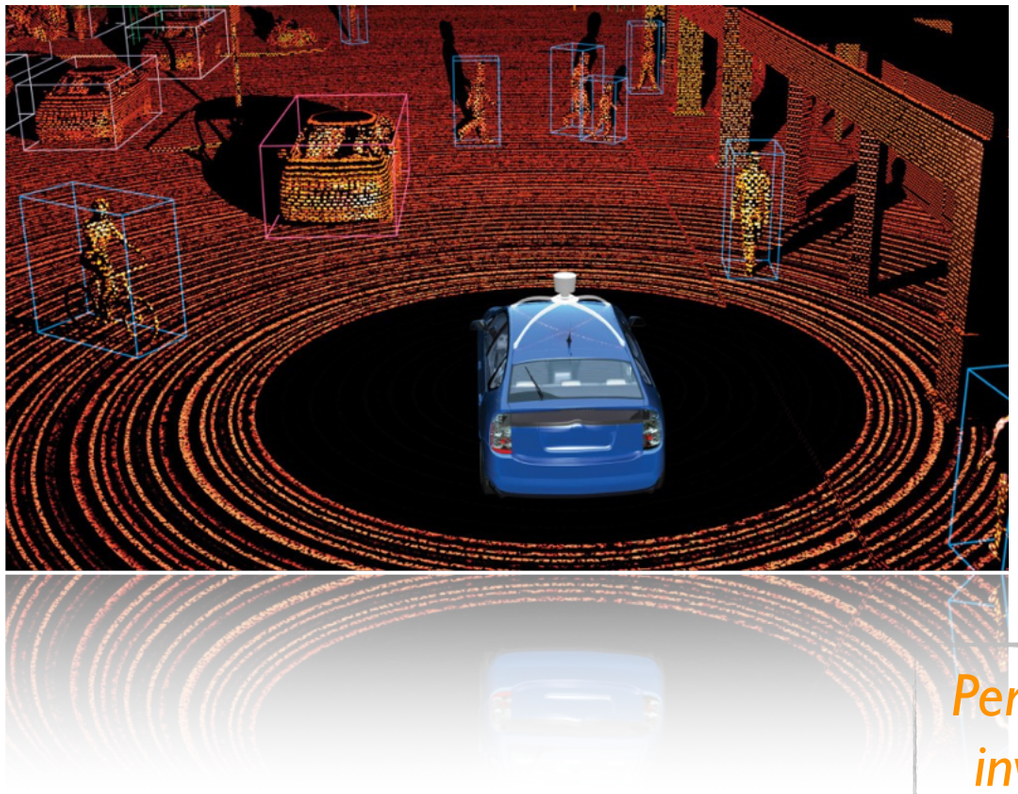
DATA REPRESENTATION: SEQUENCE



- Convert to a sequence => *Natural language processing (NLP)*
  - recurrent neural network (RNN), e.g., GRU/LSTM; Transformer
  - while words are naturally ordered in natural languages, particles are intrinsically ***unordered*** in a collision event
    - an ordering has to be ***imposed*** (pT, distance, …), which can limit the learning performance

Data Representation: Point Cloud

Point cloud

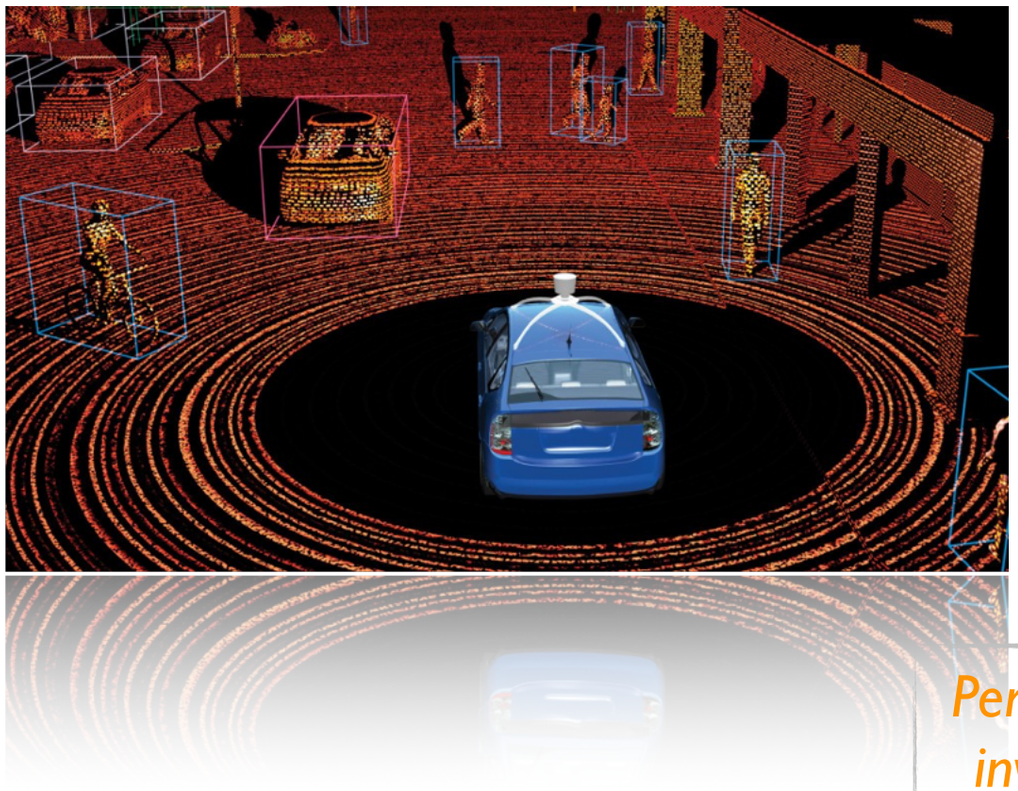


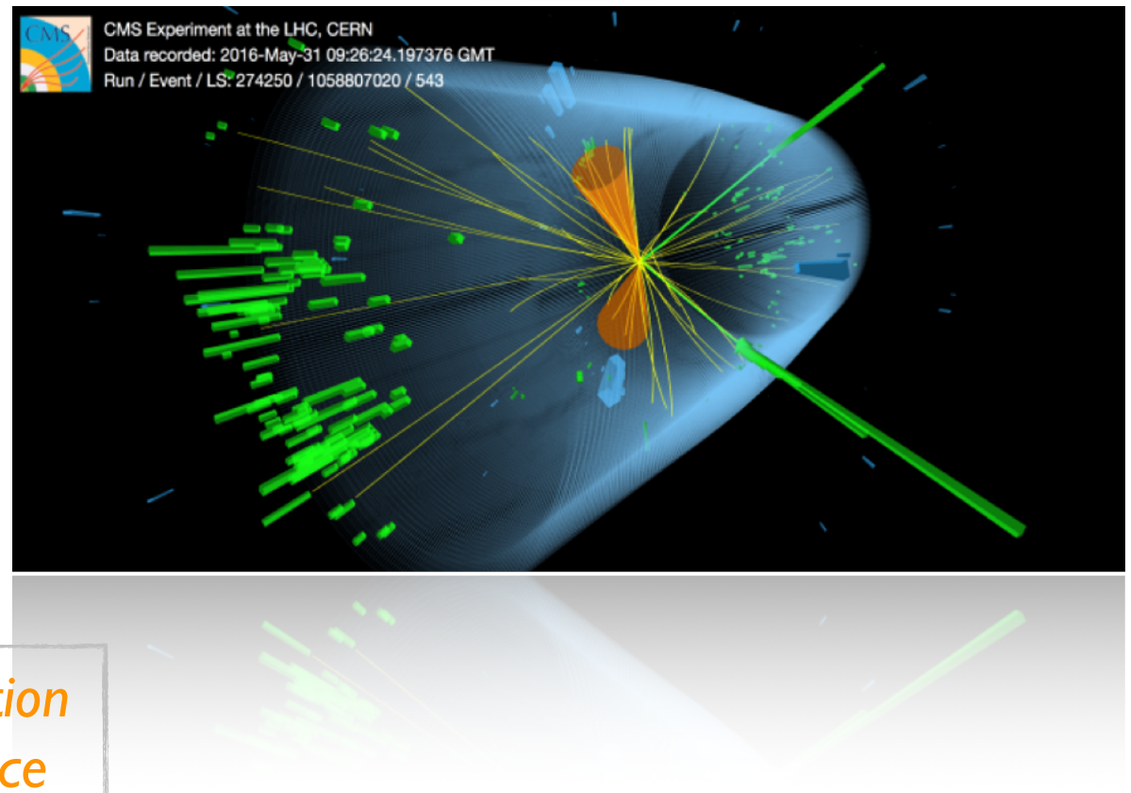
Permutation invariance

- Point cloud
  - an ***unordered set*** of points in space
    - typically produced by a LiDAR / 3D scanner
  - spatial distribution of the points => geometric structure of the objects

Data Representation: Point Cloud

Point cloud ↔ *HEP* *Particle cloud*



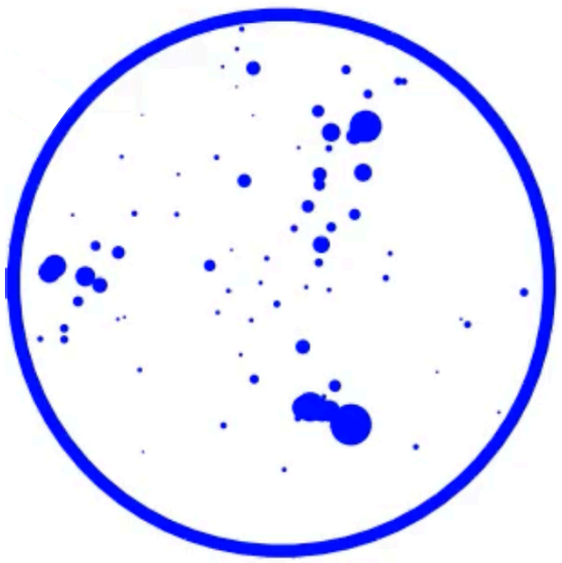


Permutation invariance

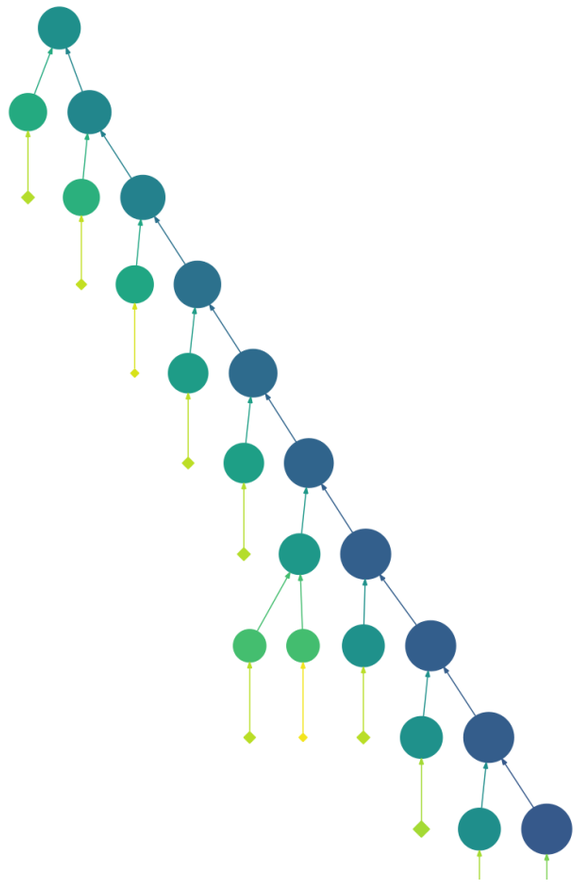
- Point cloud
  - an ***unordered set*** of points in space
    - typically produced by a LiDAR / 3D scanner
  - spatial distribution of the points => geometric structure of the objects
- Collision event: ***Particle cloud***
  - an ***unordered set*** of **particles** in space
  - spatial distribution of the particles => energy flow, (jet) substructure, …
  - but also richer information per particle:
    - energy, momentum, tracking, particle ID, …

