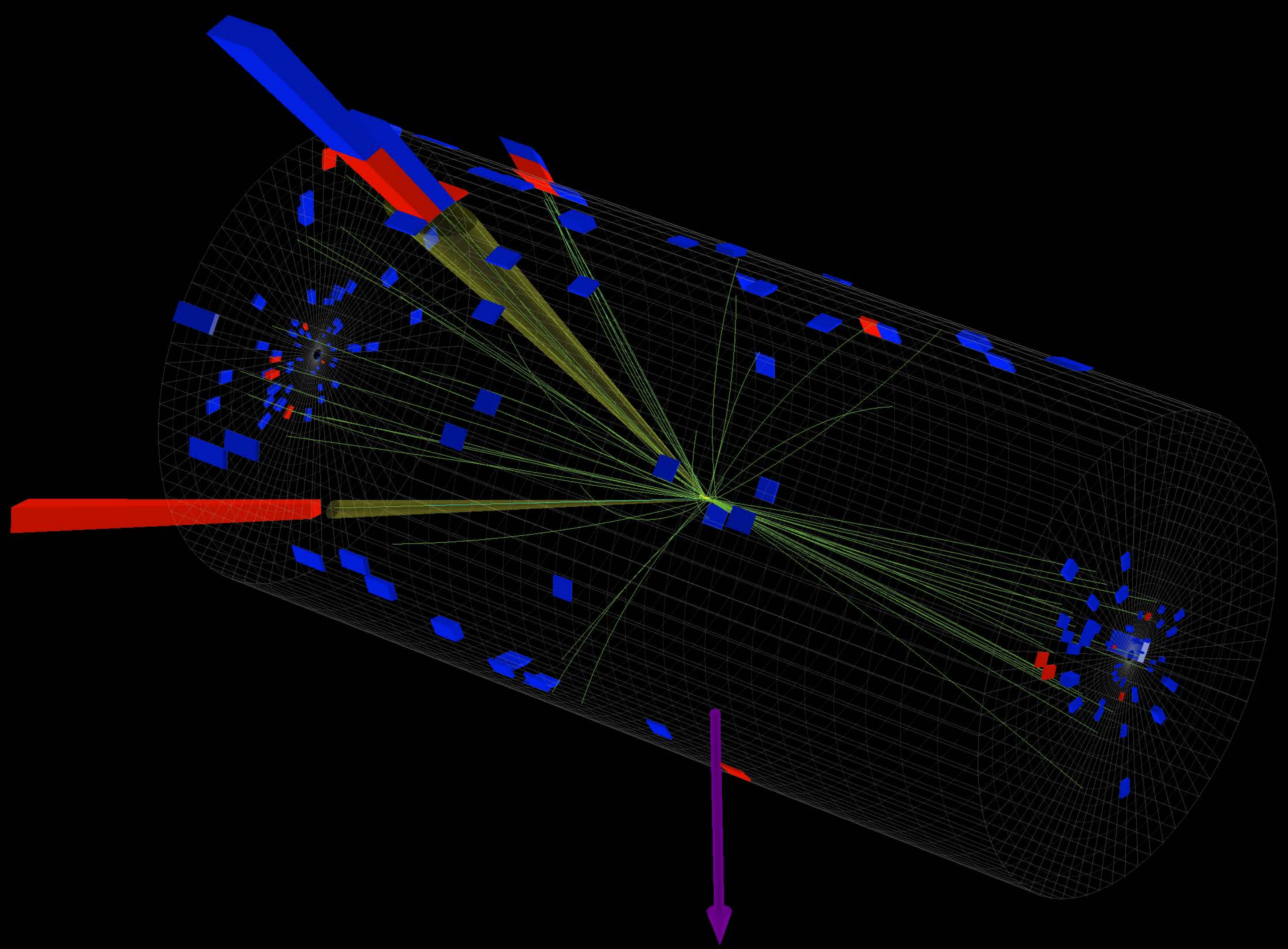
Graph Neural Networks for (Experimental) Particle Physics

Huilin Qu



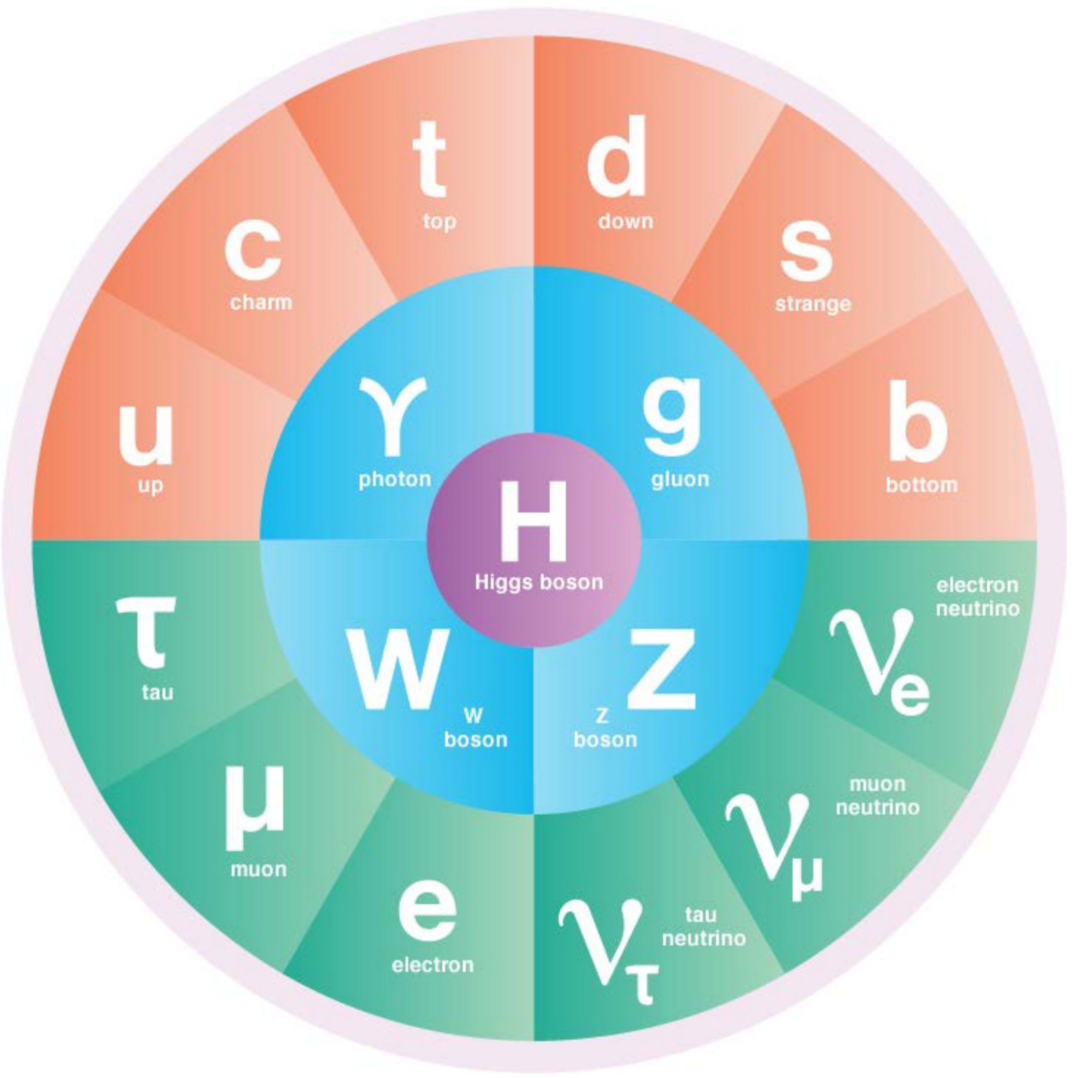
Machine Learning in Particle Theory

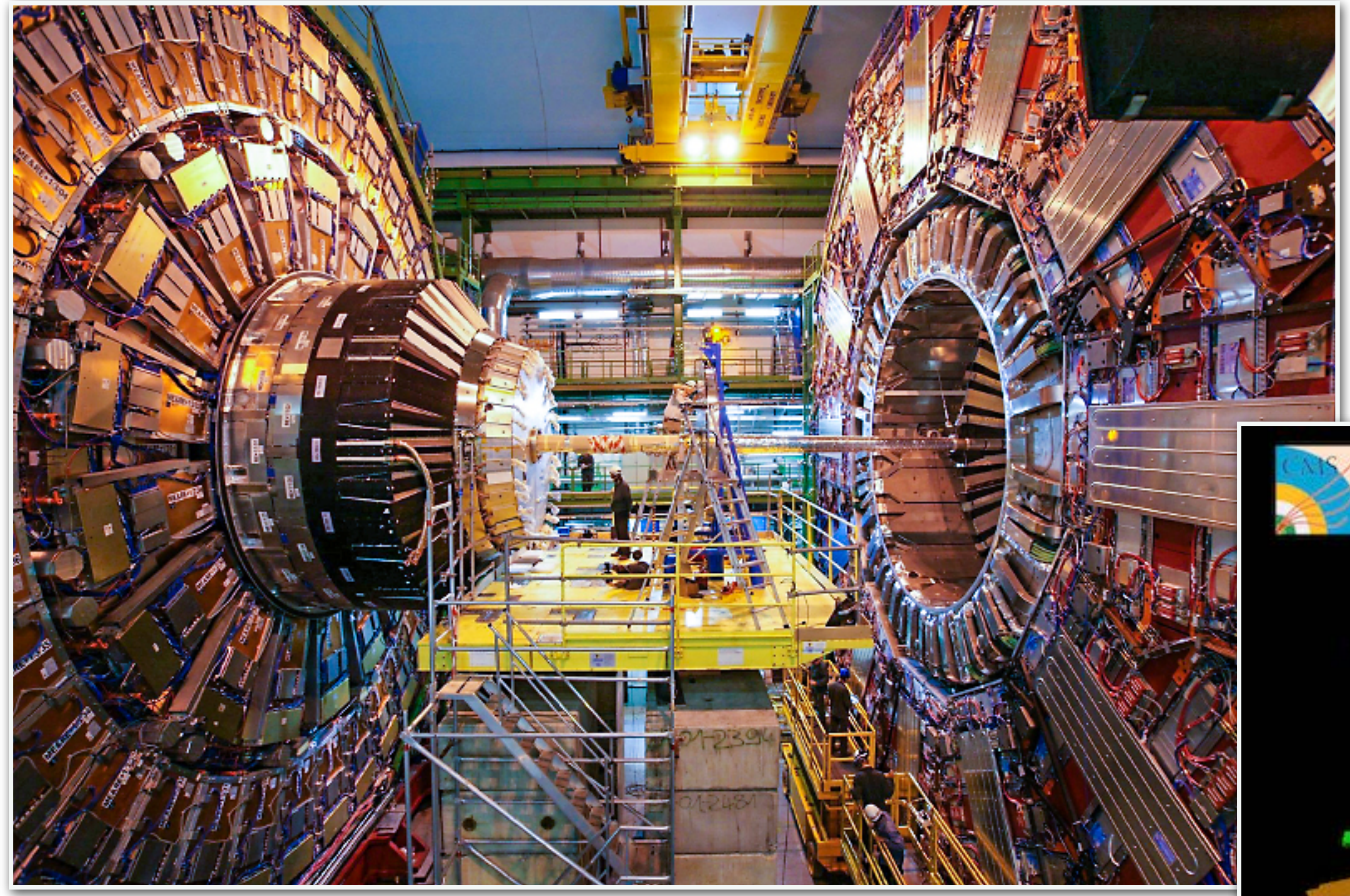
MITP Summer School 2023

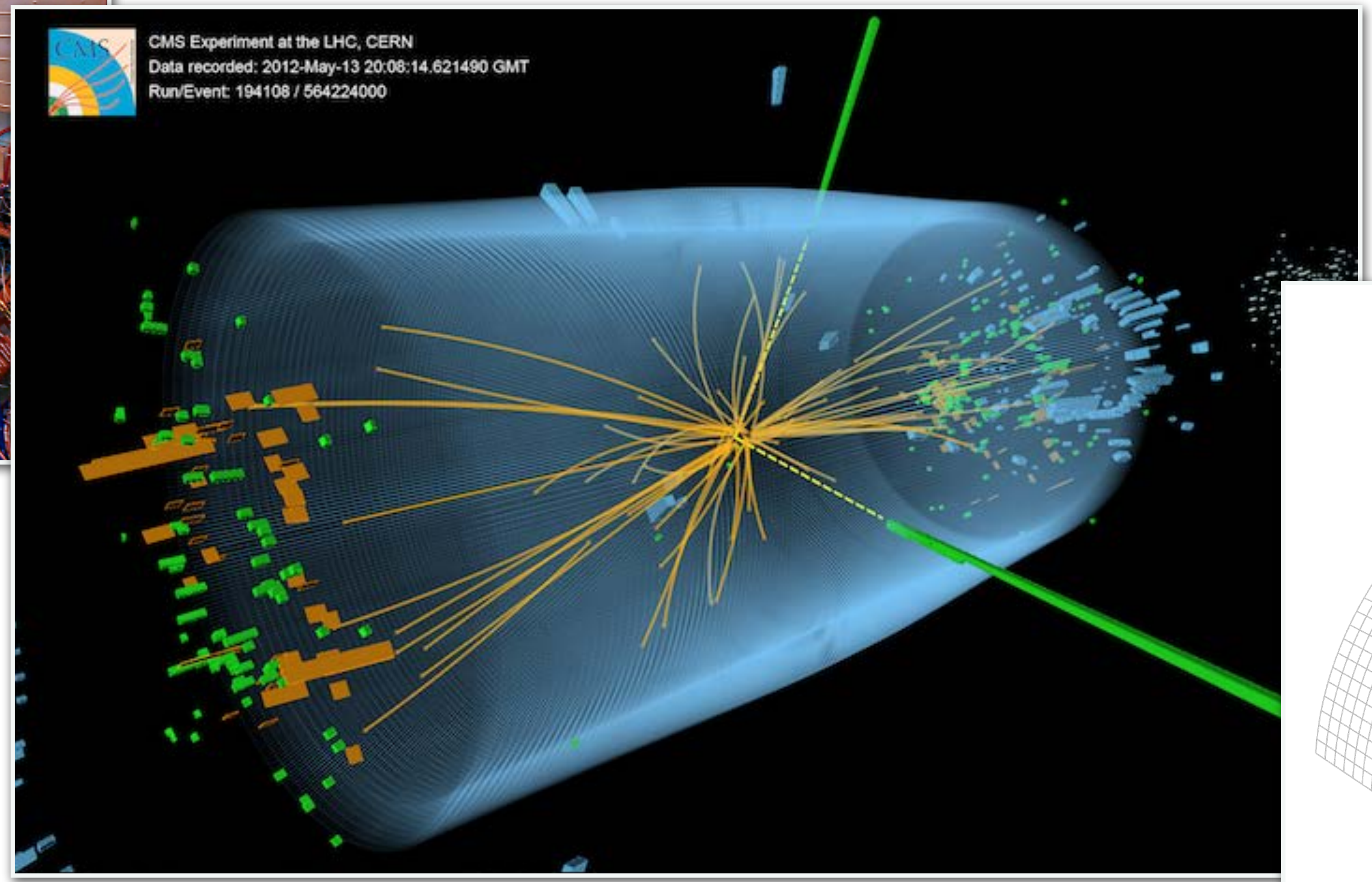
05.07.2023

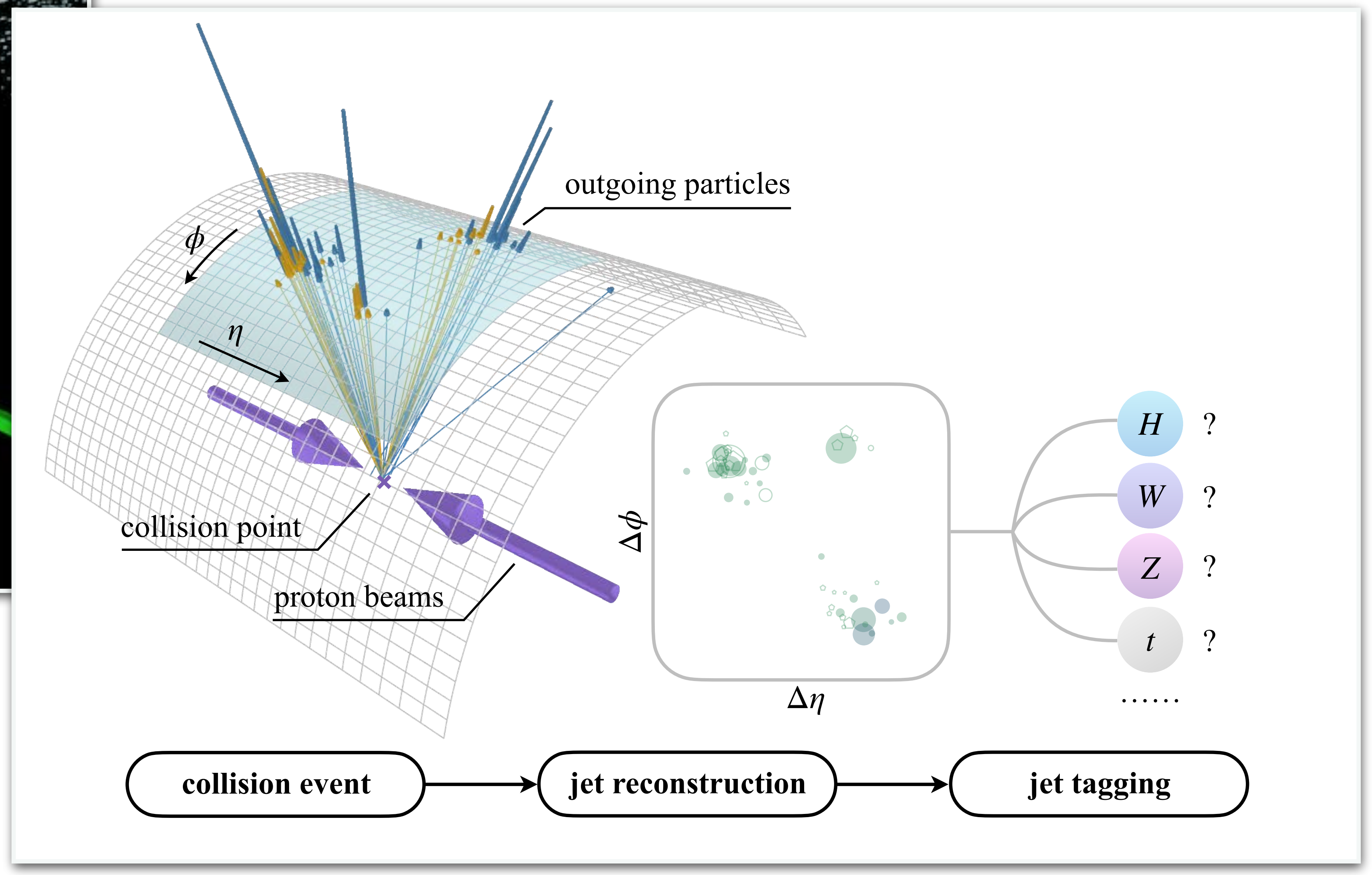
About Me

- Experimental particle physicist at CERN
  - interests: Higgs and New Physics @ LHC, jet physics, machine learning, ...