Learning on Point Clouds

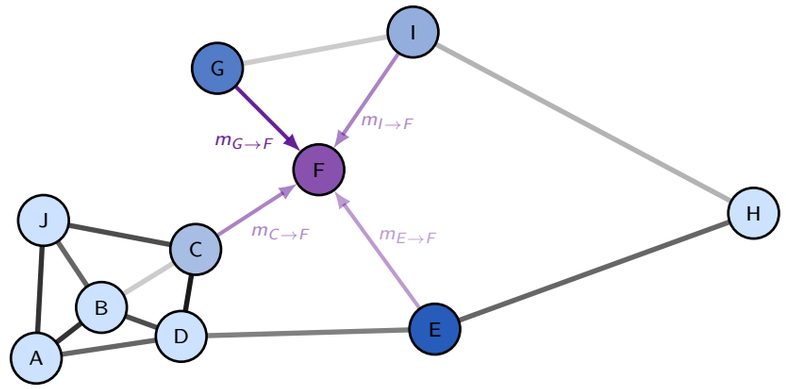
Set



Tree



Graph

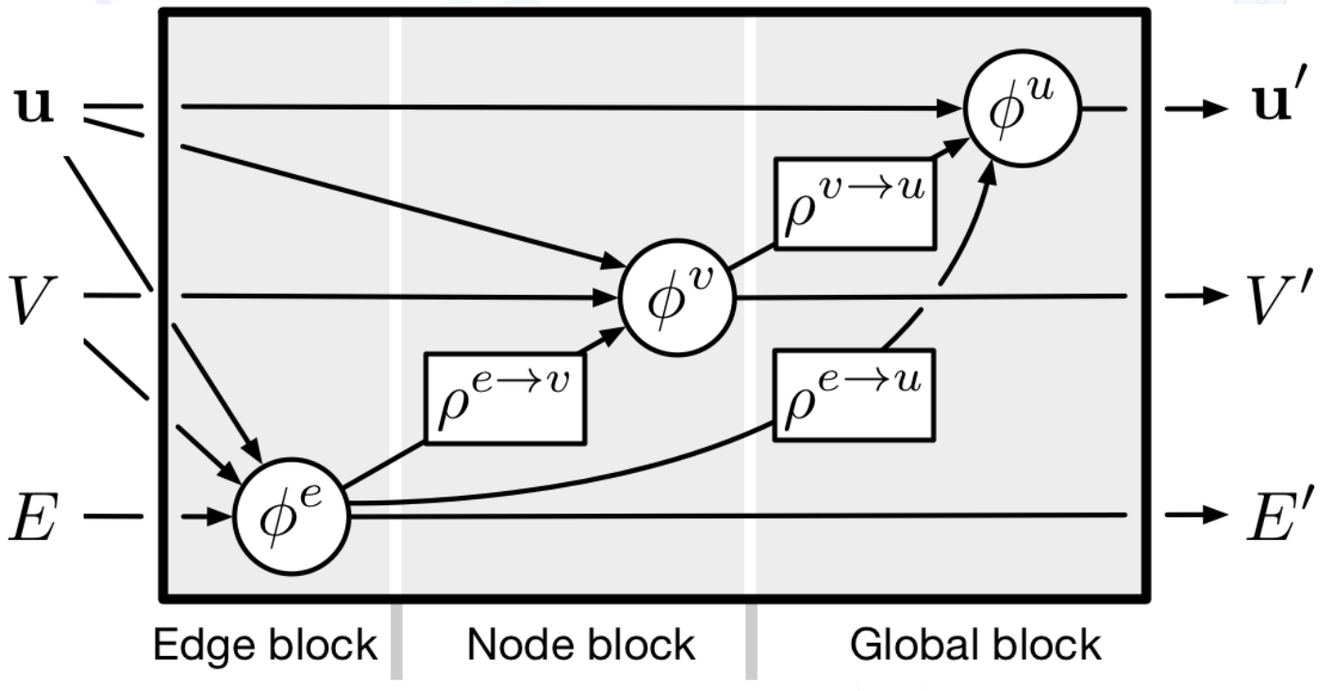


And more…

Learning on Point Clouds

Set *Tree* *Graph*

Graph neural network - A unified framework



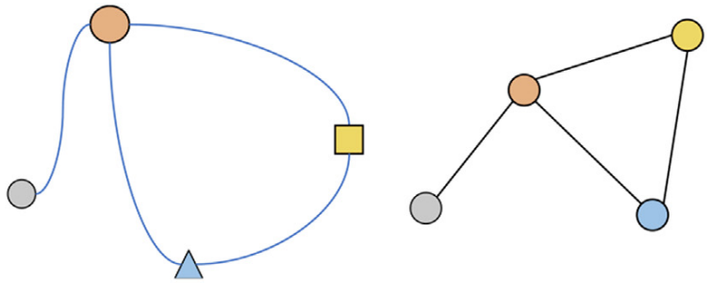
Review in Shlomi, Battaglia, Vlimant, arXiv:2007.13681

And more…

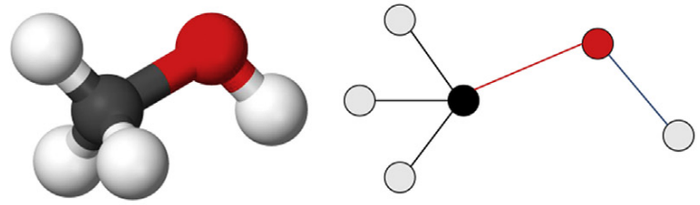
Graph Neural Networks (GNNs)

Based on Shlomi, Battaglia, Vlimant, arXiv:2007.13681

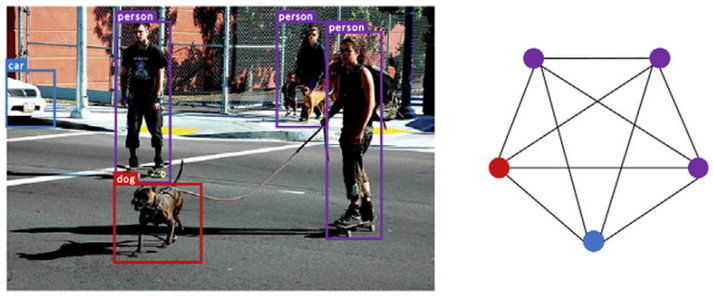
GNN Use Cases



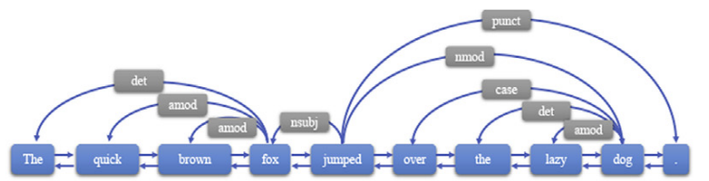
(a) Physics



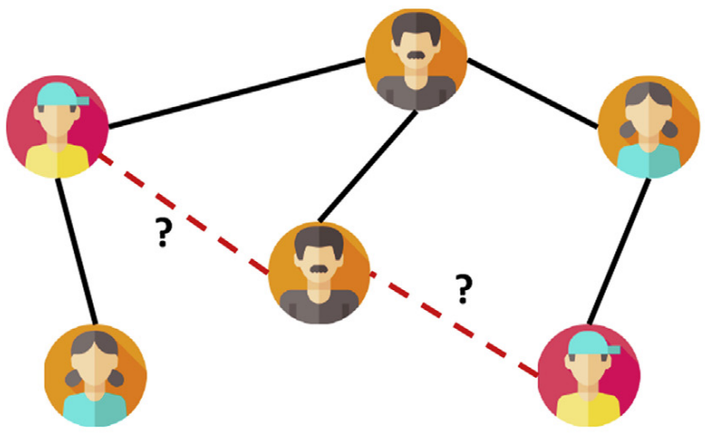
(b) Molecule



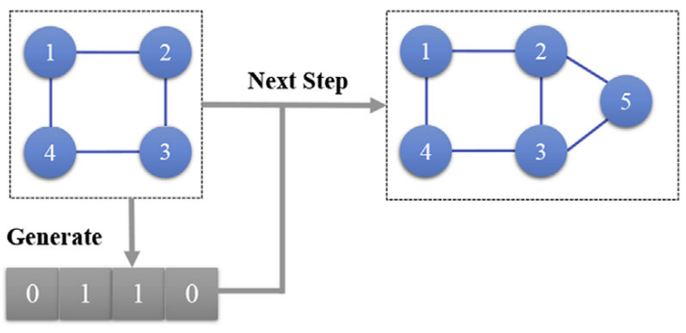
(c) Image



(d) Text



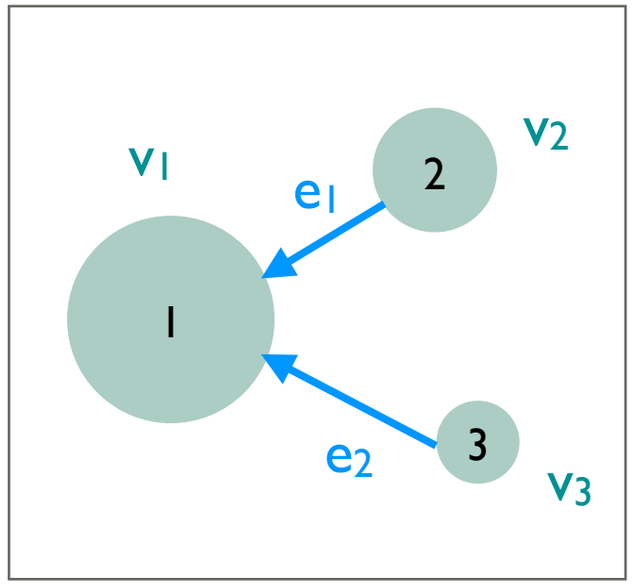
(e) Social Network



(f) Generation

e.g., review in Zhou, Cui, Hu, Zhang, Yang, Liu, Wang, Li, Sun, arXiv:1812.08434

GRAPH



graph level attributes

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

Vertices (nodes)

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

attributes of the i-th node

Edges (links)

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

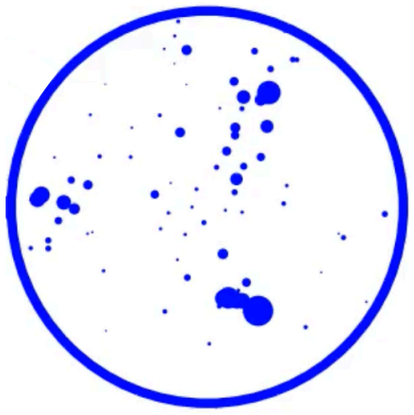
attributes of the k-th edge

indices of the two nodes (receiver and sender) connected by the k-th edge

CONSTRUCTING THE GRAPH

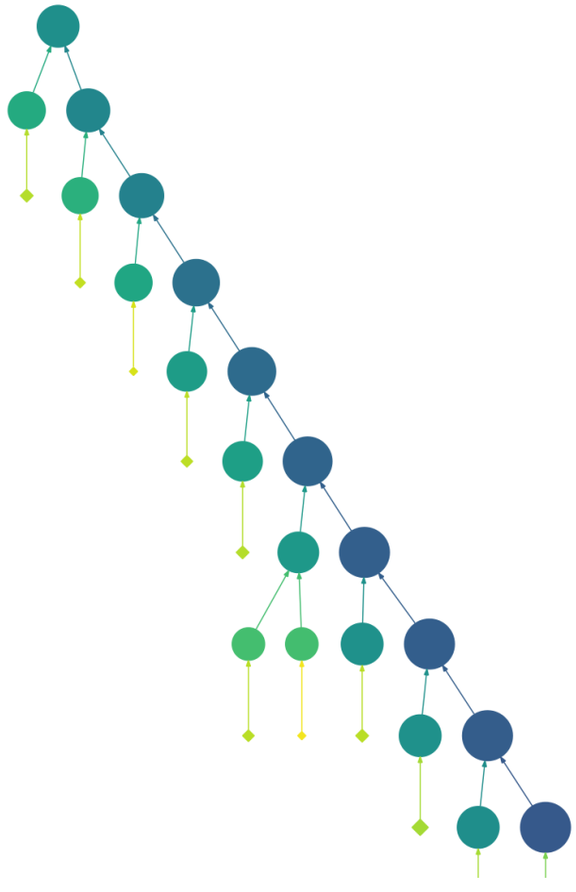
- From point/particle clouds to graphs:
  - points (particles) naturally become the **nodes** of the graph
  - but how to define the ***edges***?