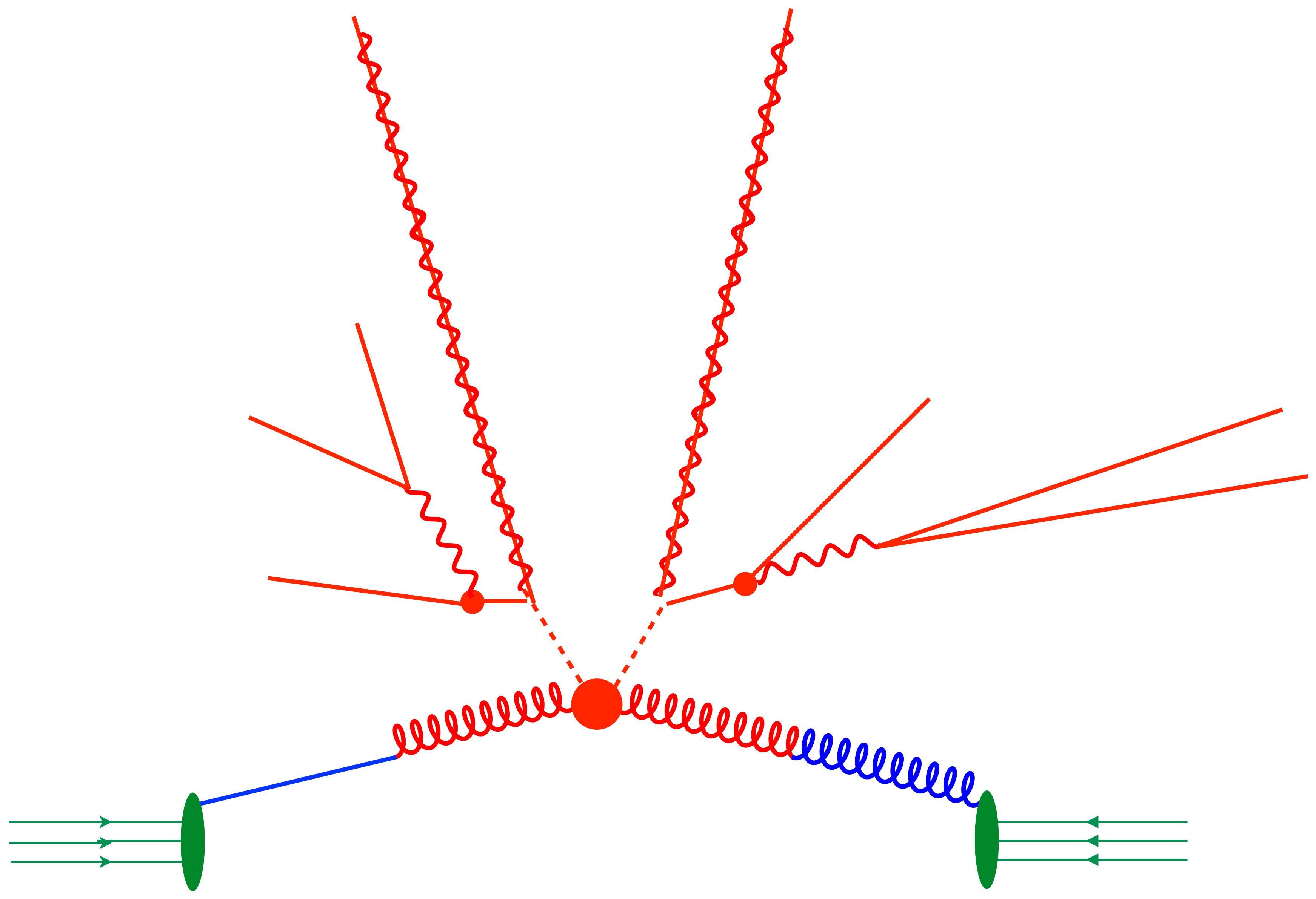




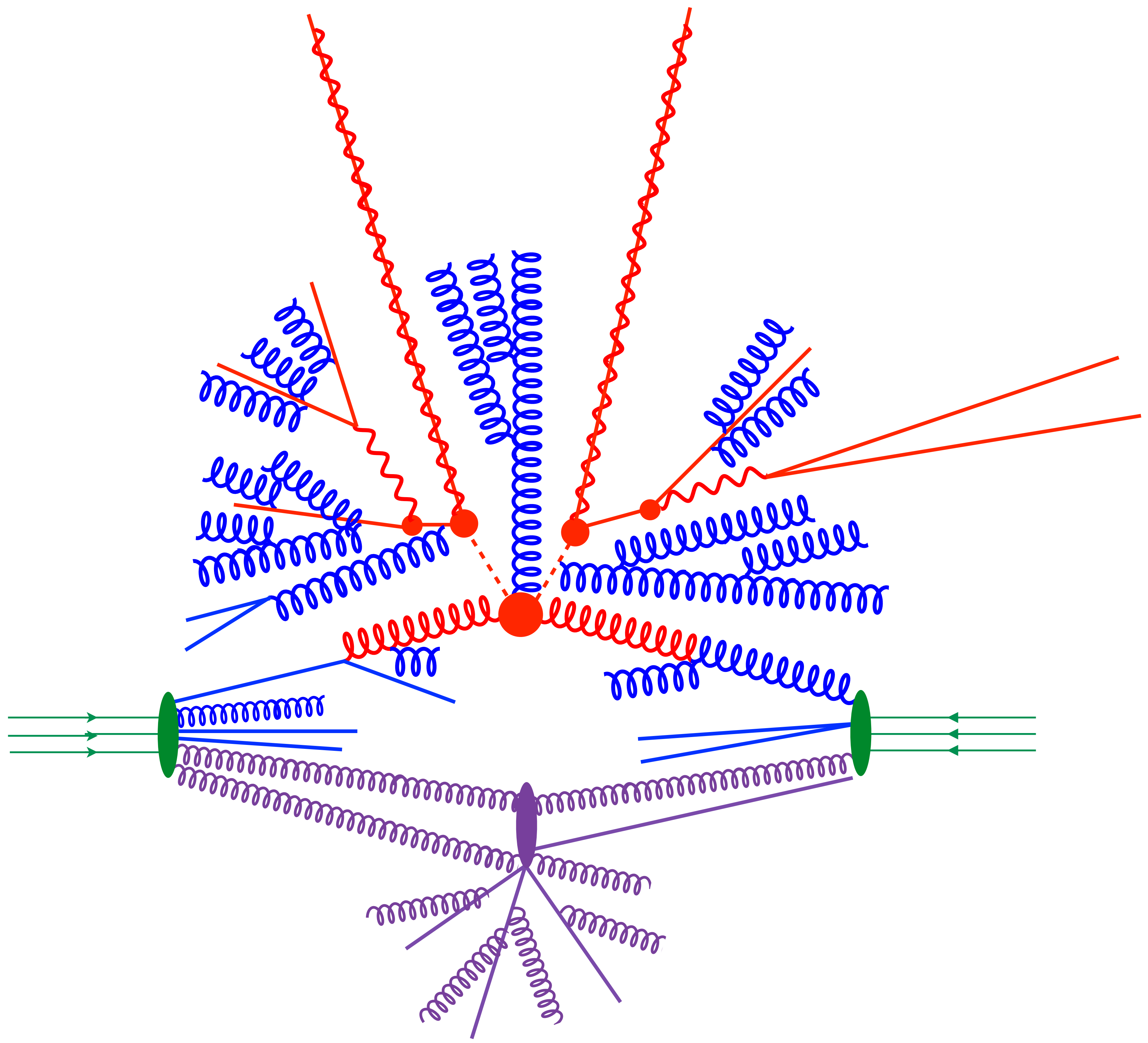
Large Hadron Collider

Particle Collision

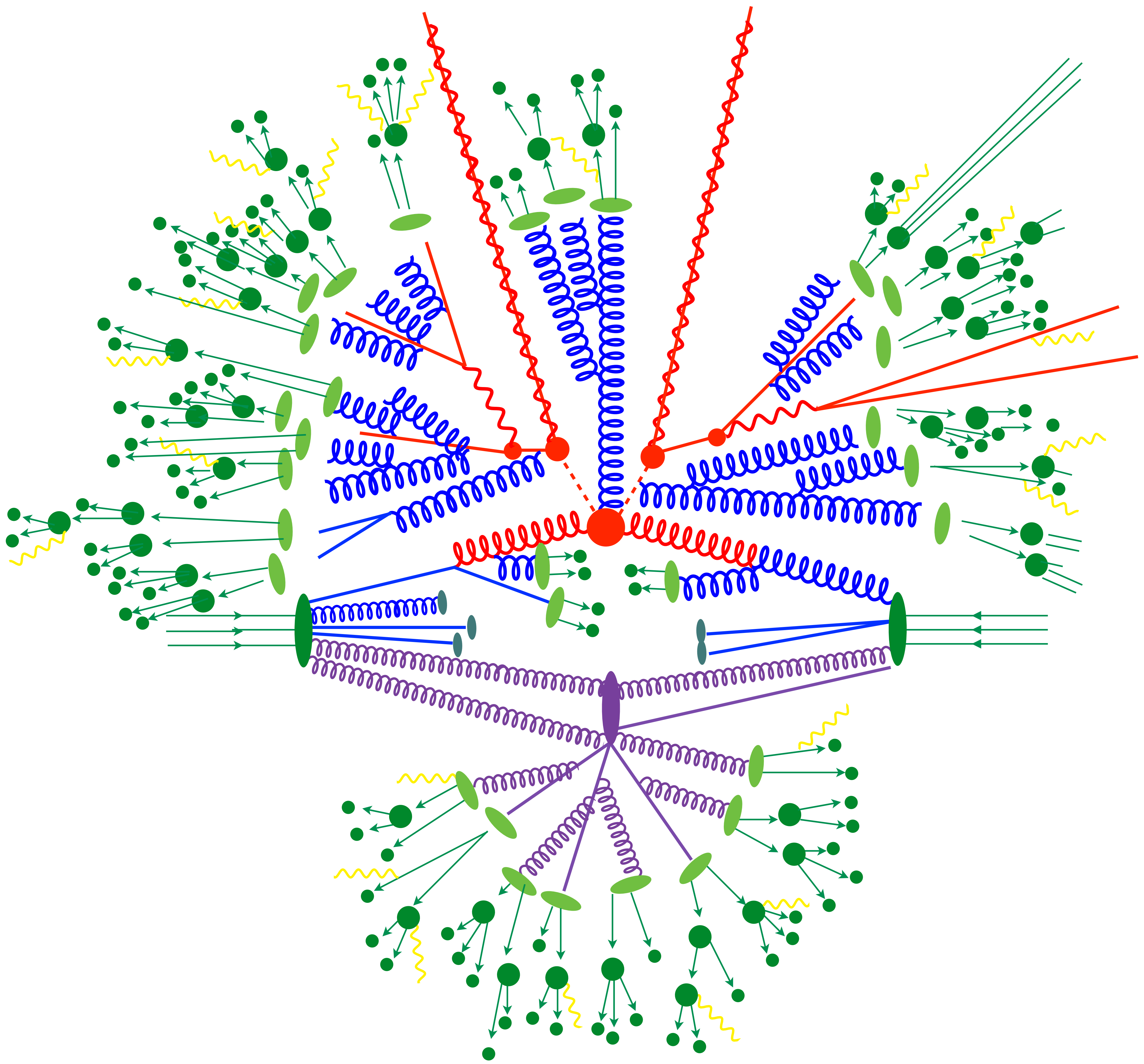
Particle Collision



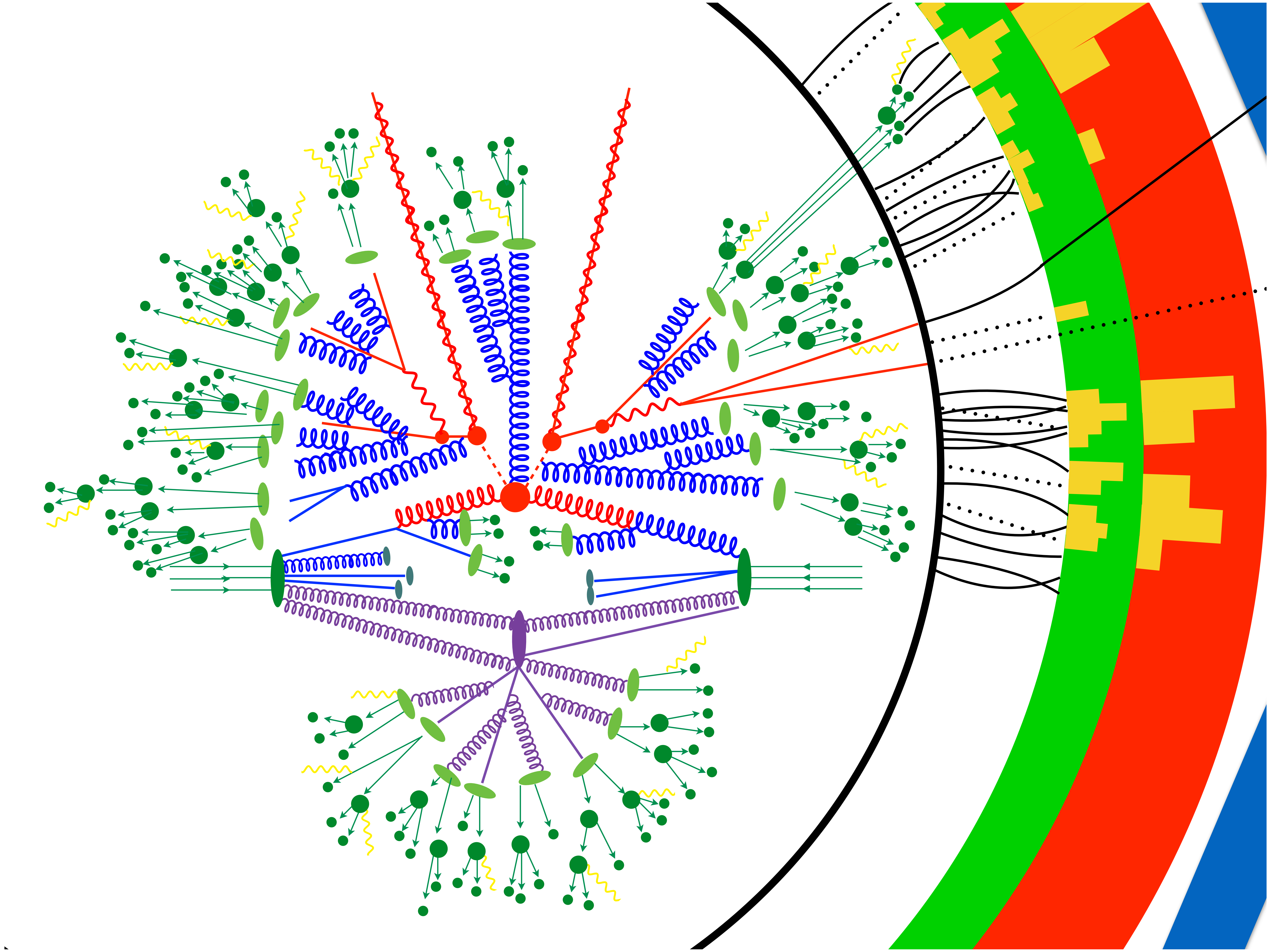
Particle Collision



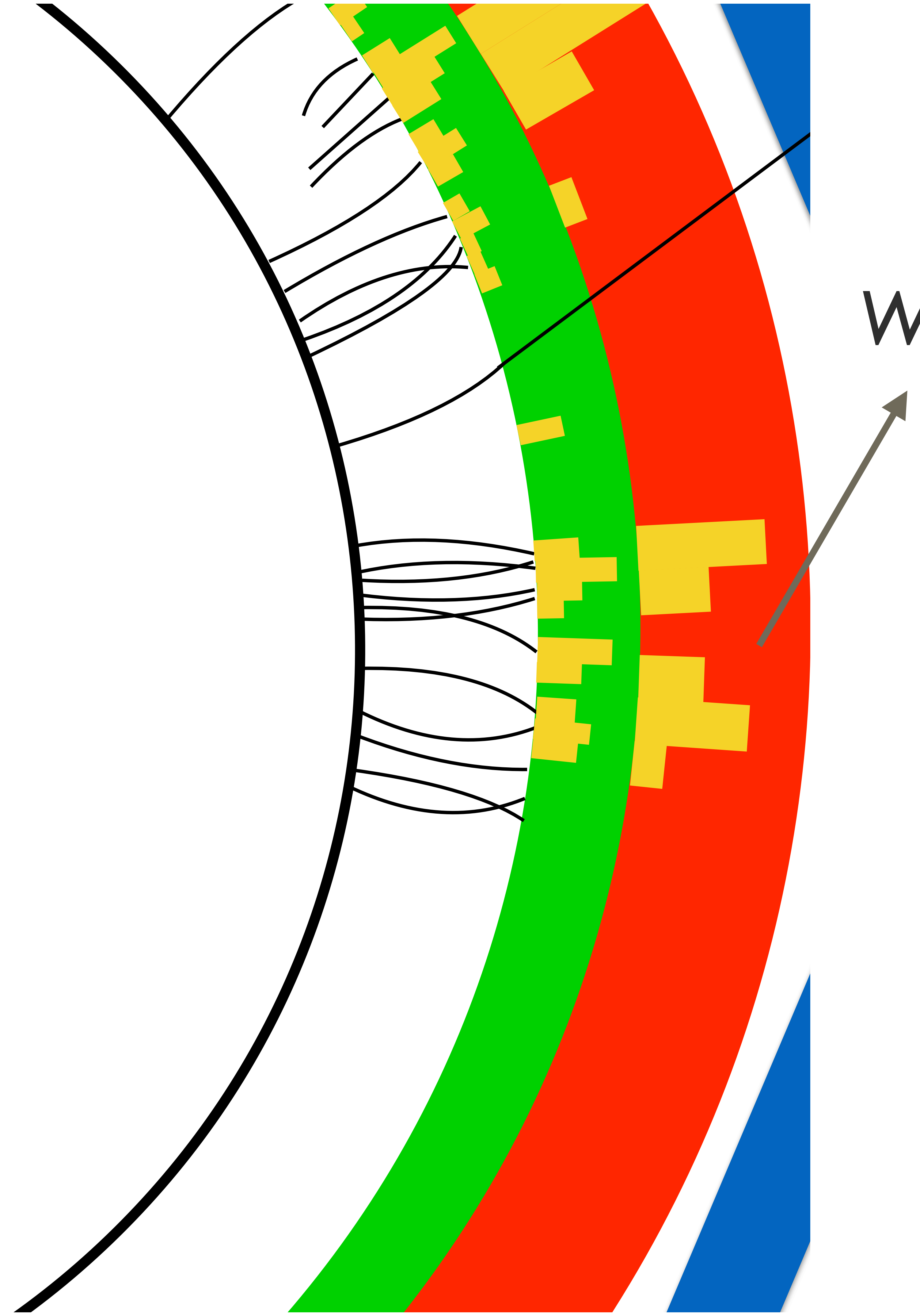
Particle Collision



Particle Collision



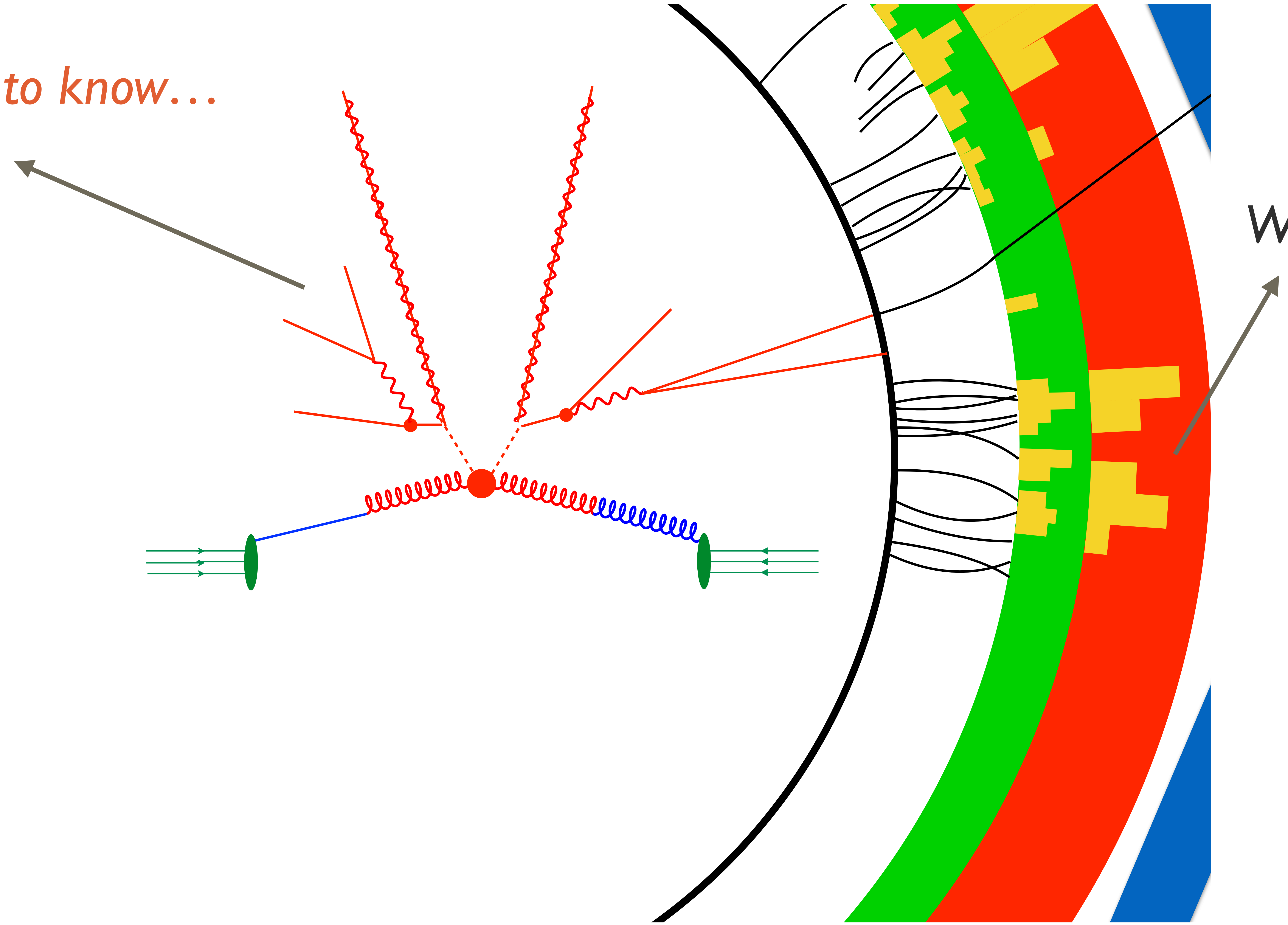
PARTICLE COLLISION



What we observe…

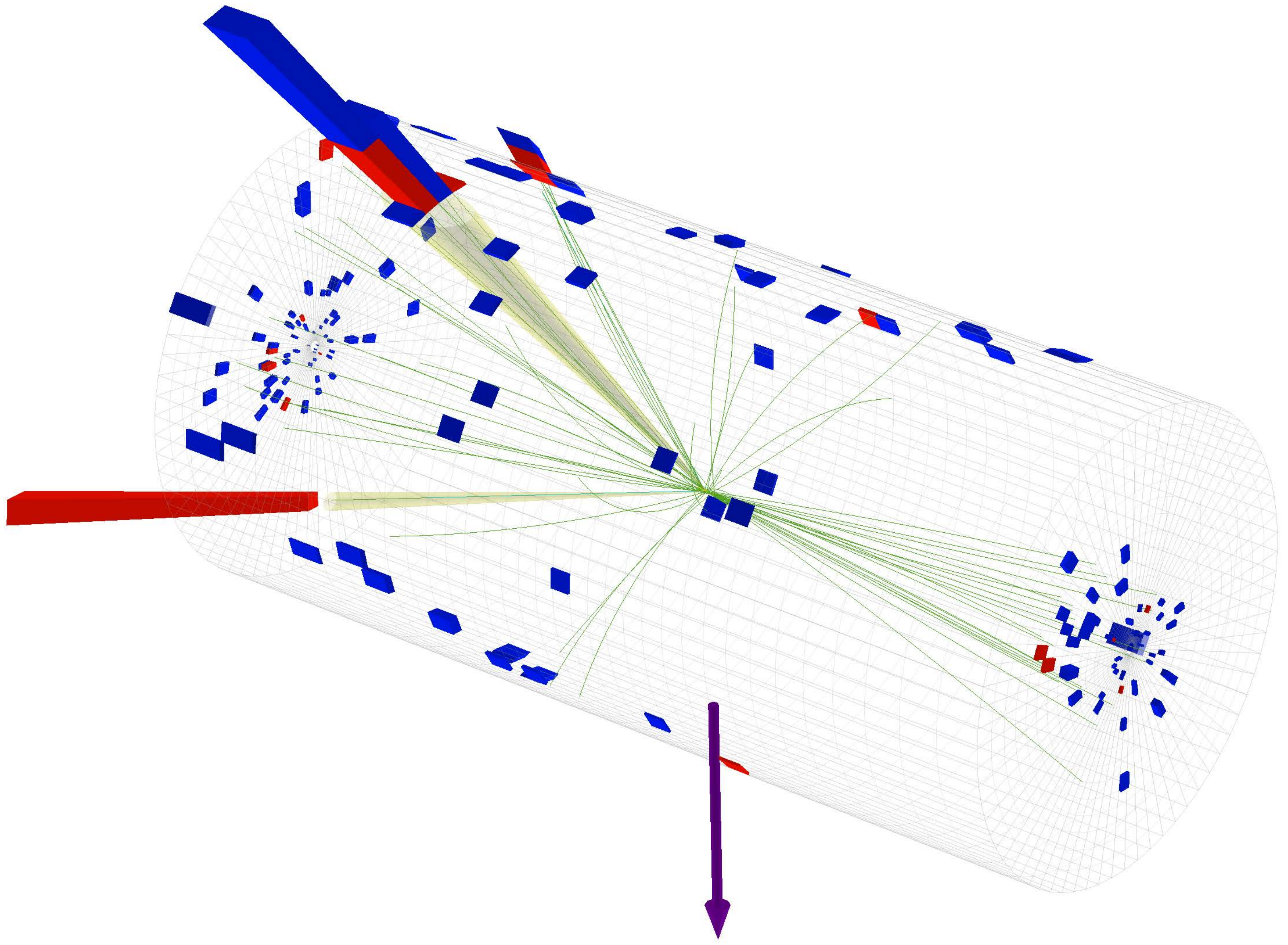
Particle Collision

What we want to know…

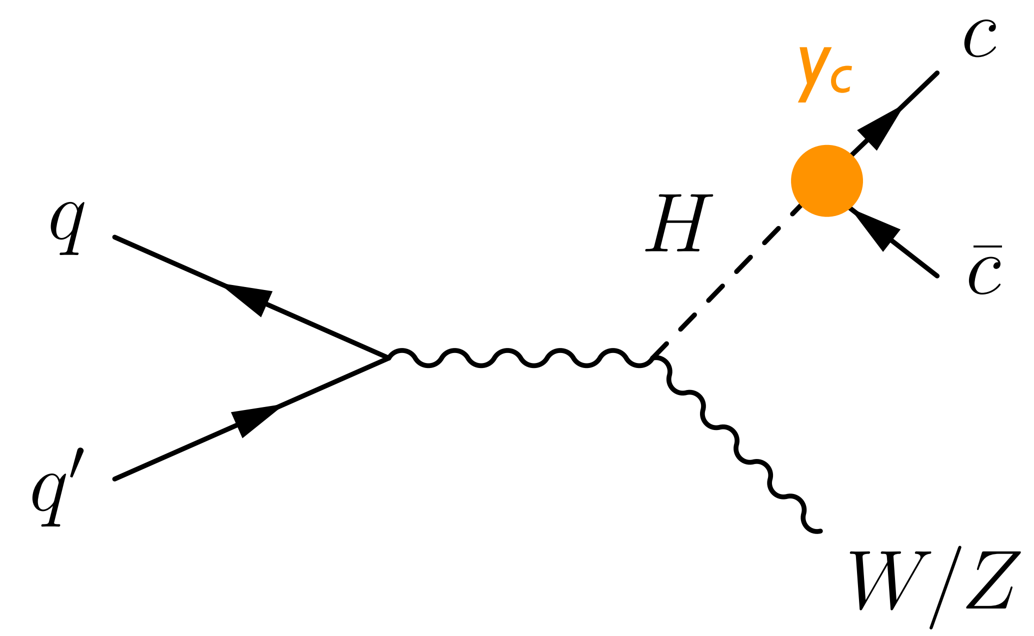


What we observe…

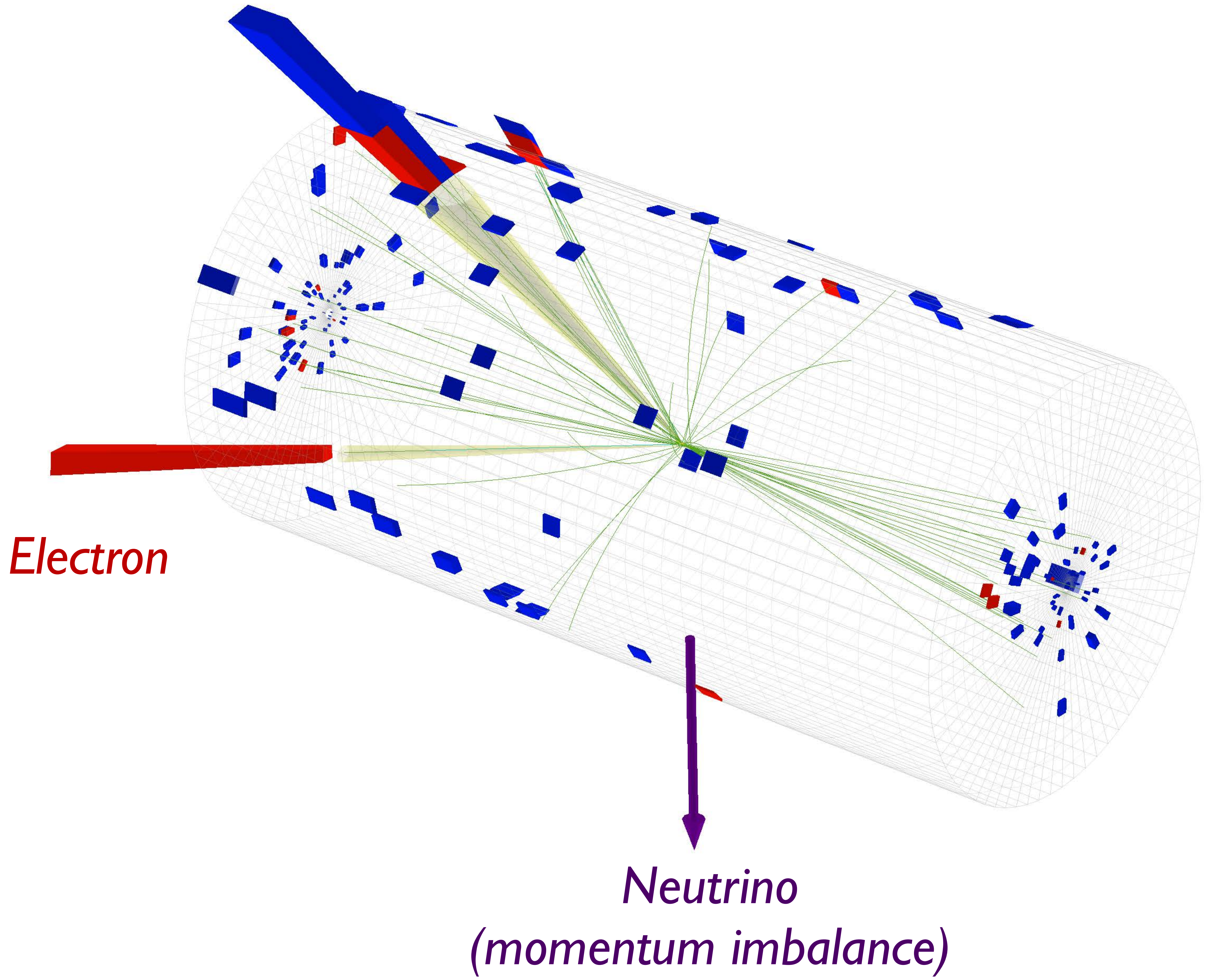
Example: What Is the Collision Event?

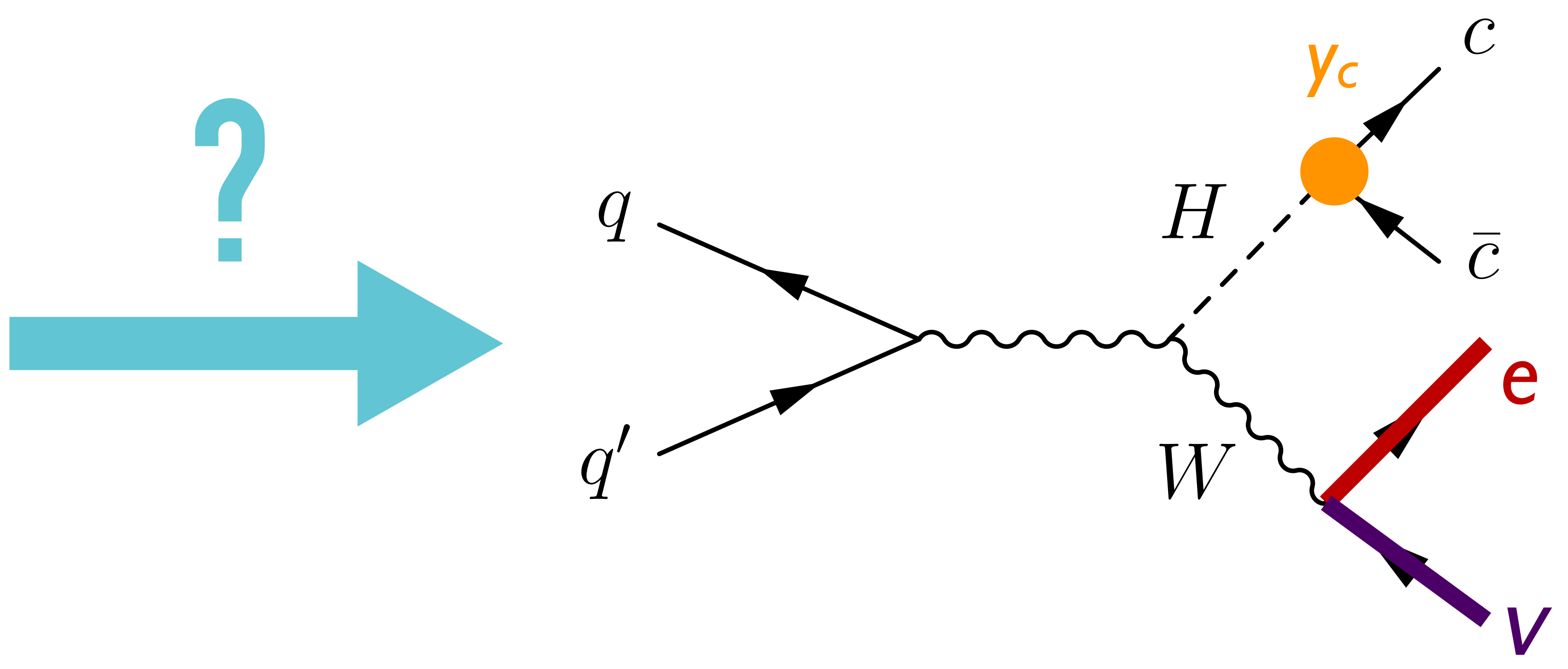


?