Set: no edges



Hierarchical trees:

- *decay chain*
- *jet clustering history*



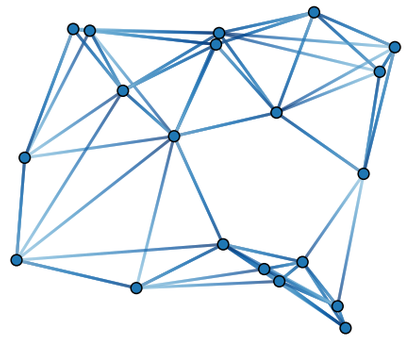
Fully connected graph

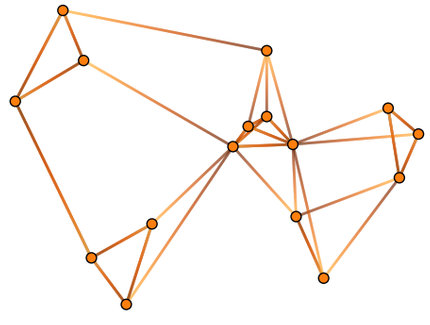
- *i.e., connect each node to all other nodes*

Locally connected graph

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

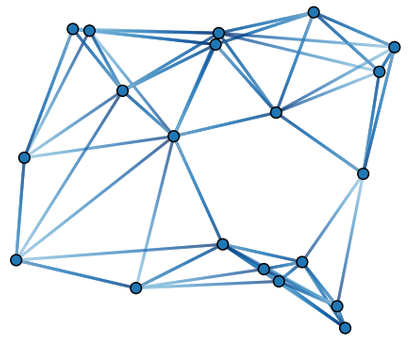
static





(dynamically) learned

CONSTRUCTING THE GRAPH



- From point/particle clouds to graphs:
  - points (particles) 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

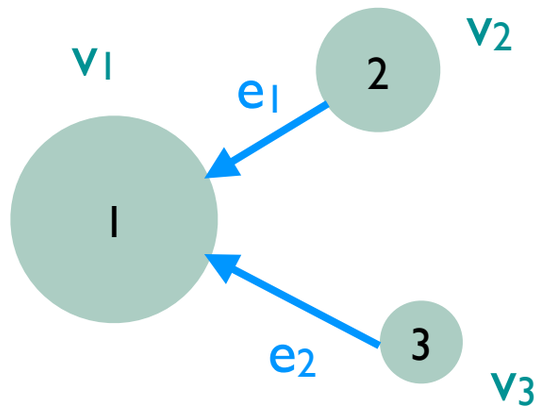
Graph: $G = (\mathbf{u}, V, E)$ with N_v vertices and N_e edges

Vertices (nodes)

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

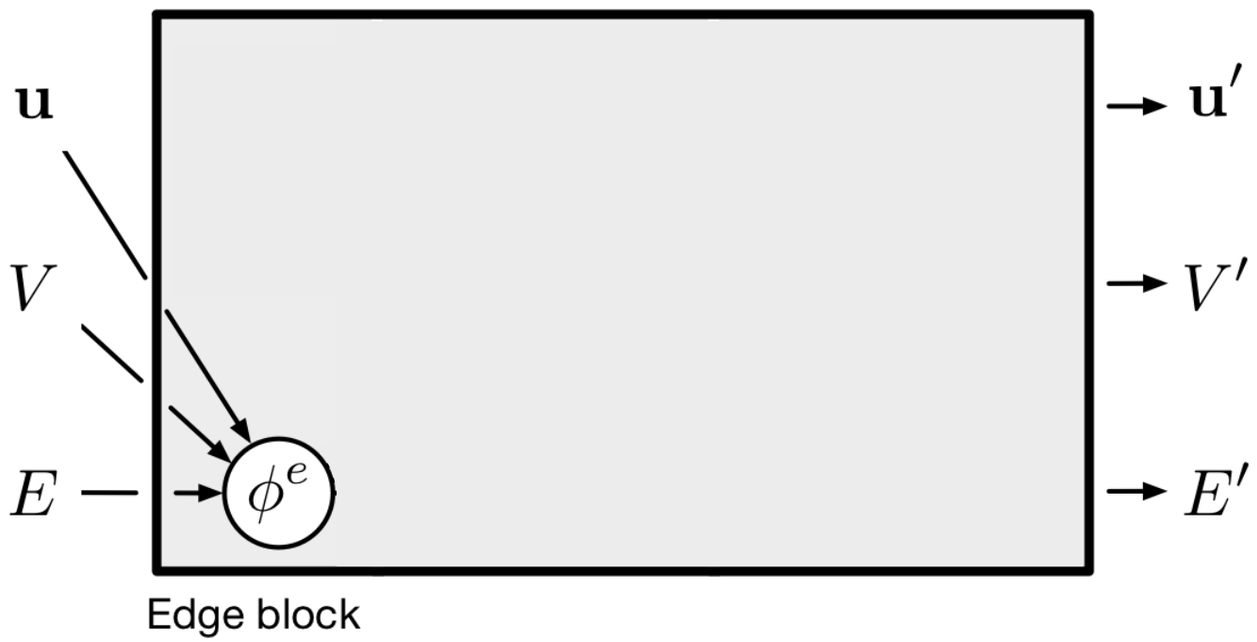
Edges (links)

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



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

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



Graph Network Formalism

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

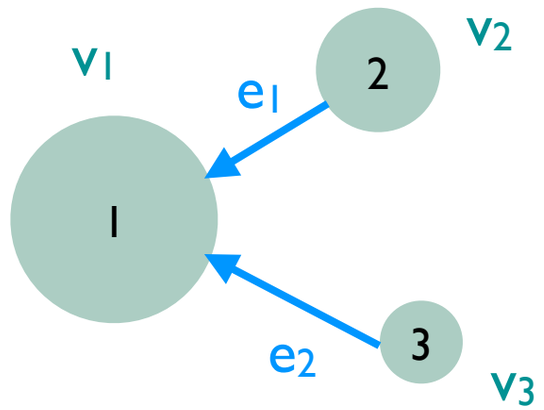
Graph: $G = (\mathbf{u}, V, E)$ with N_v vertices and N_e edges

Vertices (nodes)

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

Edges (links)

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

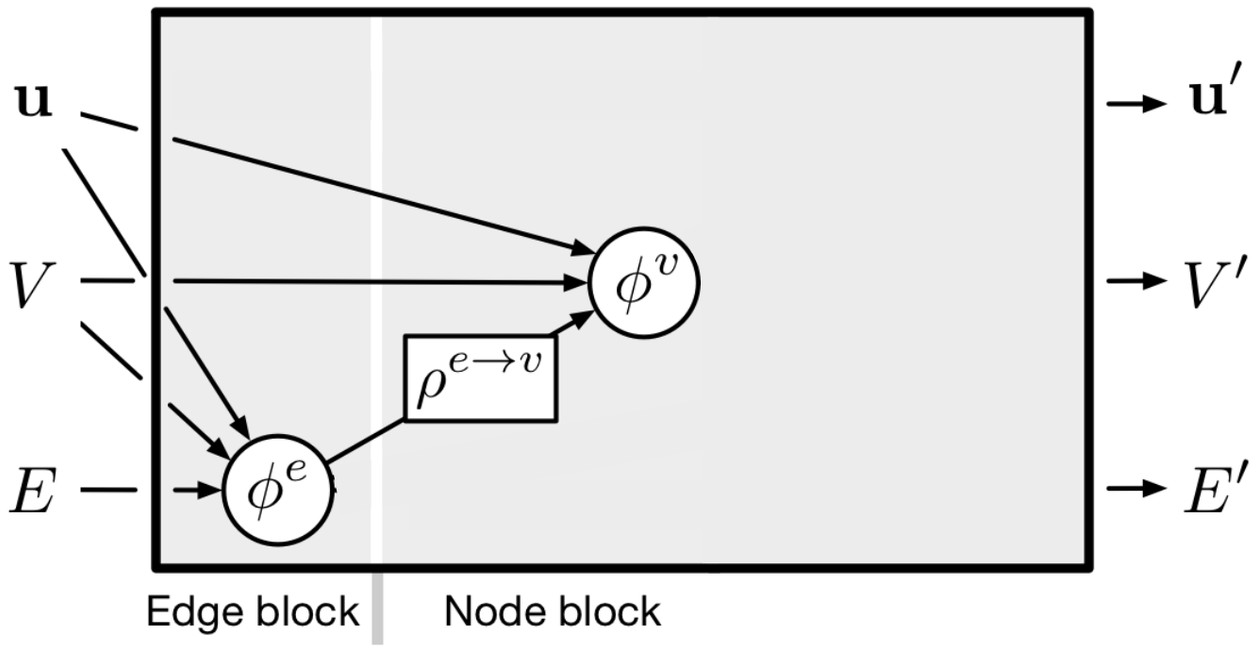


e'_k: message computed for edge k connecting nodes r_k, s_k

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

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

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



Graph Network Formalism

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

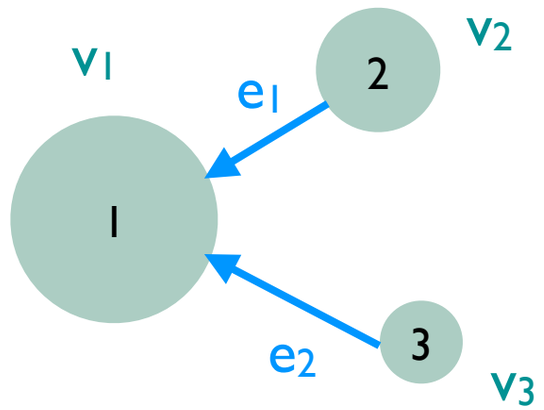
Graph: $G = (\mathbf{u}, V, E)$ with N_v vertices and N_e edges

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

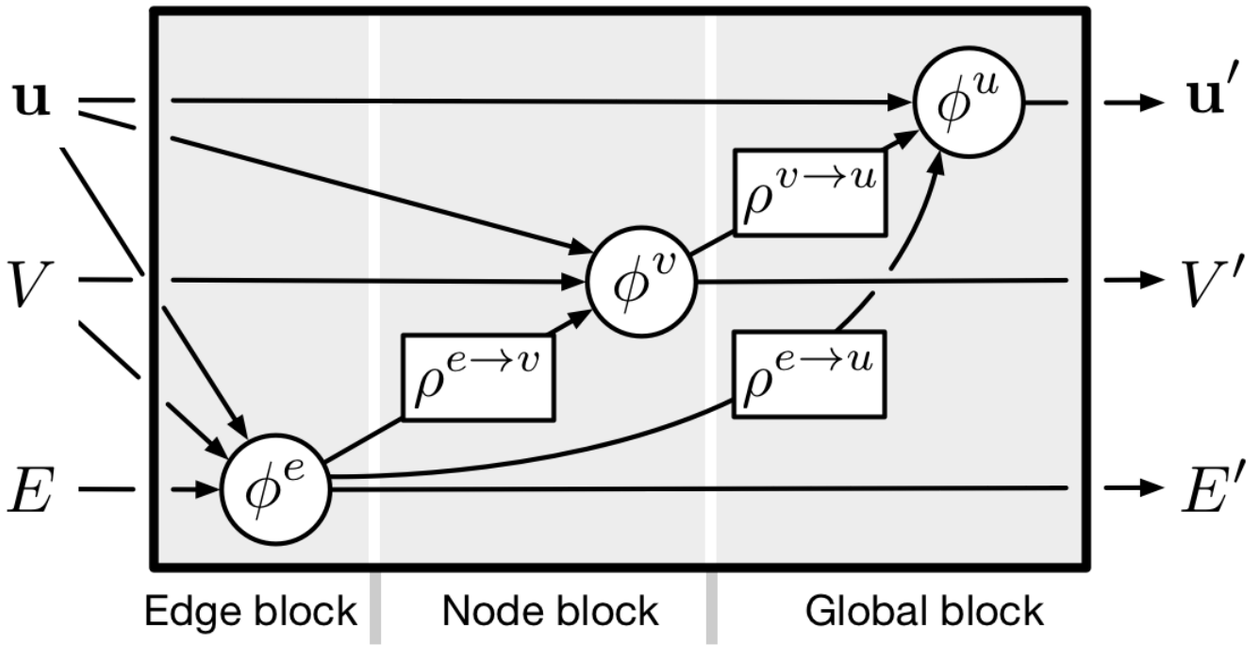
v'_i: node feature update based on aggregated messages and previous features

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

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

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

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



Graph Network Formalism

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

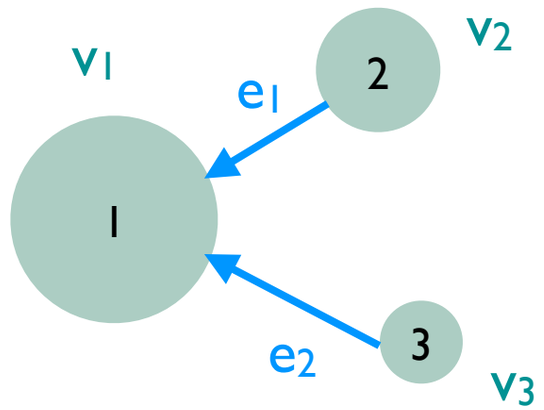
Graph: $G = (\mathbf{u}, V, E)$ with N_v vertices and N_e edges