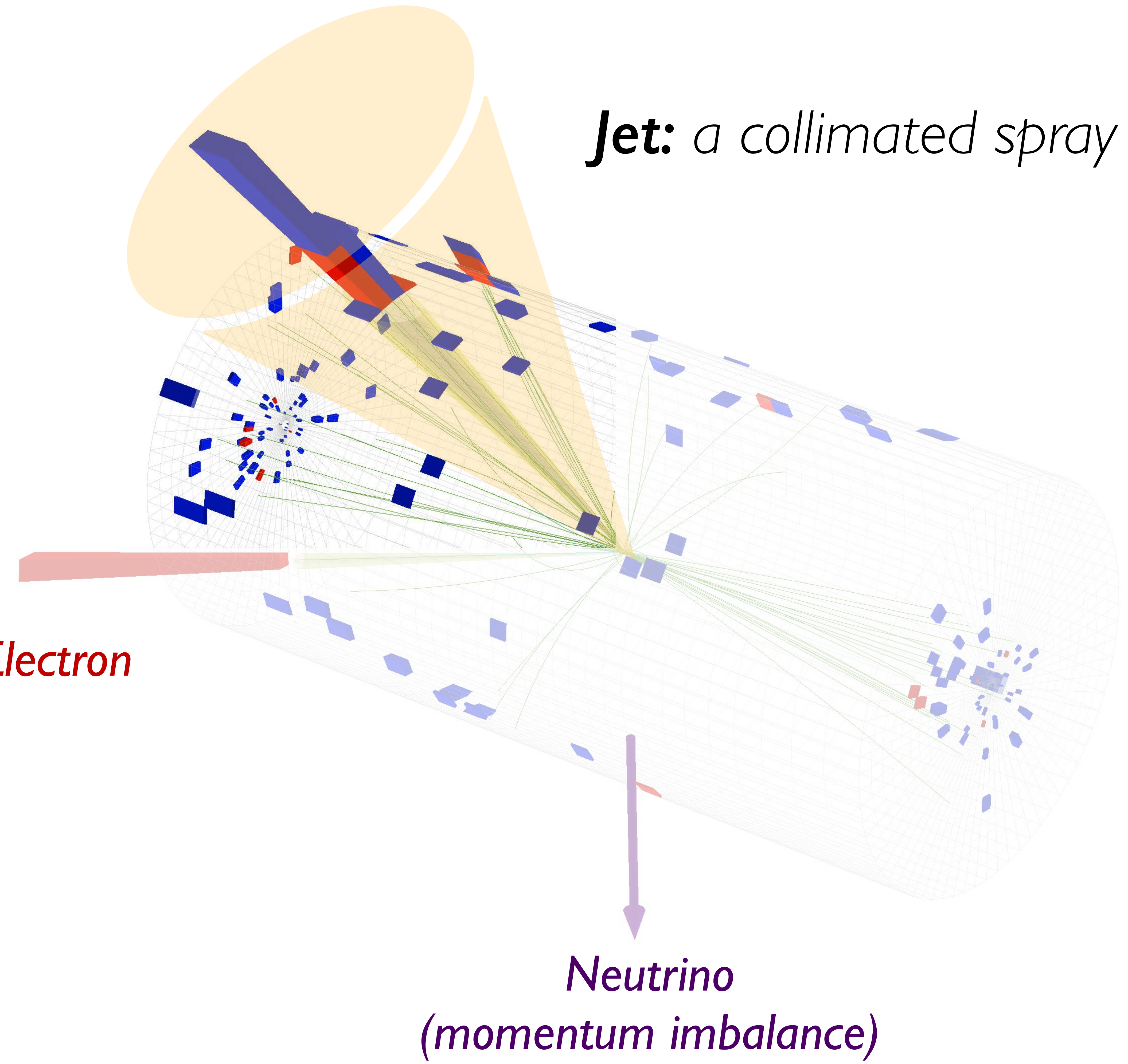


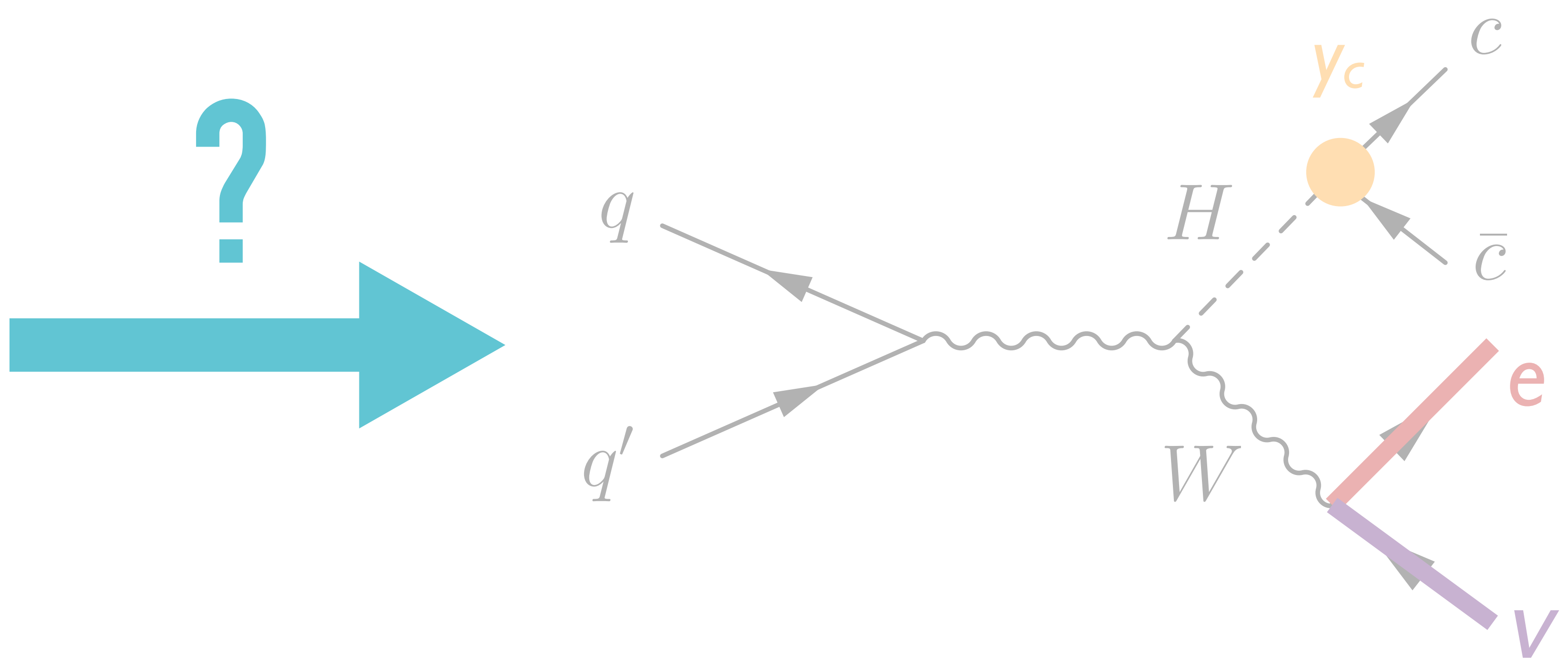
Example: What Is the Collision Event?





Example: What Is the Collision Event?

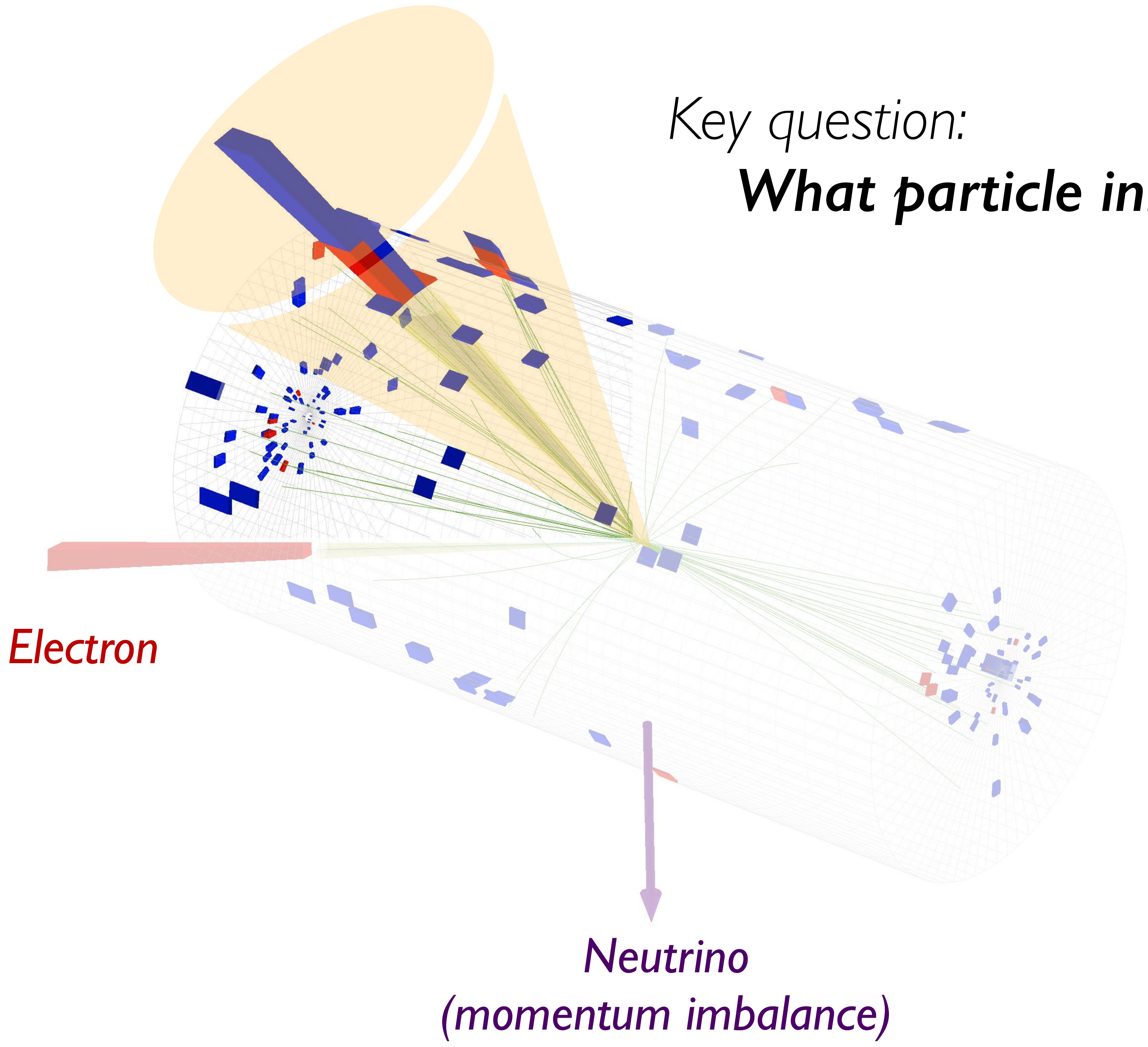


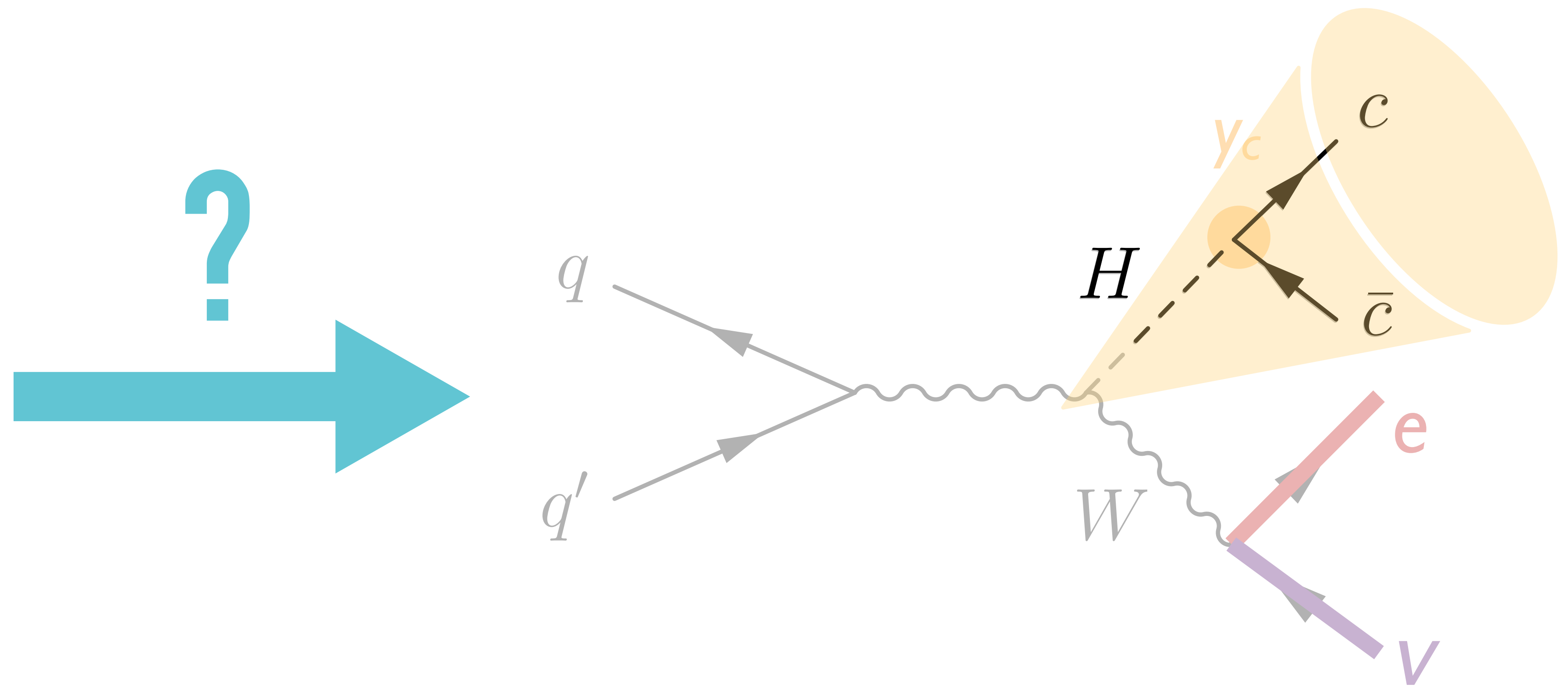


Example: What Is the Collision Event?

Key question:

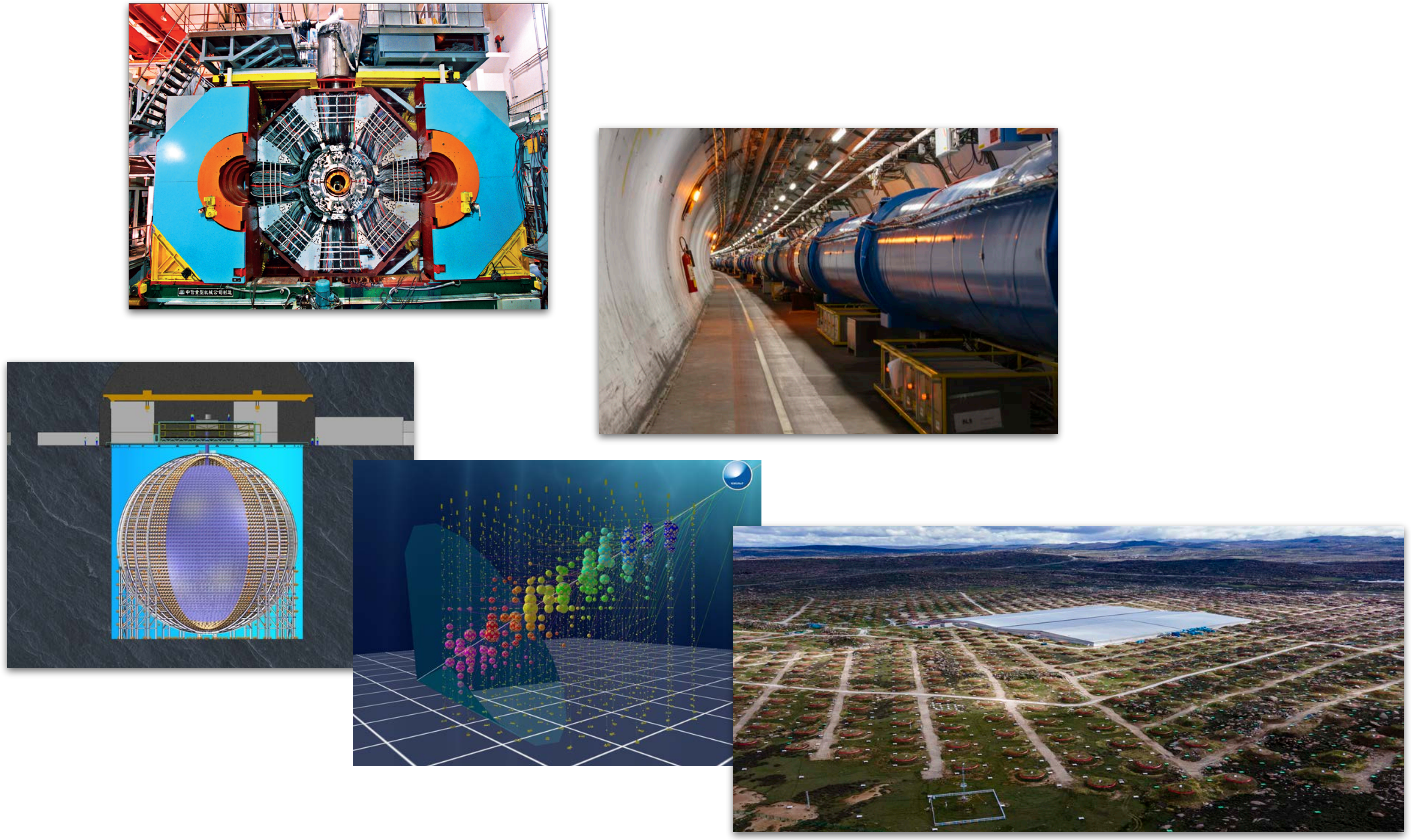
What particle initiates the jet?