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

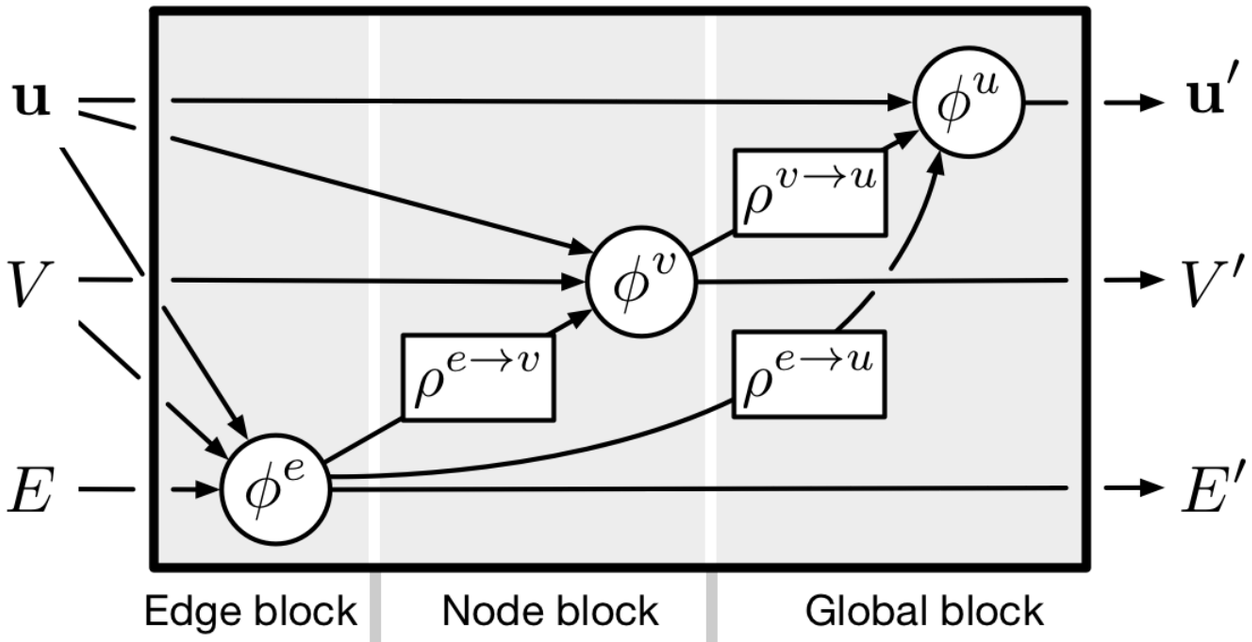
v'_i: node feature update based on aggregated messages and previous features

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

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

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

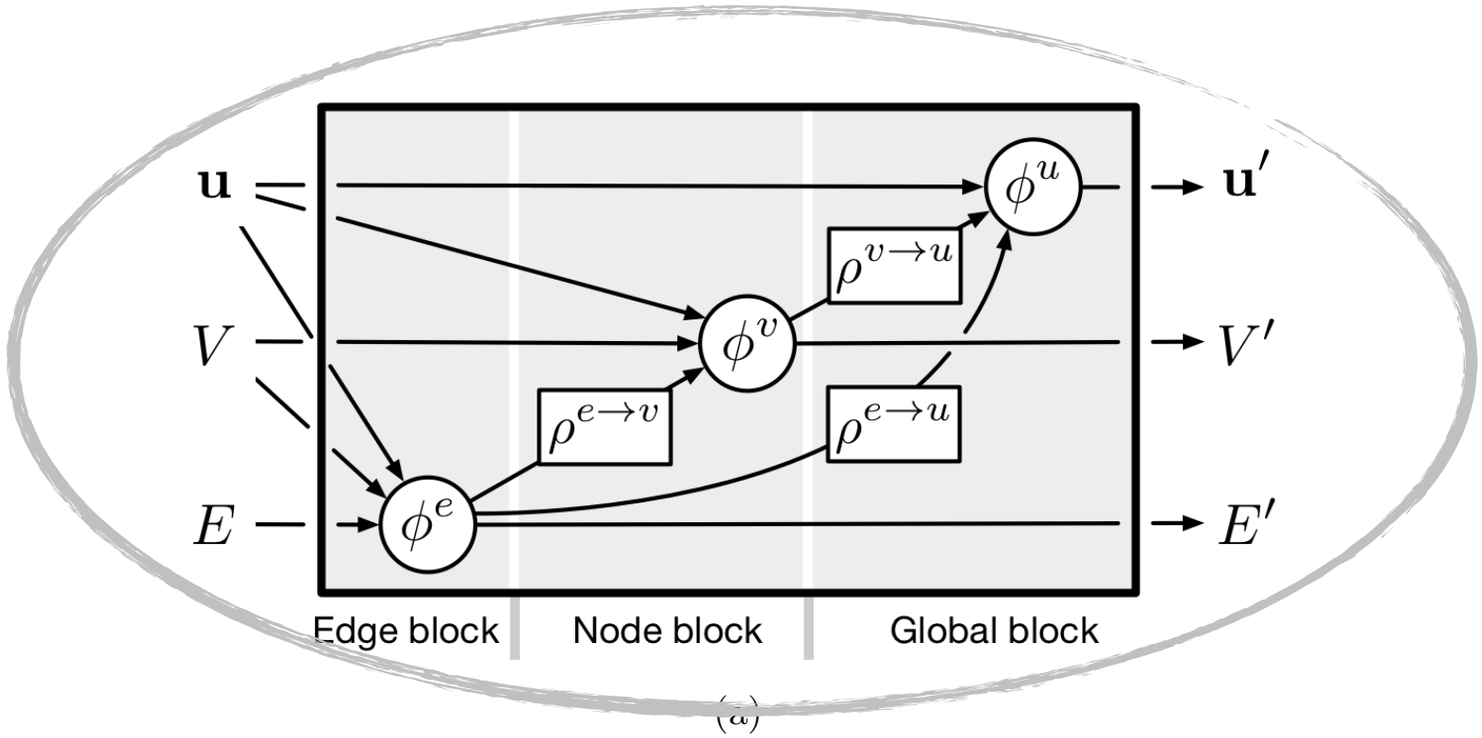
$$\boldsymbol{u}' = \phi^u(\bar{\boldsymbol{e}}', \bar{\boldsymbol{v}}', \boldsymbol{u}) \qquad \bar{\boldsymbol{v}}' = \rho^{v\to u}(V')$$



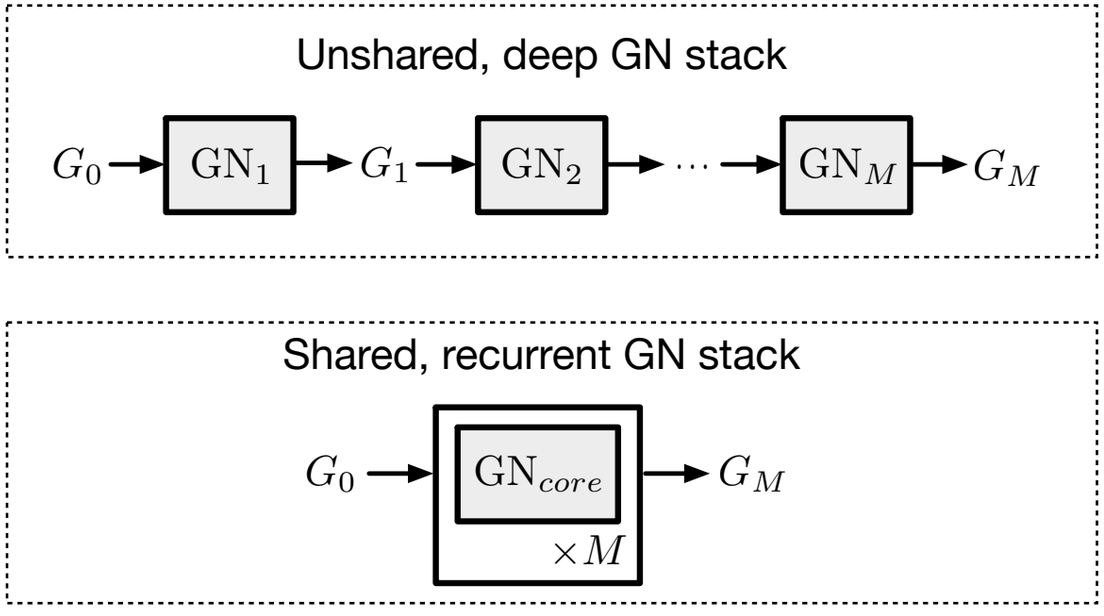
Shared-weight NN — *Symmetric functions (e.g., sum, mean, max, etc.)* → *Permutation invariance*

GRAPH NETWORK FORMALISM

GN layer

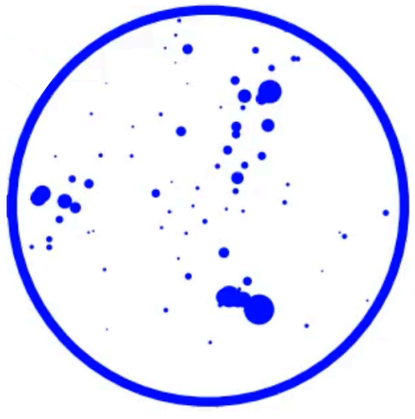


Full GNN



EXAMPLE: DEEP SETS

Set: no edges

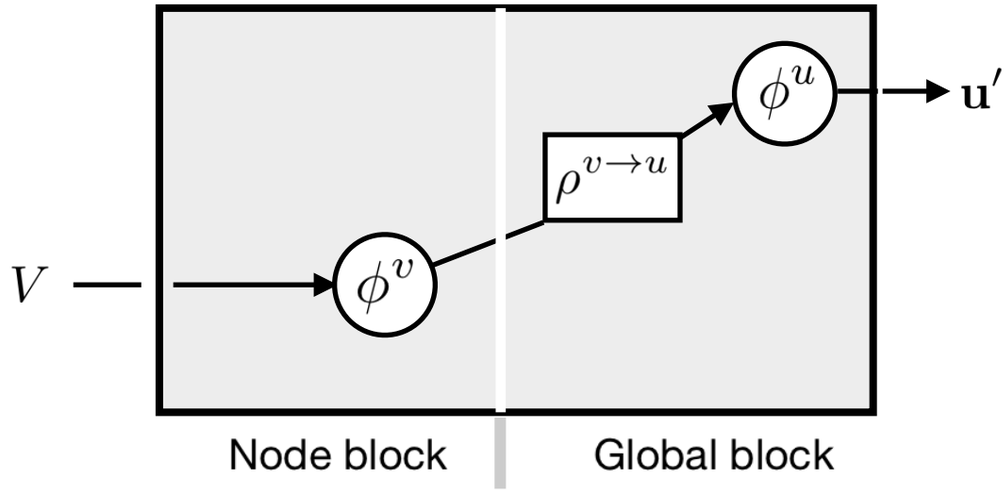


Deep Sets [1703.06114]

Manzil Zaheer[1,2], **Satwik Kottur**[1], **Siamak Ravanbhakhsh**[1],
Barnabás Póczos[1], **Ruslan Salakhutdinov**[1], **Alexander J Smola**[1,2]
[1] Carnegie Mellon University [2] Amazon Web Services

Deep Sets Theorem [63]. *Let $\mathfrak{X} \subset \mathbb{R}^d$ be compact, $X \subset 2^{\mathfrak{X}}$ be the space of sets with bounded cardinality of elements in $\mathfrak{X}$, and $Y \subset \mathbb{R}$ be a bounded interval. Consider a continuous function $f : X \to Y$ that is invariant under permutations of its inputs, i.e. $f(x_1, \ldots, x_M) = f(x_{\pi(1)}, \ldots, x_{\pi(M)})$ for all $x_i \in \mathfrak{X}$ and $\pi \in S_M$. Then there exists a sufficiently large integer ℓ and continuous functions $\Phi : \mathfrak{X} \to \mathbb{R}^\ell$, $F : \mathbb{R}^\ell \to Y$ such that the following holds to an arbitrarily good approximation:*[1]

$$f(\{x_1, \ldots, x_M\}) = F\left(\sum_{i=1}^{M} \Phi(x_i)\right)$$

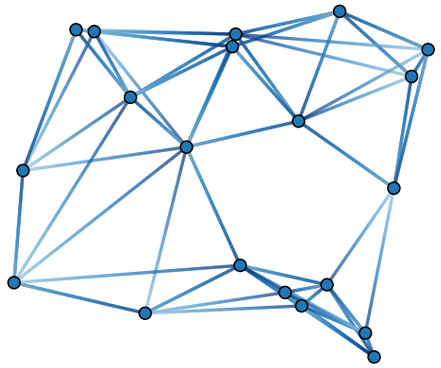


EXAMPLE: DYNAMIC GRAPH CNN

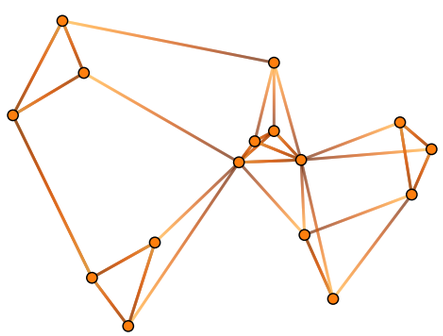
Dynamic *locally connected graph*

- *k-nearest neighbors*

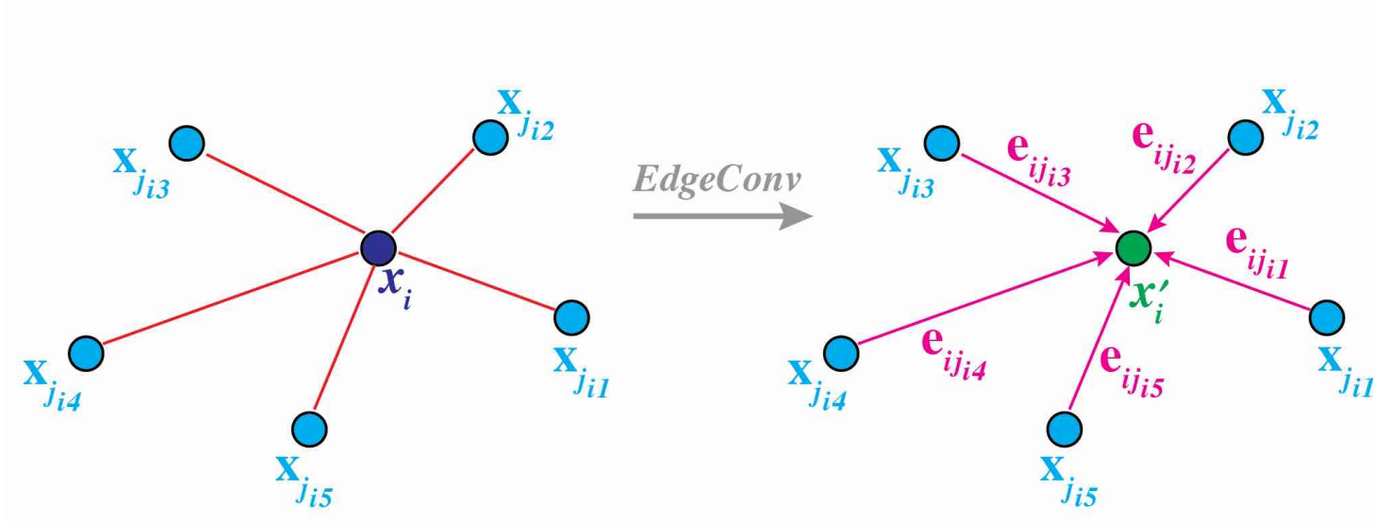
For the 1st layer:
kNN in input coordinates (xyz/η-φ)



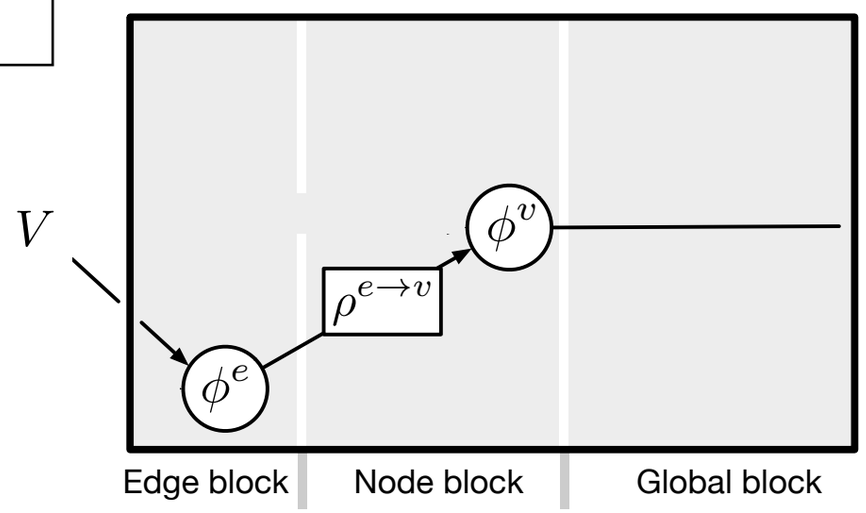
For subsequent layers:
kNN in learned latent space



Key building block: EdgeConv



Wang, Sun, Liu, Sarma, Bronstein, Solomon, arXiv:1801.07829

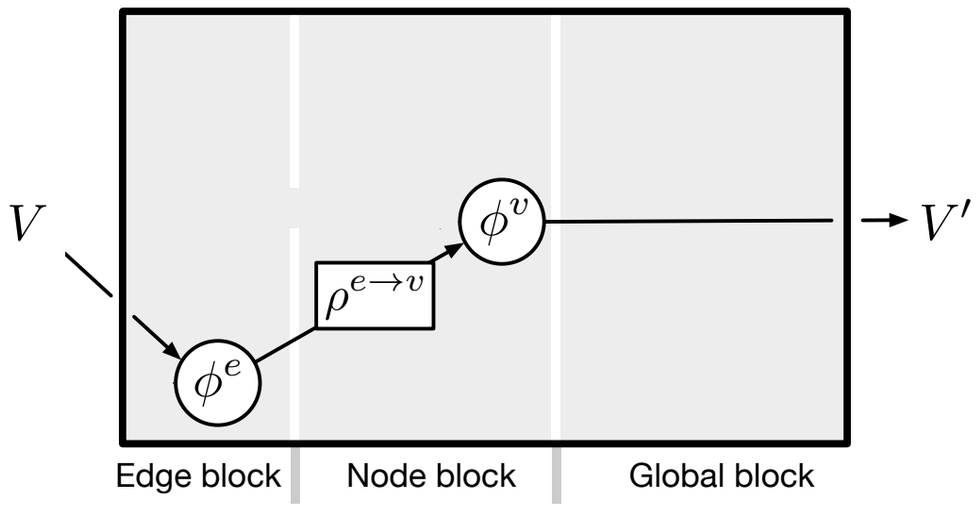


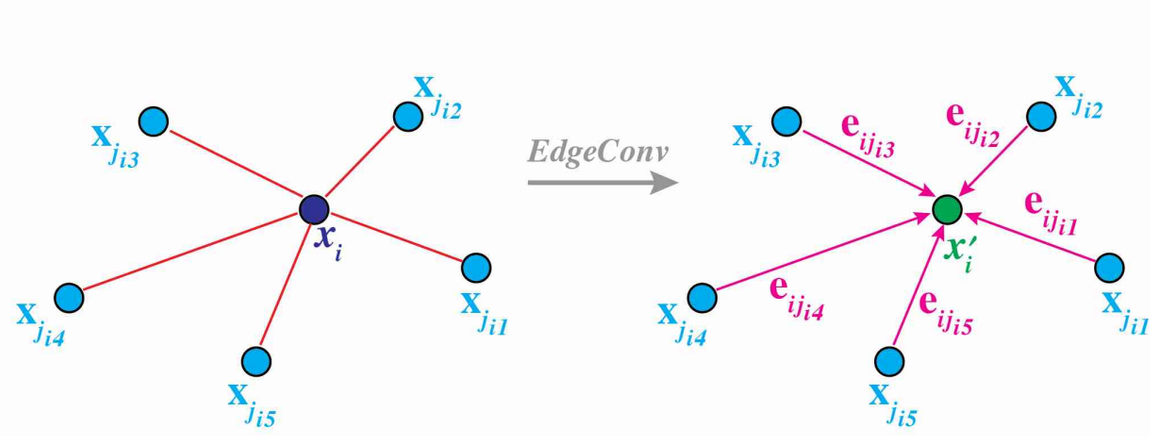
$$e'_{ij} = \phi^e(v_i, v_j) = \text{MLP}(v_i, v_j)$$

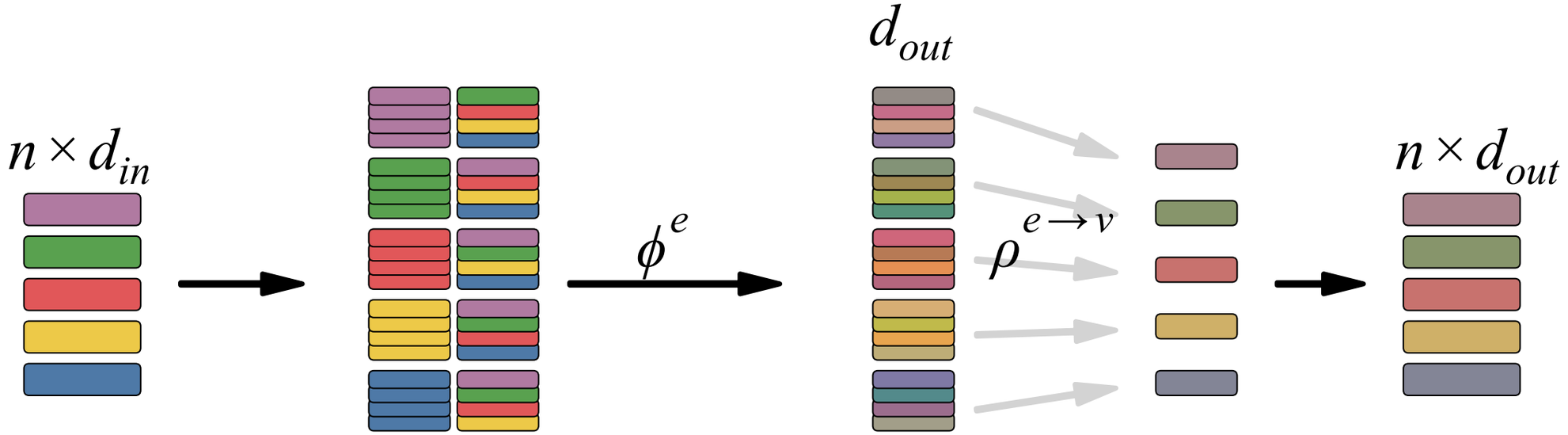
$$v'_i = \rho^{e\to v}(E'_i) = \square_j e'_{ij}$$

$\square$ = sum, mean, max, etc.