The Data Challenge in High Energy Physics

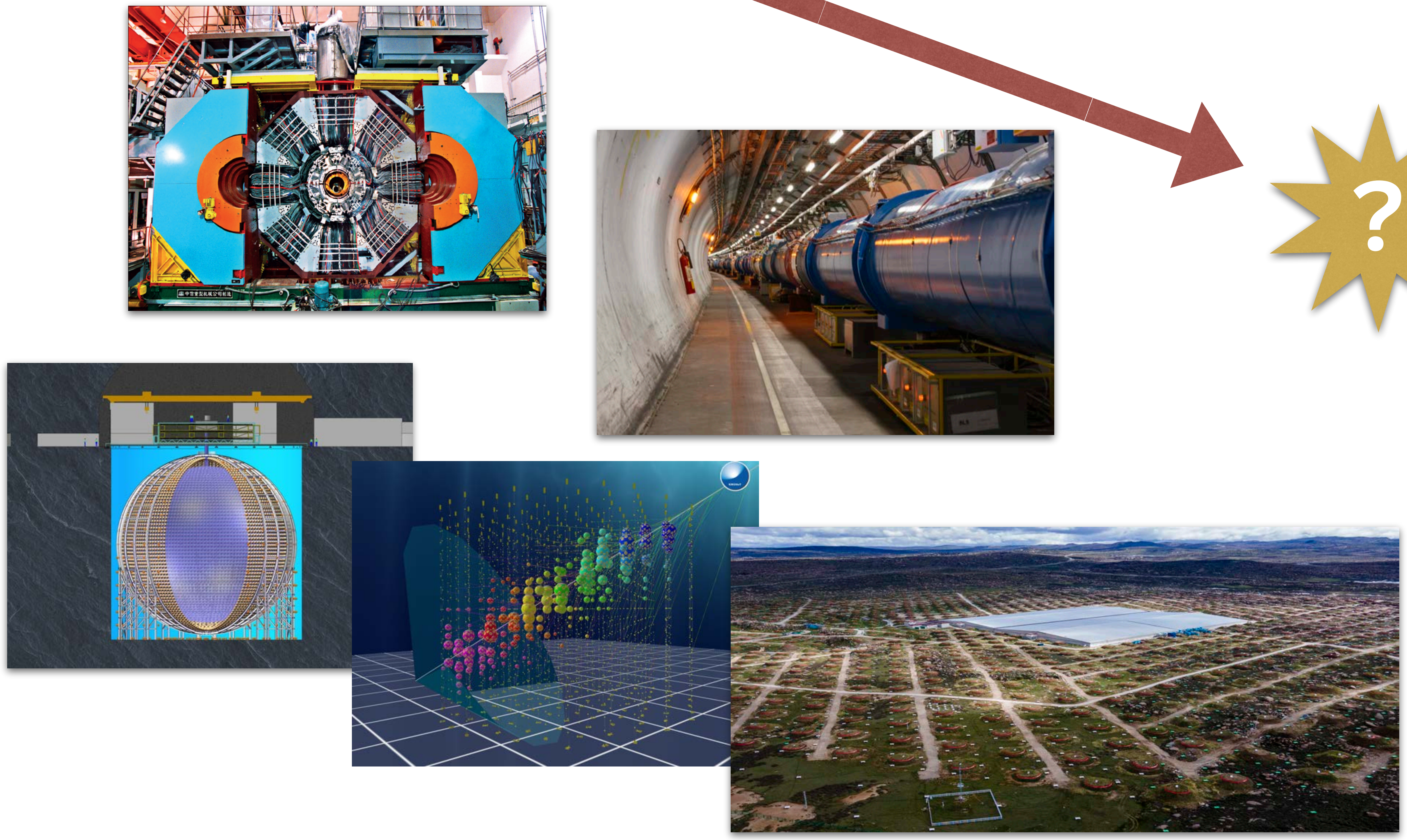
HEP



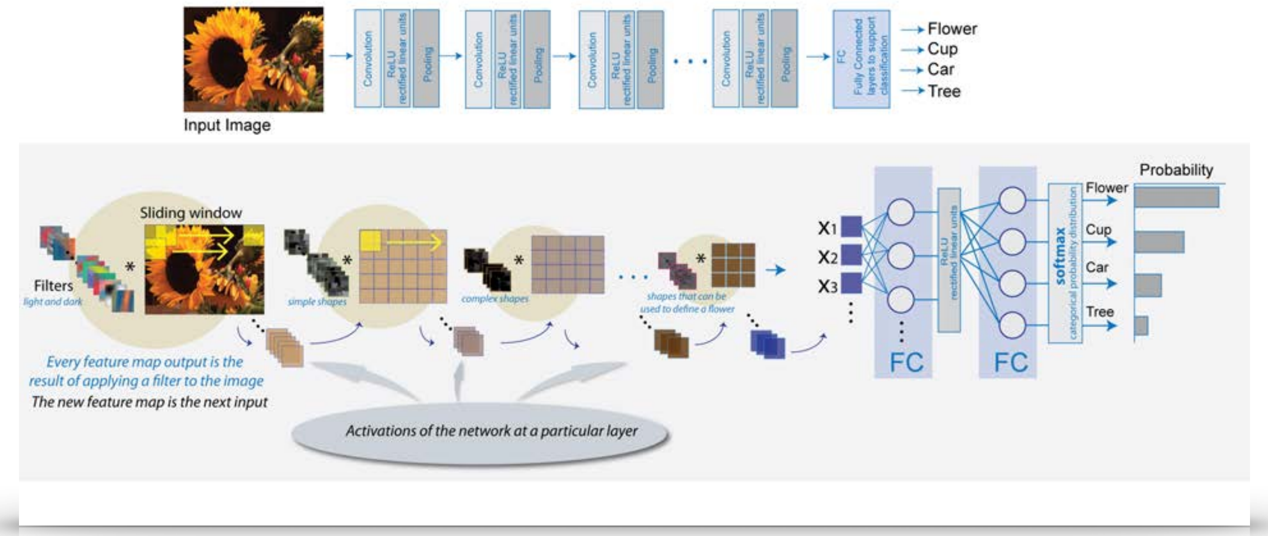
Large volume of data, complex topology, …

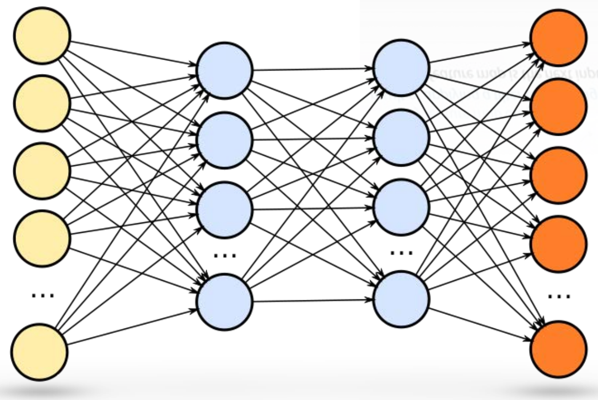
AI + HEP: At the Collision Point

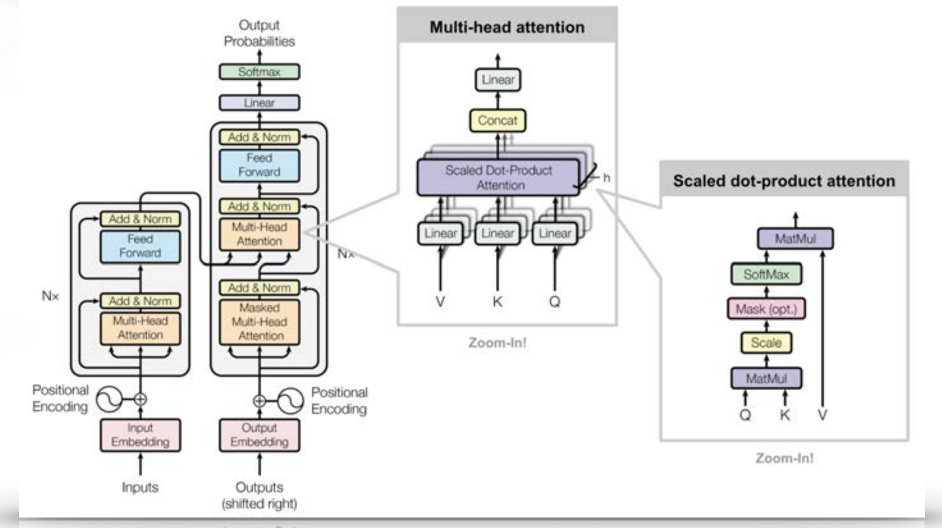
HEP



?







AI

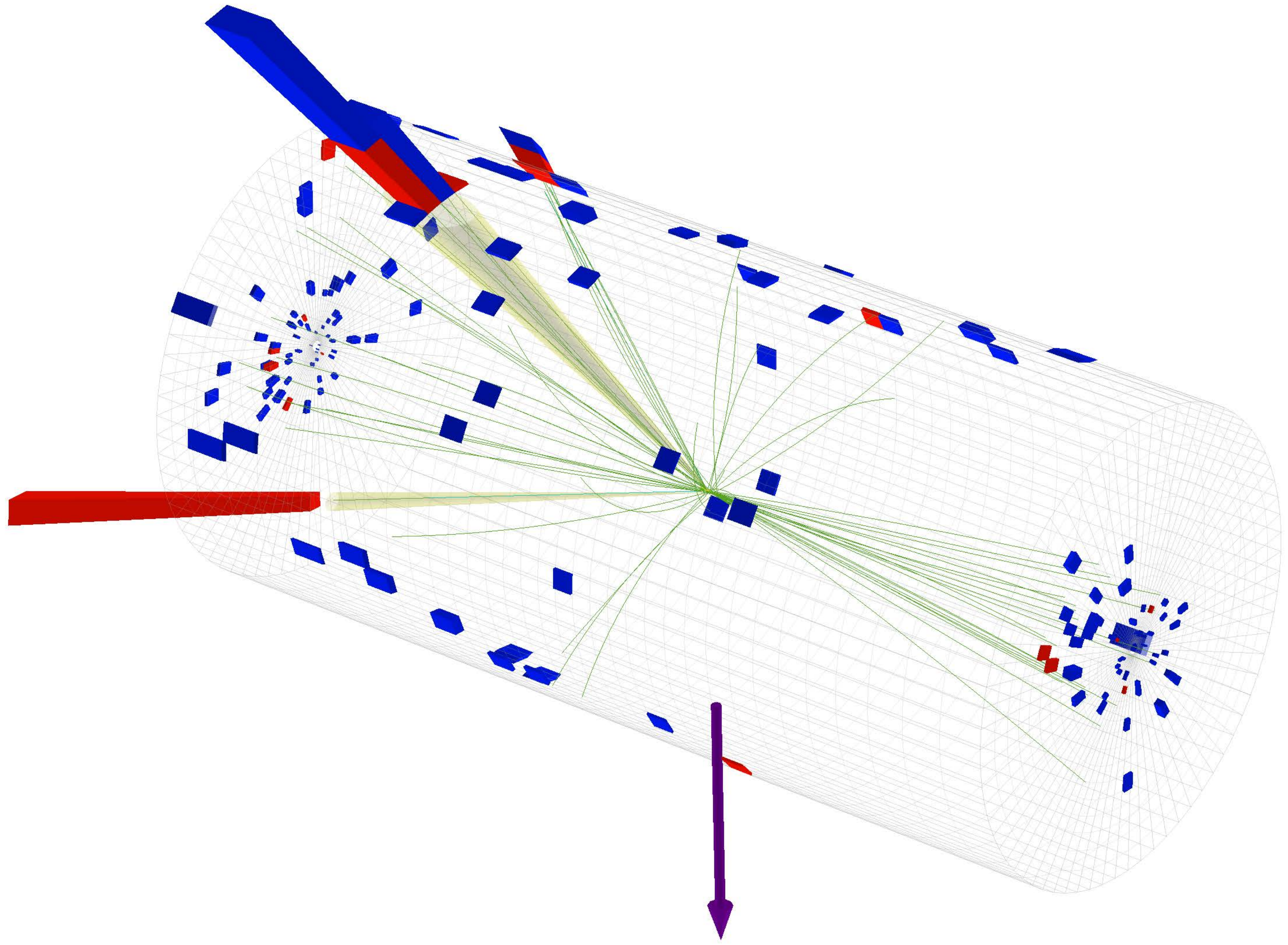
Collimate HEP and AI to make them collide!