EXAMPLE: DYNAMIC GRAPH CNN

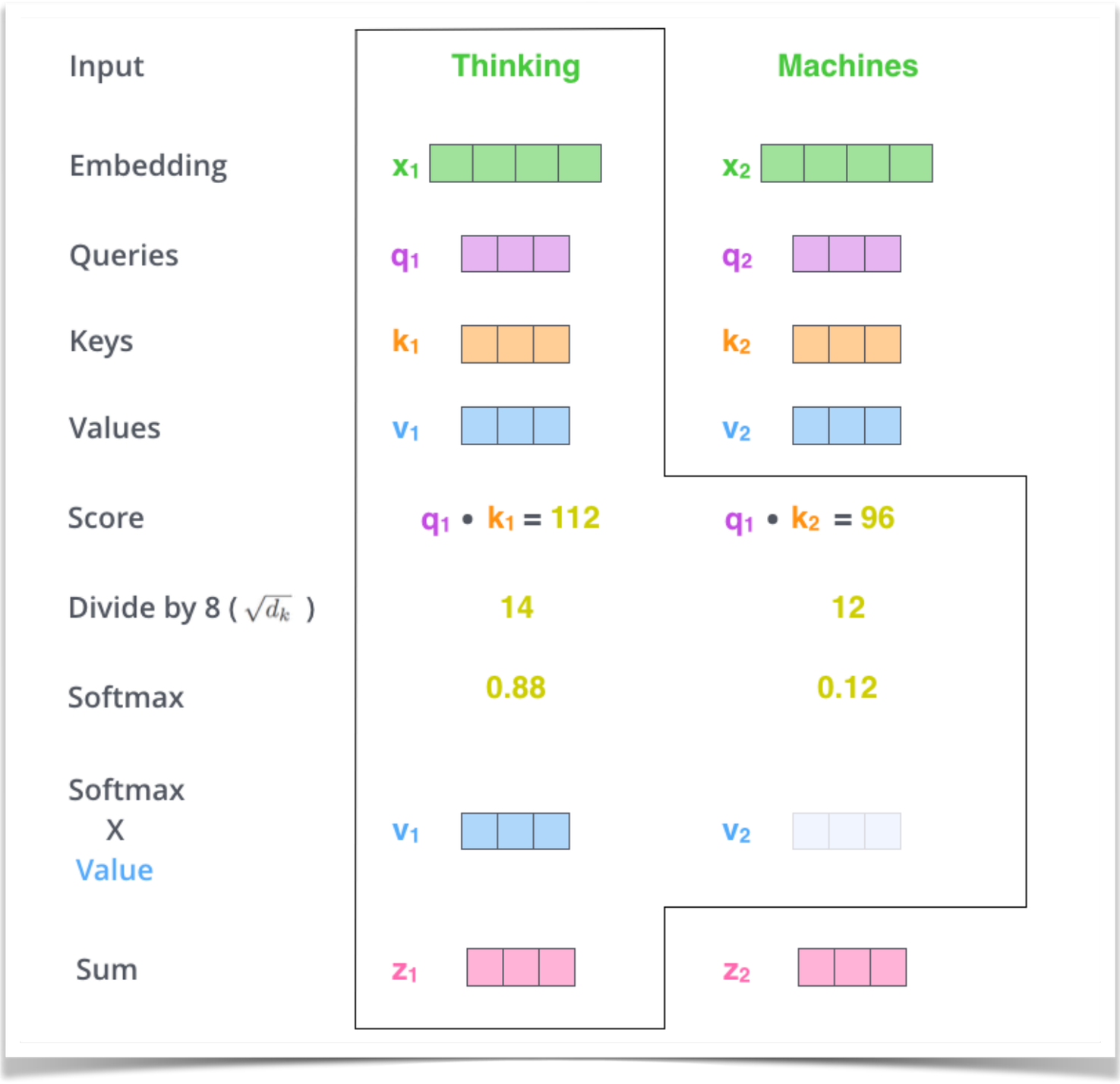


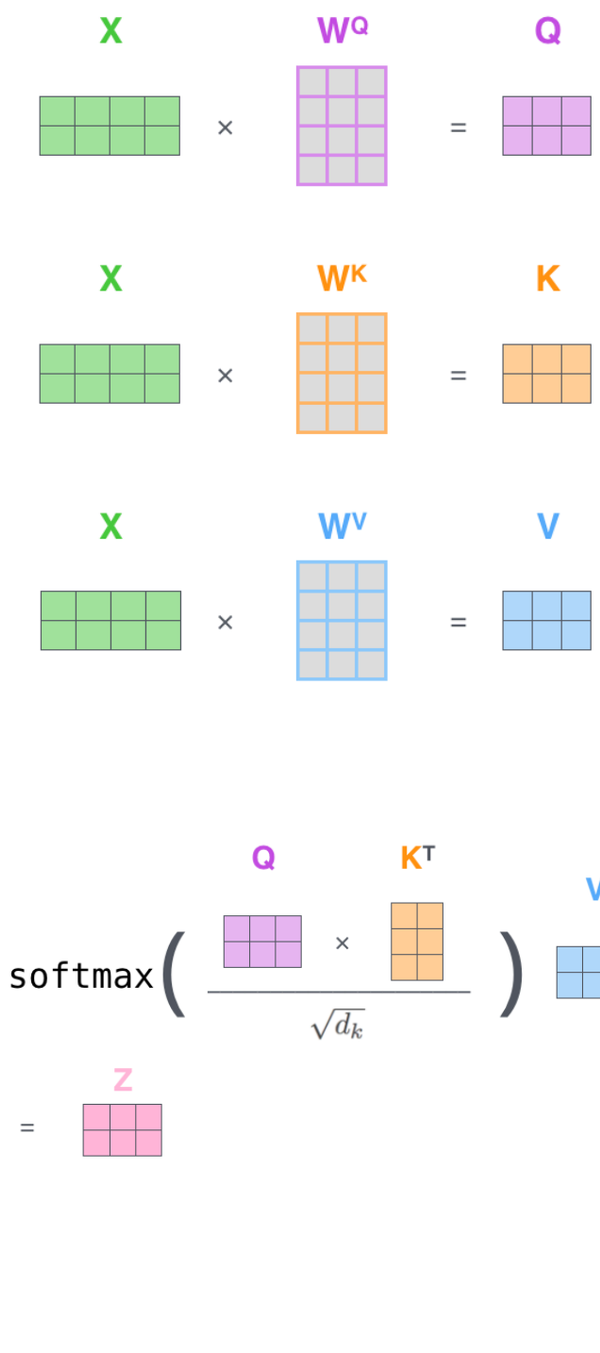




EXAMPLE: TRANSFORMER

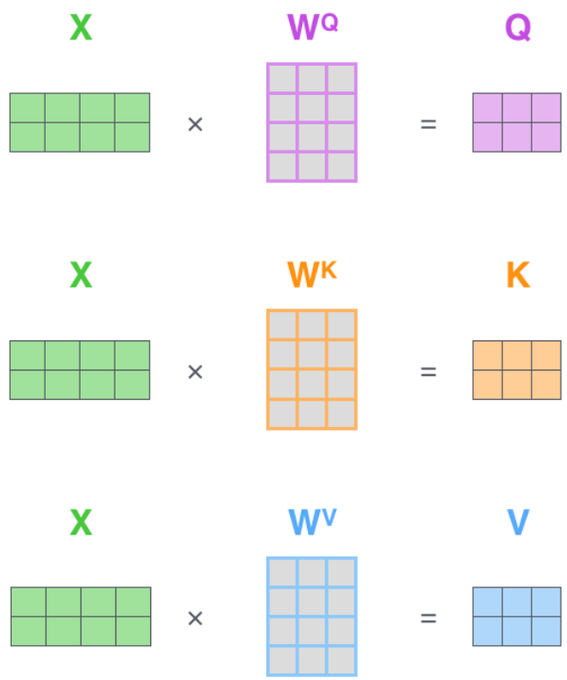
Vaswani, Shazeer, Parmar, Uszkoreit, Jones, Gomez, Kaiser, Polosukhin, arXiv:1706.03762

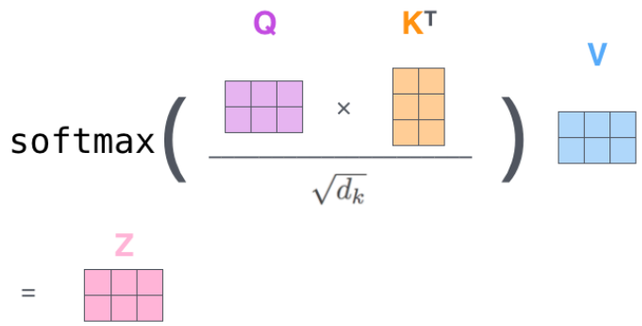




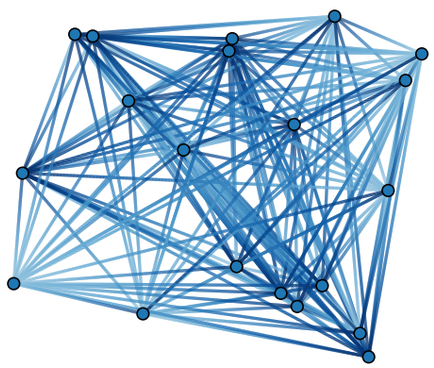
https://jalammar.github.io/illustrated-transformer/

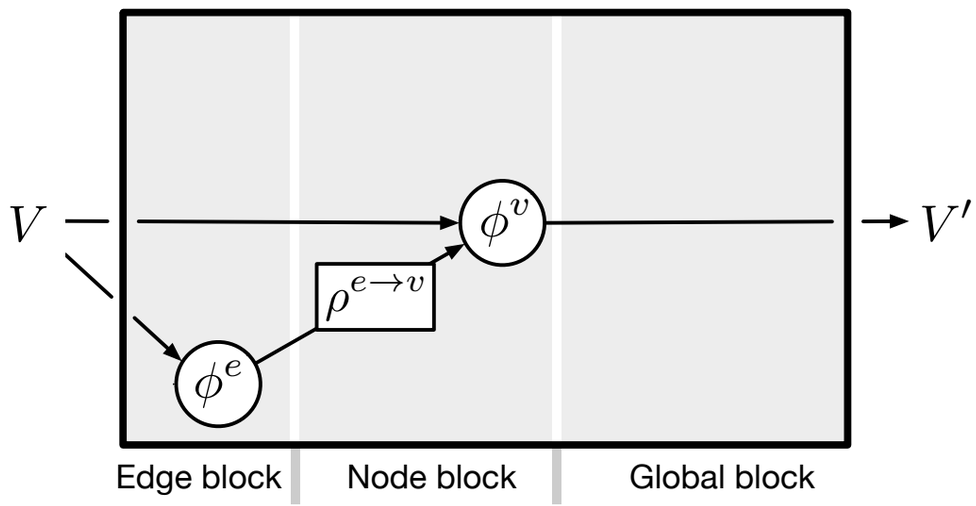
EXAMPLE: TRANSFORMER





Fully connected graph





$$e'_{ij} = \phi^e(v_i, v_j) = (W_Q v_i)^T (W_K v_j)$$

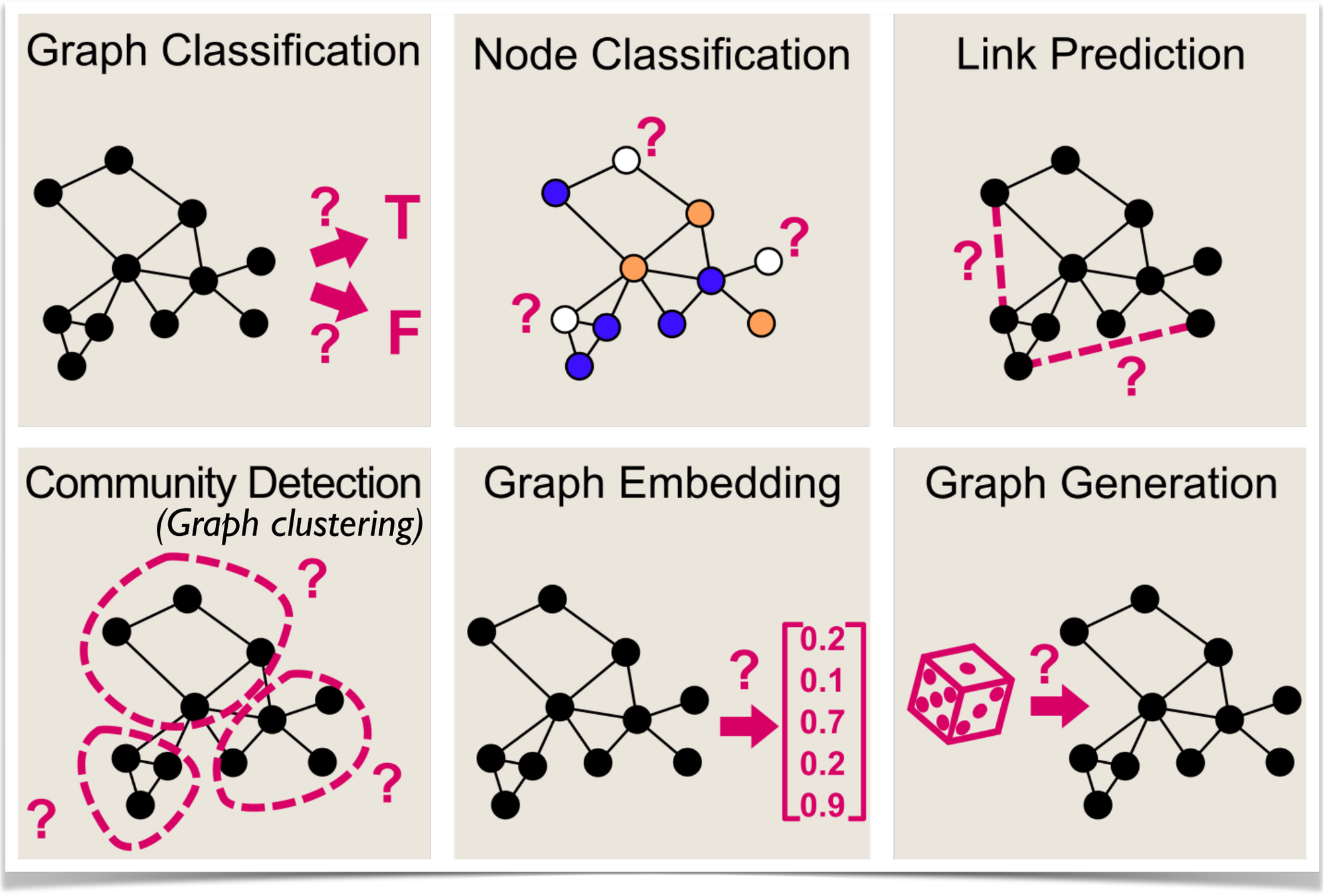
$$w_{ij} = \mathrm{softmax}_j \frac{e'_{ij}}{\sqrt{d_k}}$$

$$v'_i = \sum_j w_{ij} (W_V v_j)$$

- The transformer architecture is also permutation-invariant as long as positional encoding is not used

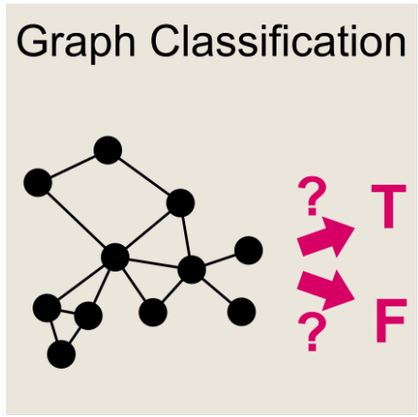
GNN Applications in HEP

GRAPH ML TASKS

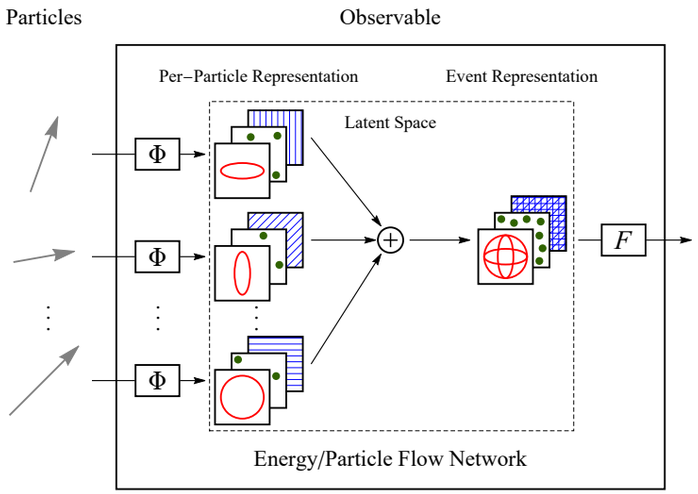


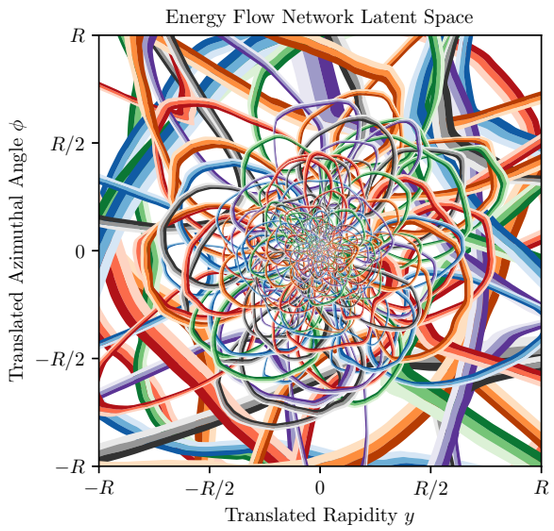
https://towardsdatascience.com/graph-convolutional-networks-deep-99d7fee5706f

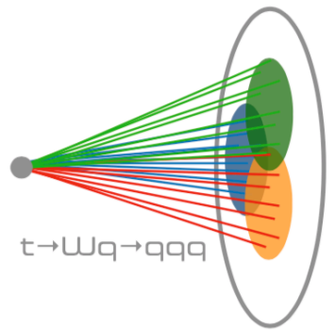
Jet Tagging

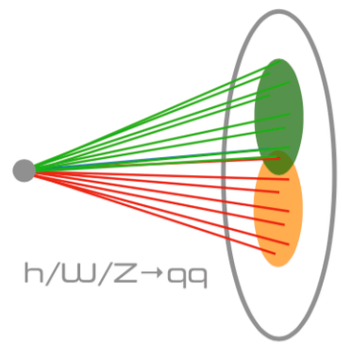


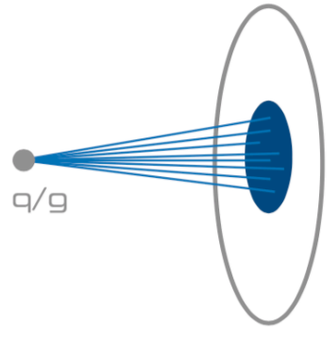
- Jet tagging: classifying jets from different origins
- Energy flow network [Komiske, Metodiev, Thaler, arXiv:1810.05165]
  - adapted from Deep Sets



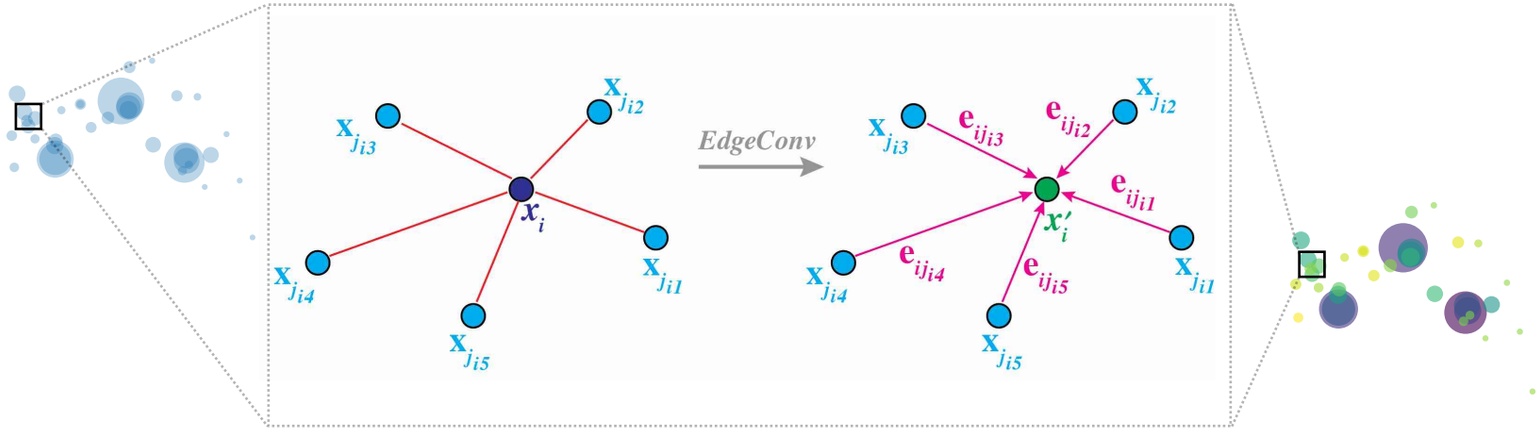








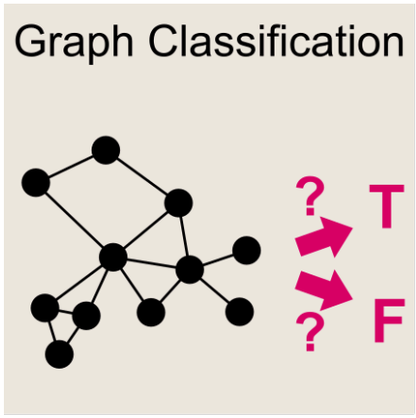
- ParticleNet [Qu, Gouskos, arXiv:1902.08570]
  - adapted from Dynamic Graph CNN – significant performance improvement



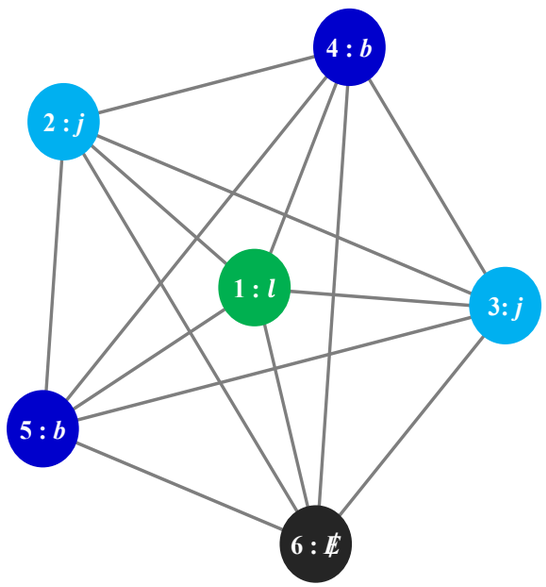
- More in tomorrow's session

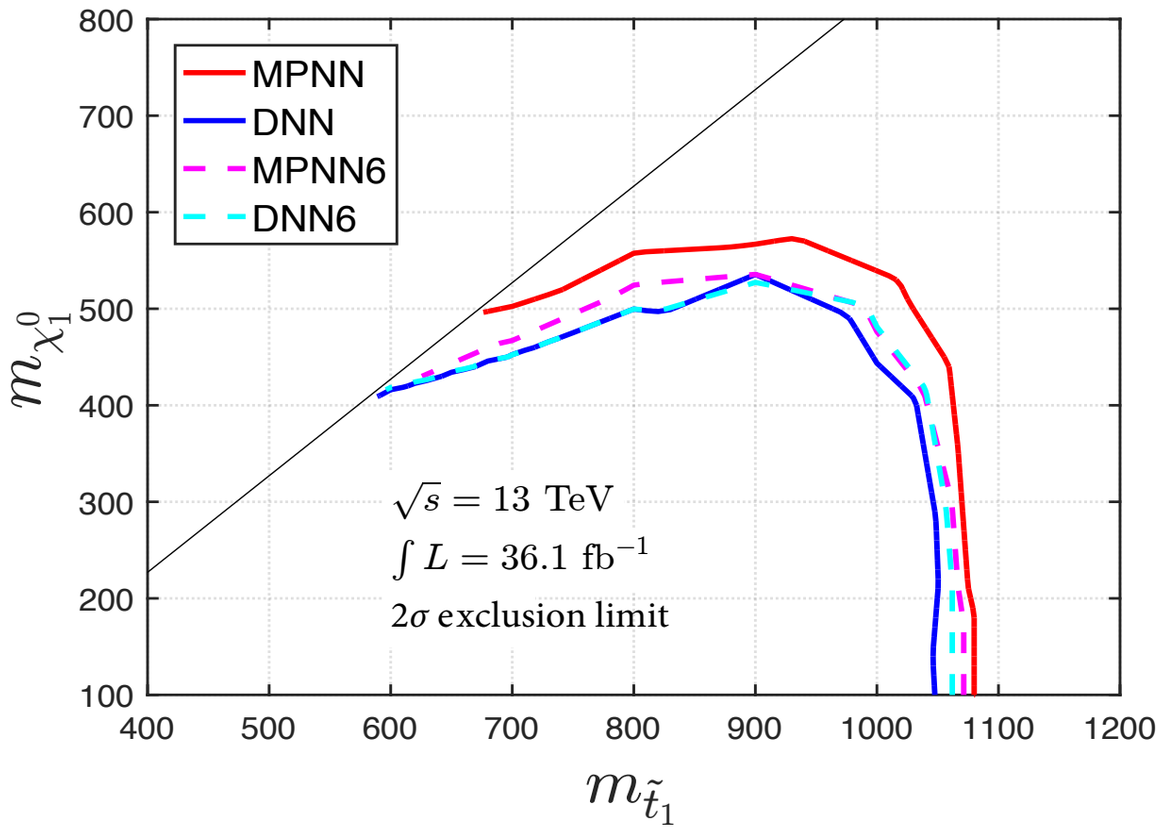
Image credit

EVENT CLASSIFICATION

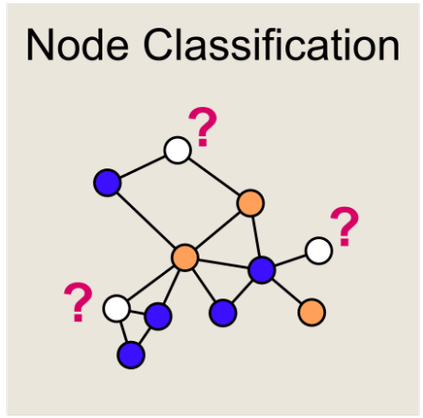


- Message passing neural network (MPNN) used to distinguish SUSY signal from background [Abdughani, Ren, Wu, Yang, arXiv: 1807.09088]
- Using all reconstructed objects in the event as input
  - leptons, jets, missing transverse momentum
  - fully connected graph structure
- Significant improvement compared to the baseline dense neural network (DNN)
- Similar approach also exploited to study the CP property of the top-Higgs coupling [Ren, Wu, Yang, arXiv: 1901.05627] and the triple Higgs coupling [Abdughani, Wang, Wu, Yang, Zhao, arXiv: 2005.11086]





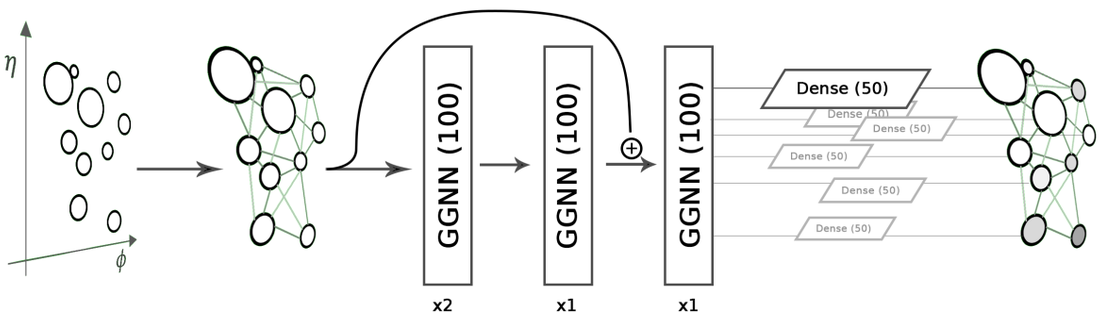
Pileup Mitigation



- Pileup: additional proton-proton interactions in the colliding beams
- One approach for pileup mitigation: classify each particle into "hard-scattering" vs "pileup" (or, compute a pileup weight for each particle)

Gated graph neural network

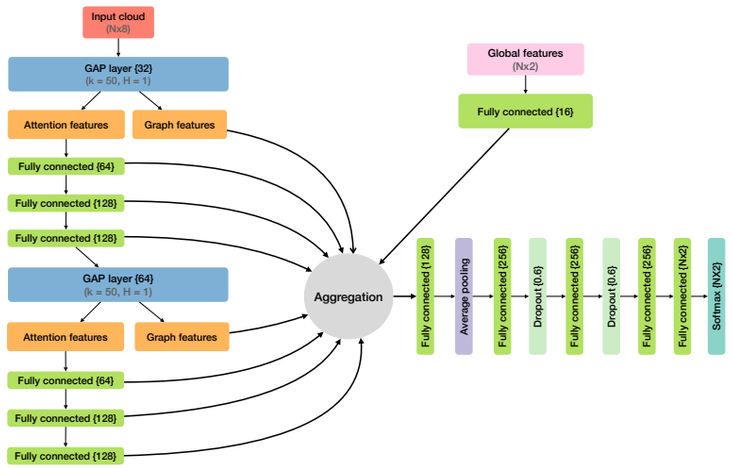
(radius graph, $\Delta R = 0.3$)



Martinez, Cerri, Pierini, Spiropulu, Vlimant, arXiv:1810.07988

ABCNet (Graph attention network)