Large volume of data, complex topology, …

Data Representations

DATA REPRESENTATION

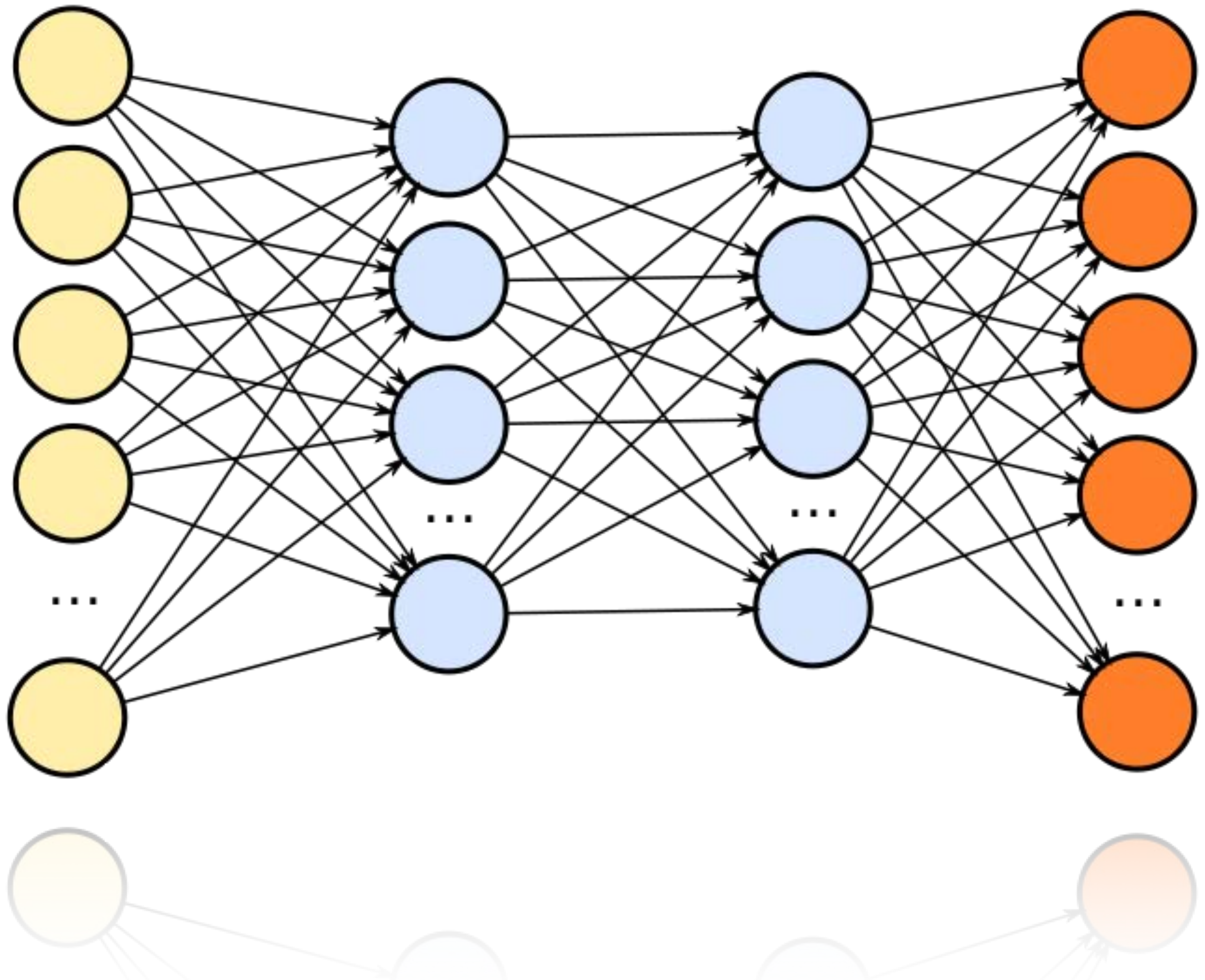
HEP



Collision events, detector hits, sensor arrays, …

×

ML

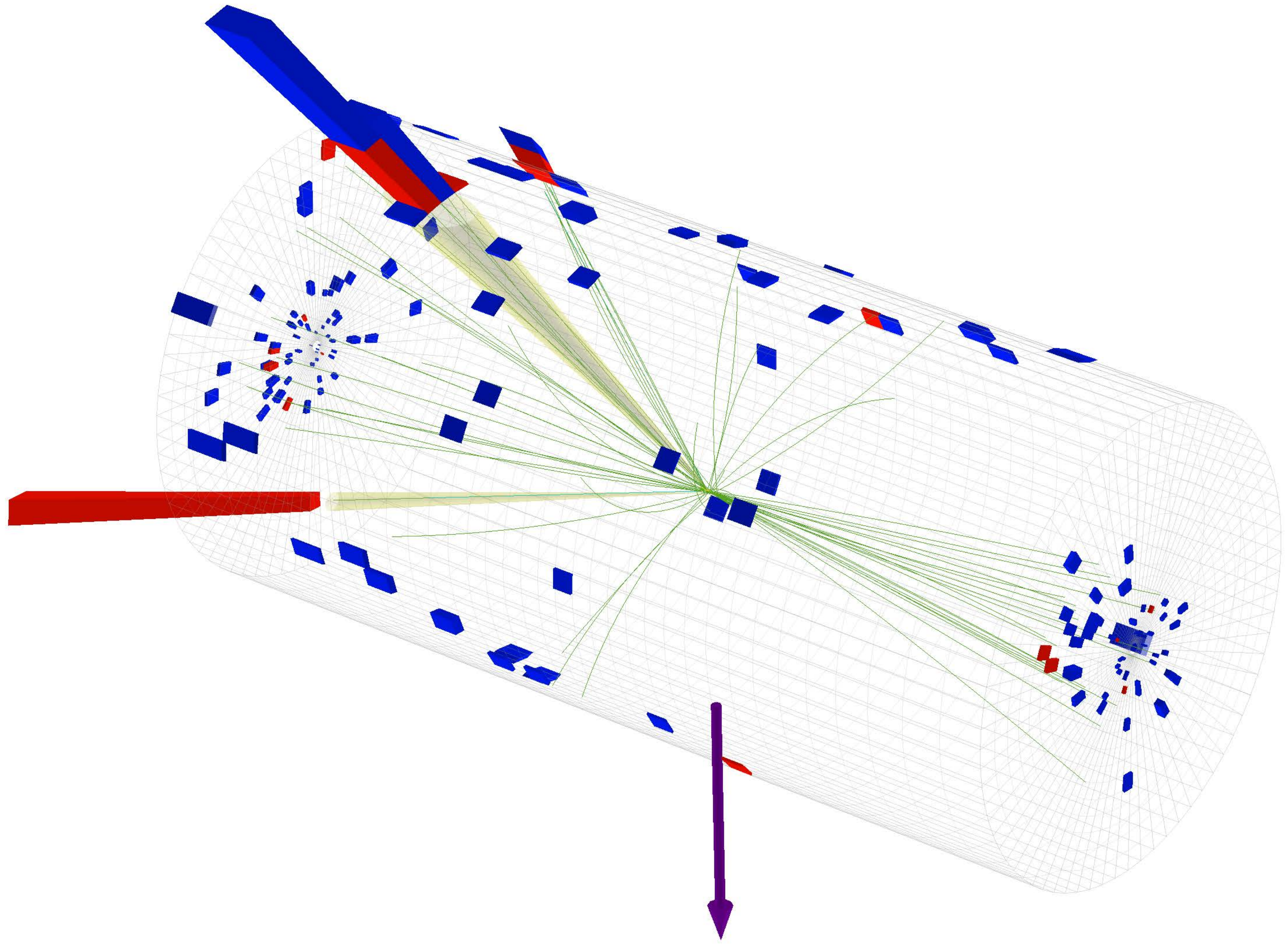


First and foremost:

How to represent the data?