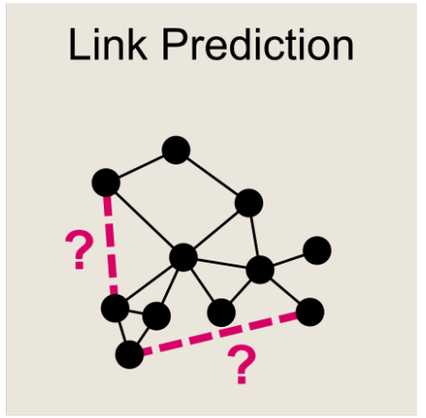
(kNN graph, $k = 50$)

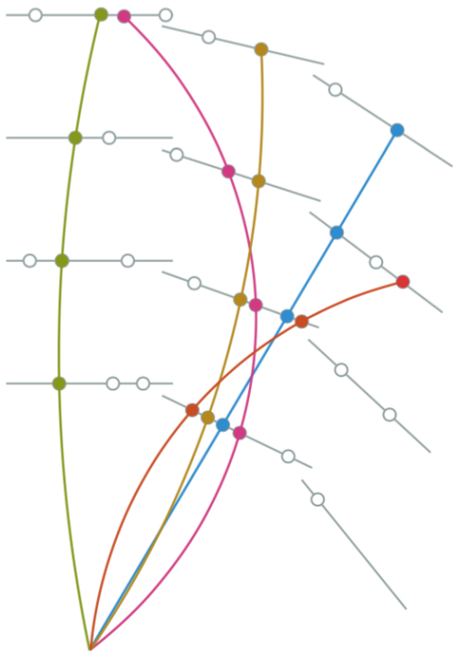


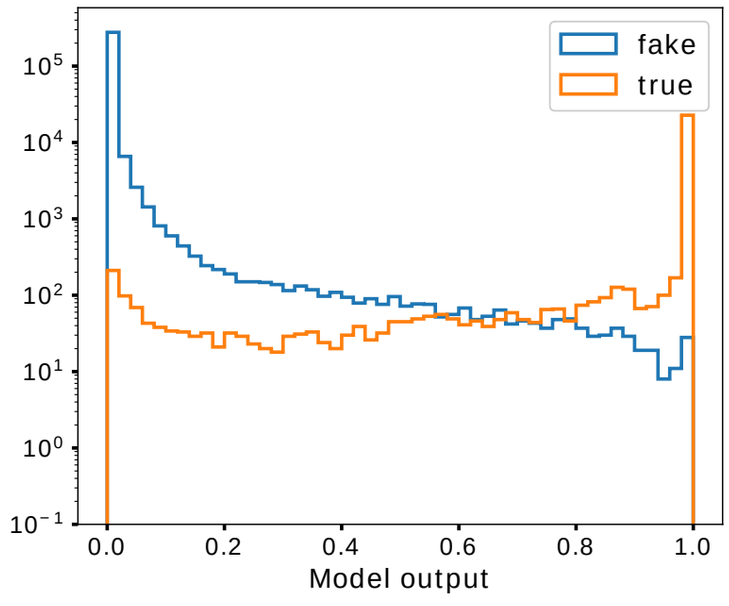
Mikuni, Canelli, arXiv: 2001.05311

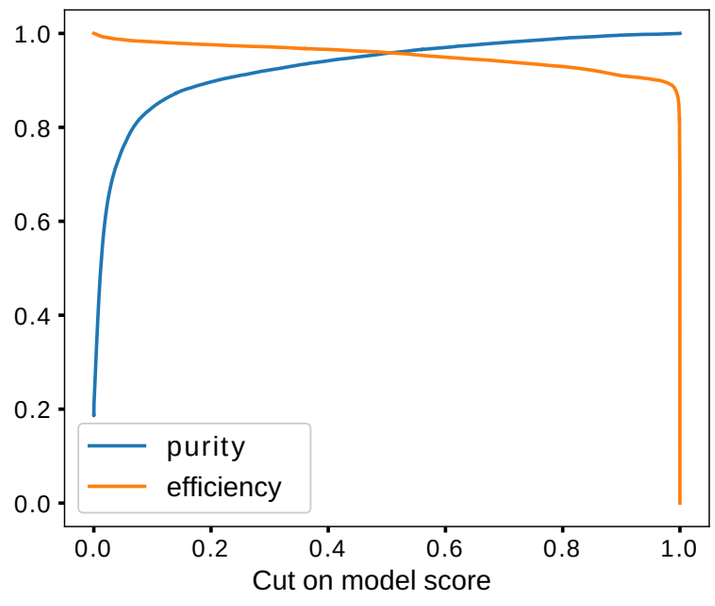
CHARGED PARTICLE TRACKING

- Tracking: "connecting the dots"
  - finding what set of isolated measurement (hits) belong to the same particle
- Formulated as edge classification w/ GNN
  - each node is a hit 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



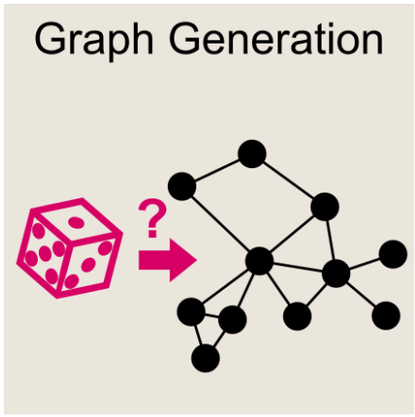




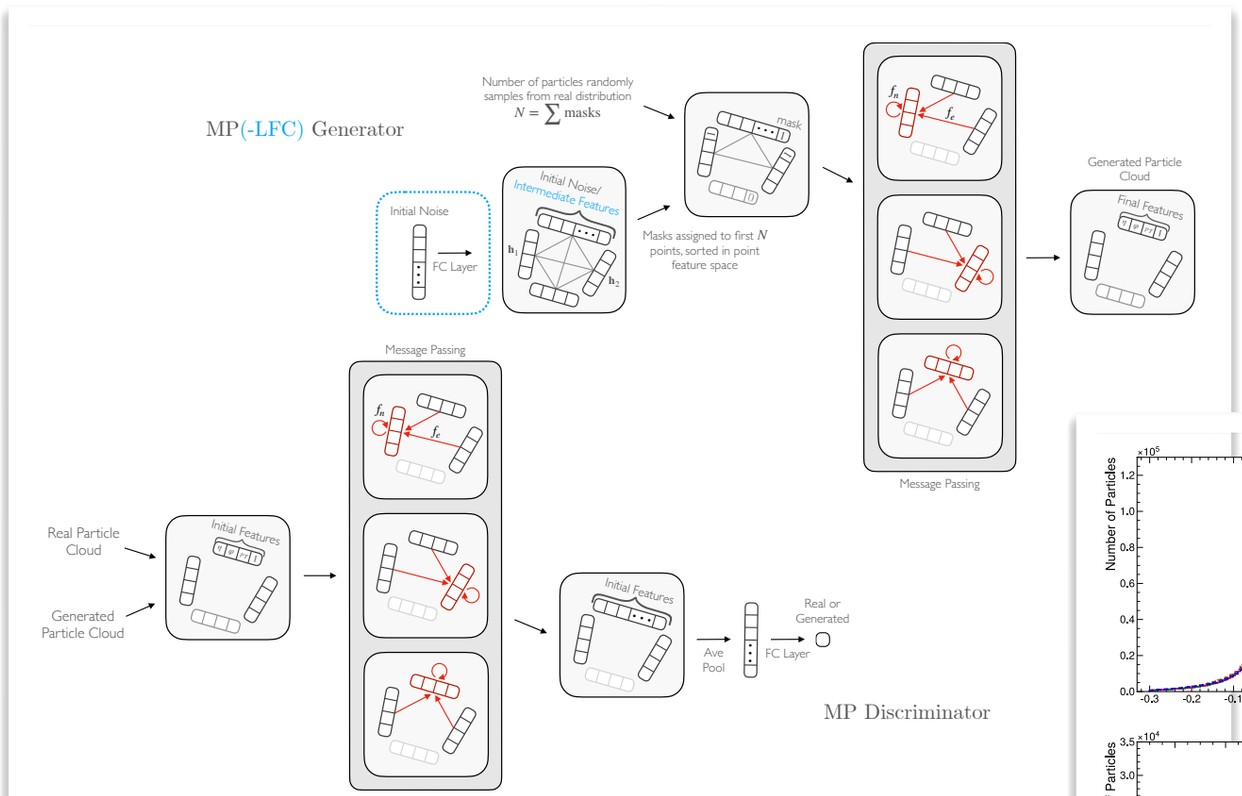


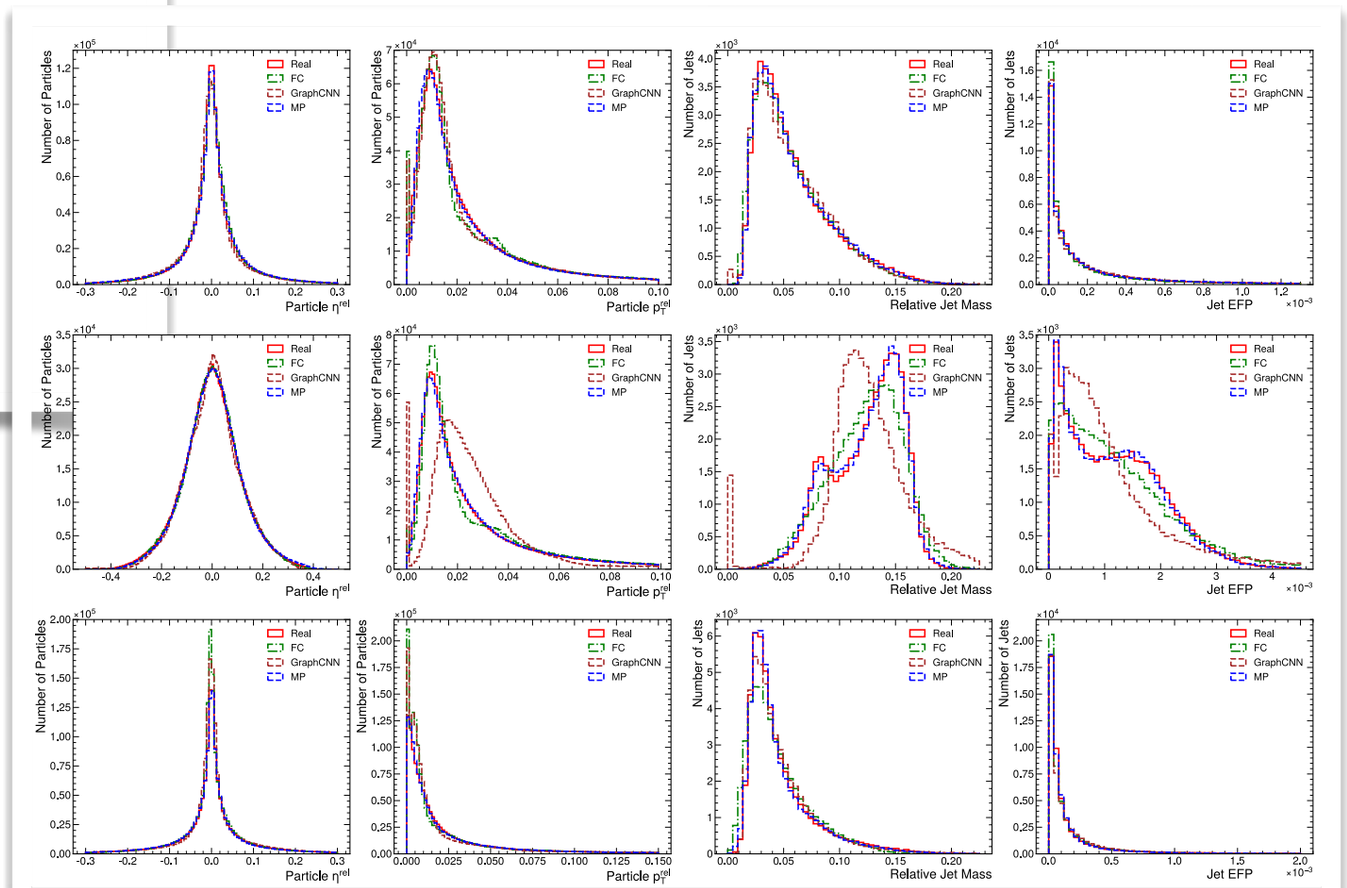
Farrell *et. al.*, arXiv:1810.06111
Ju *et. al.*, arXiv:2003.11603

Particle Cloud Generation



- Message Passing Generative Adversarial Network (MPGAN)





Kansal, Duarte, Su, Orzari, Tomei, Pierini, Touranakou, Vlimant, Gunopulos, arXiv:2106.11535

QUESTIONS?

PREPARATION

- Set up Weaver:
  - git clone https://github.com/hqucms/weaver.git
  - follow the instructions to set up the environment
    - https://github.com/hqucms/weaver#set-up-your-environment

- Download the top tagging reference dataset:
  - https://zenodo.org/record/2603256