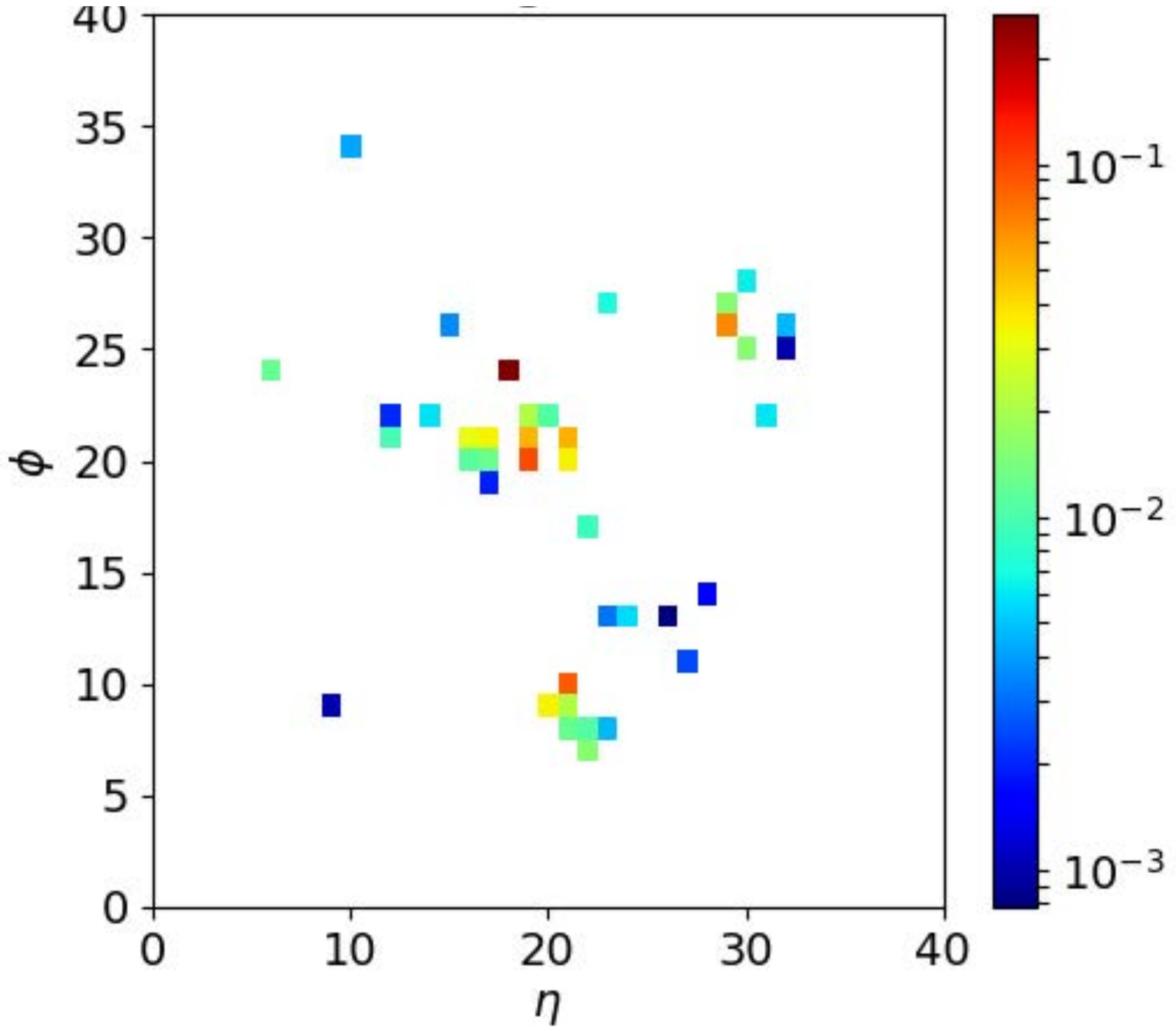
DATA REPRESENTATION: IMAGE

HEP



Collision events, detector hits, sensor arrays, …

Image

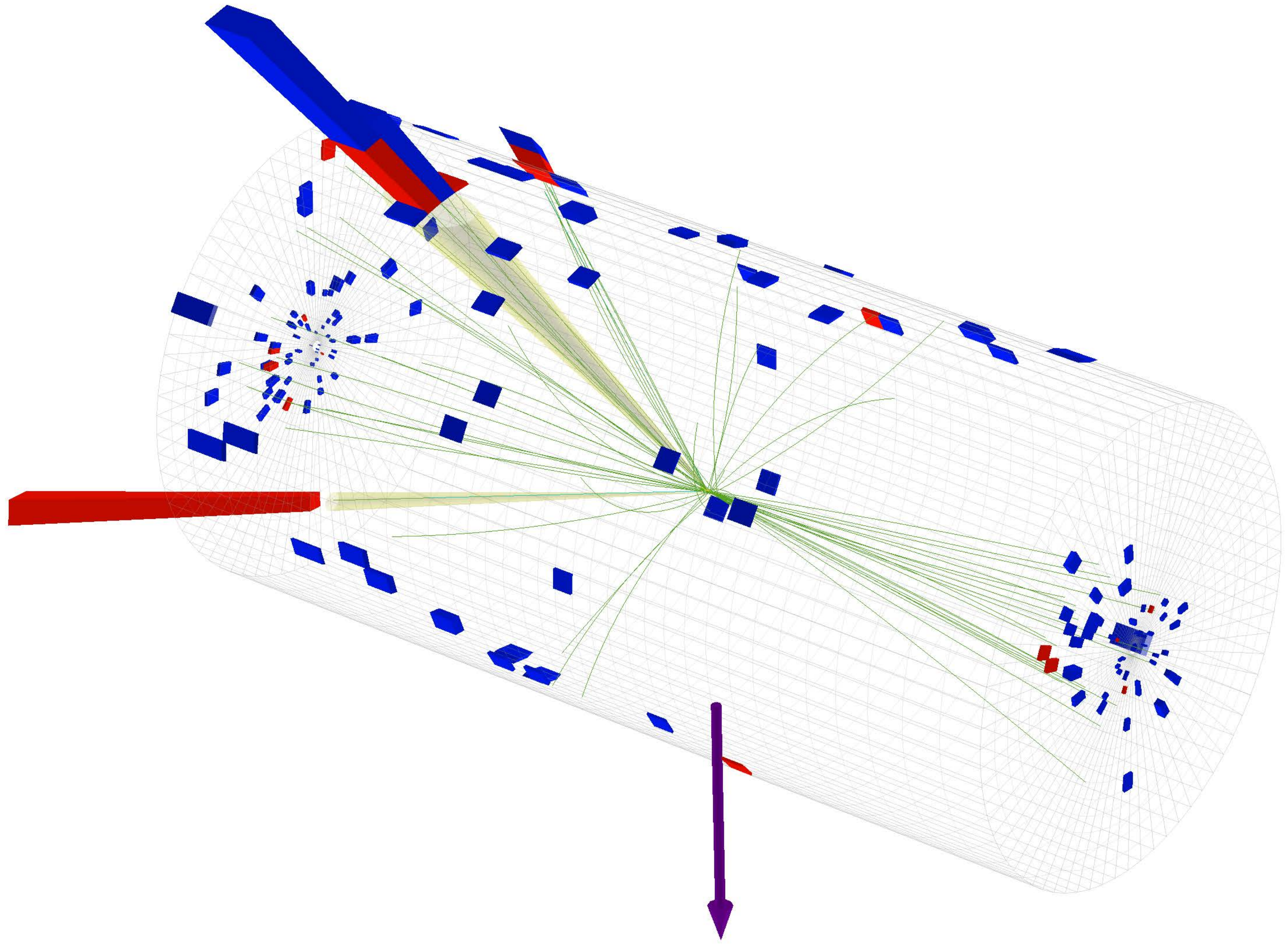


e.g., de Oliveira, Kagan, Mackey, Nachman and Schwartzman, arXiv:1511.05190

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

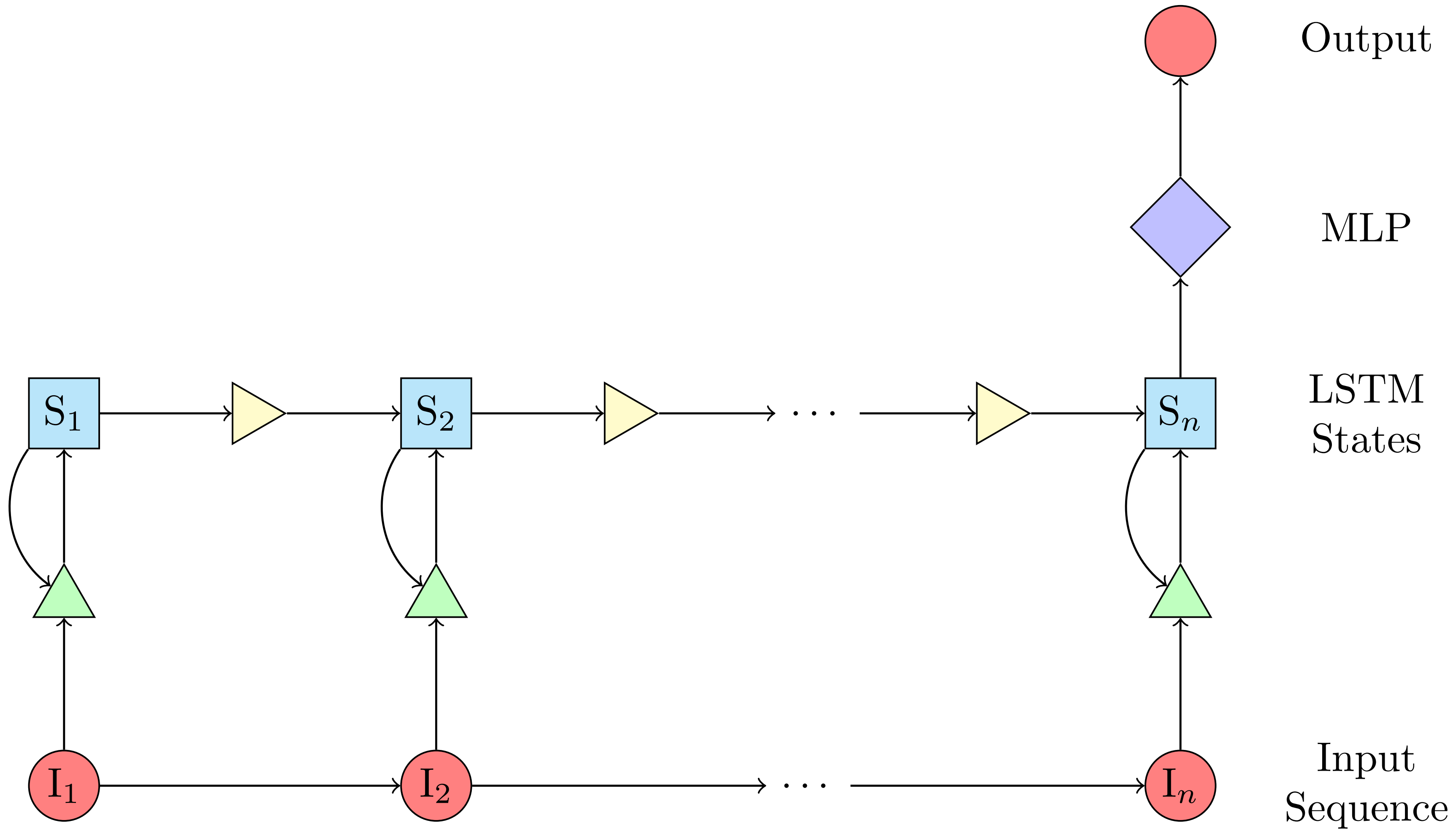
Data Representation: Sequence

HEP



Collision events, detector hits, sensor arrays, …

Sequence

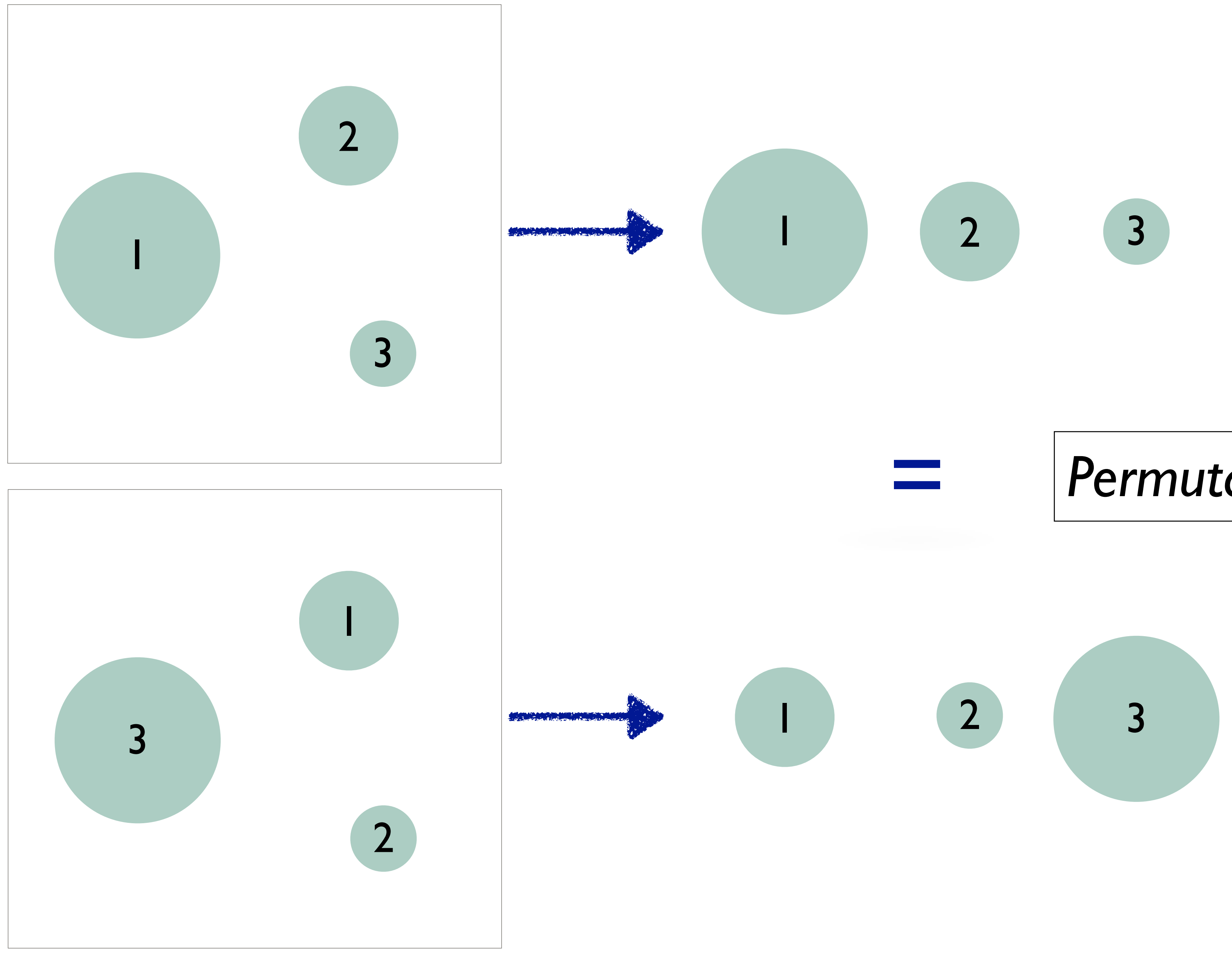


e.g., Guest, Collado, Baldi, Hsu, Urban, Whiteson
arXiv: 1607.08633

- Convert to a sequence => **Natural language processing (NLP)**
  - recurrent neural network (RNN), e.g., GRU/LSTM; 1D CNNs; etc.

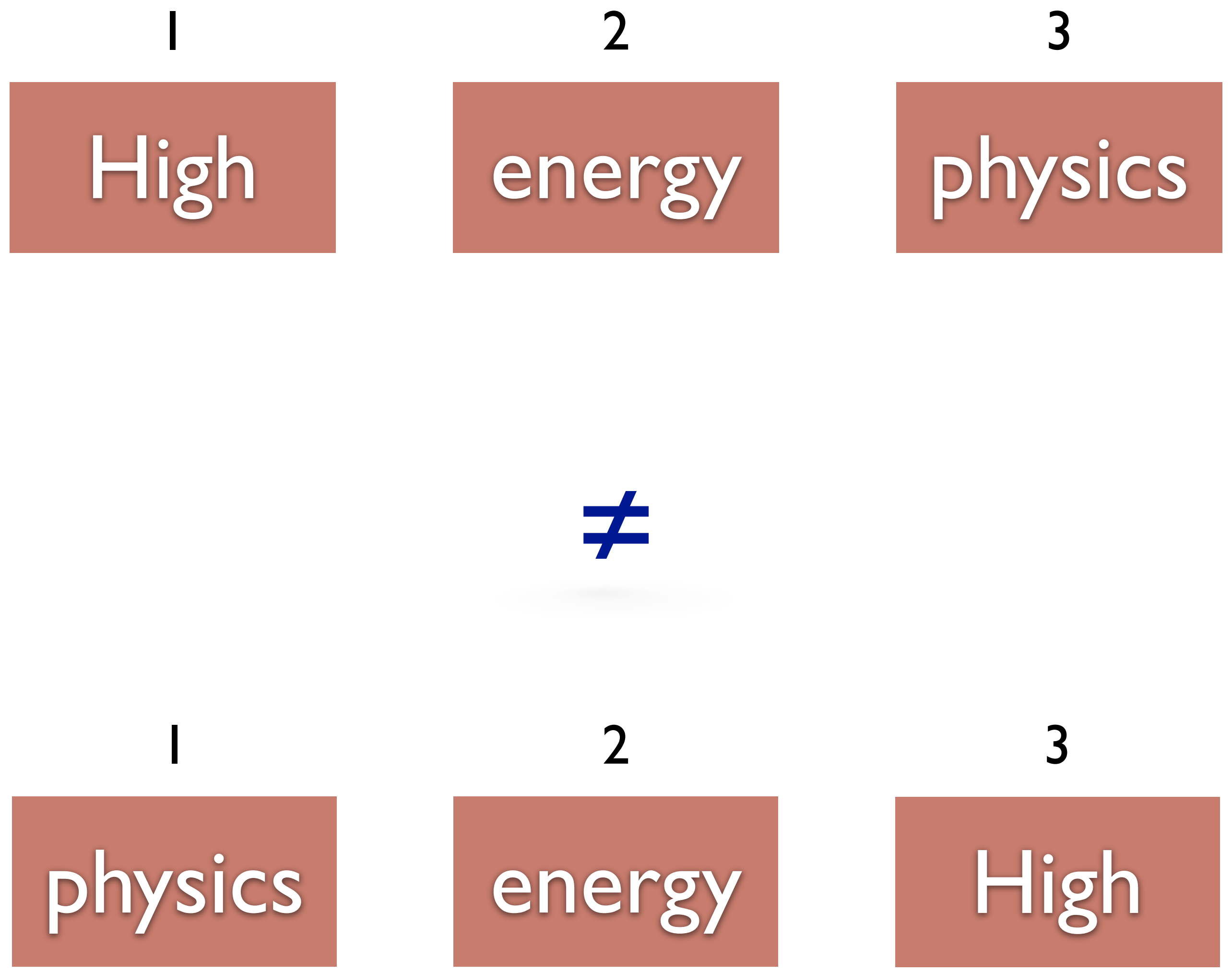
Data Representation: Sequence?

HEP



= Permutation symmetry

Sequence



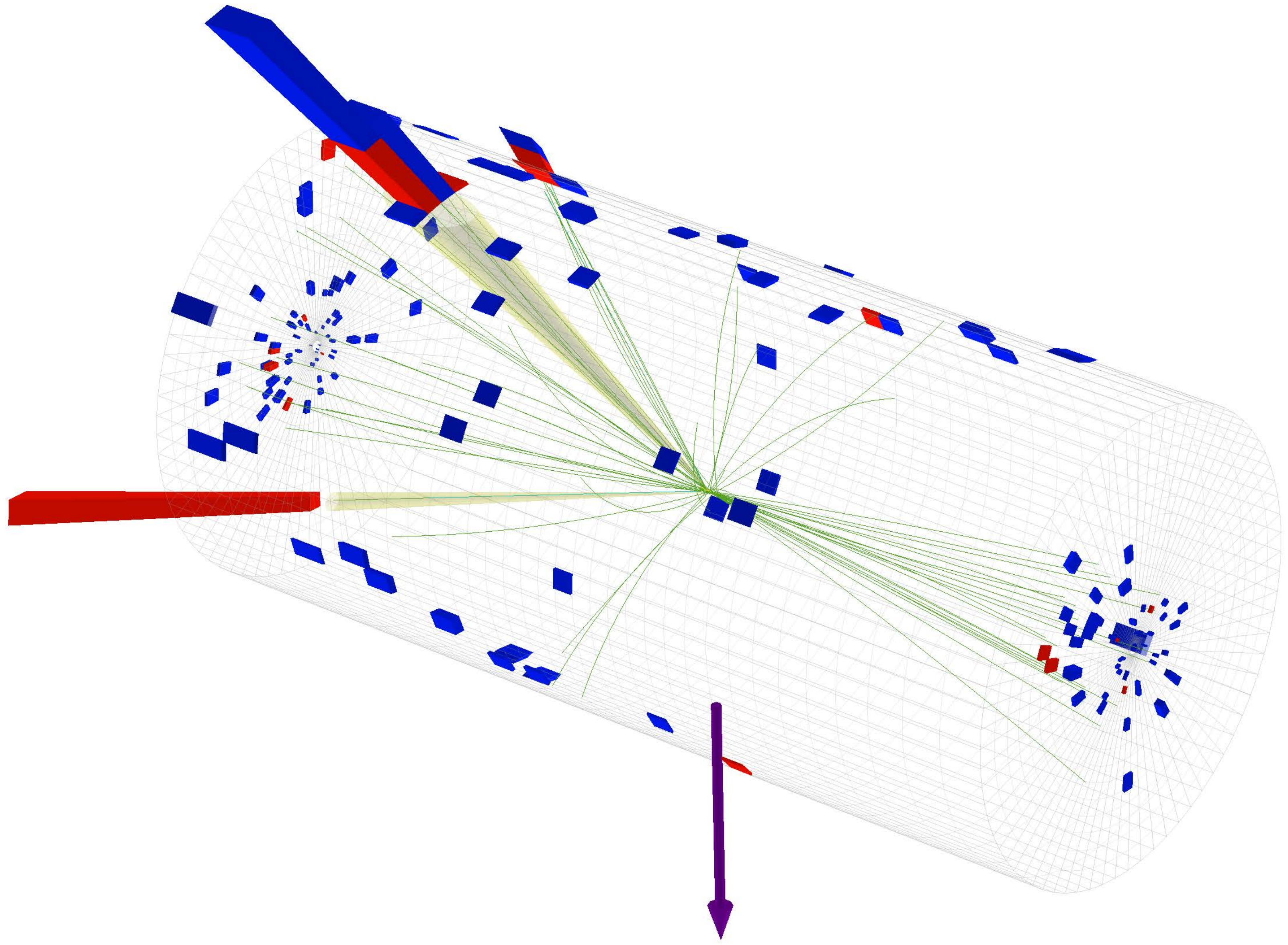
- Convert to a sequence => **Natural language processing (NLP)**
  - recurrent neural network (RNN), e.g., GRU/LSTM; 1D CNNs; etc.
  - but:
    - must impose an ***ordering*** on the particles/hits, which can limit the learning performance

POINT CLOUD

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

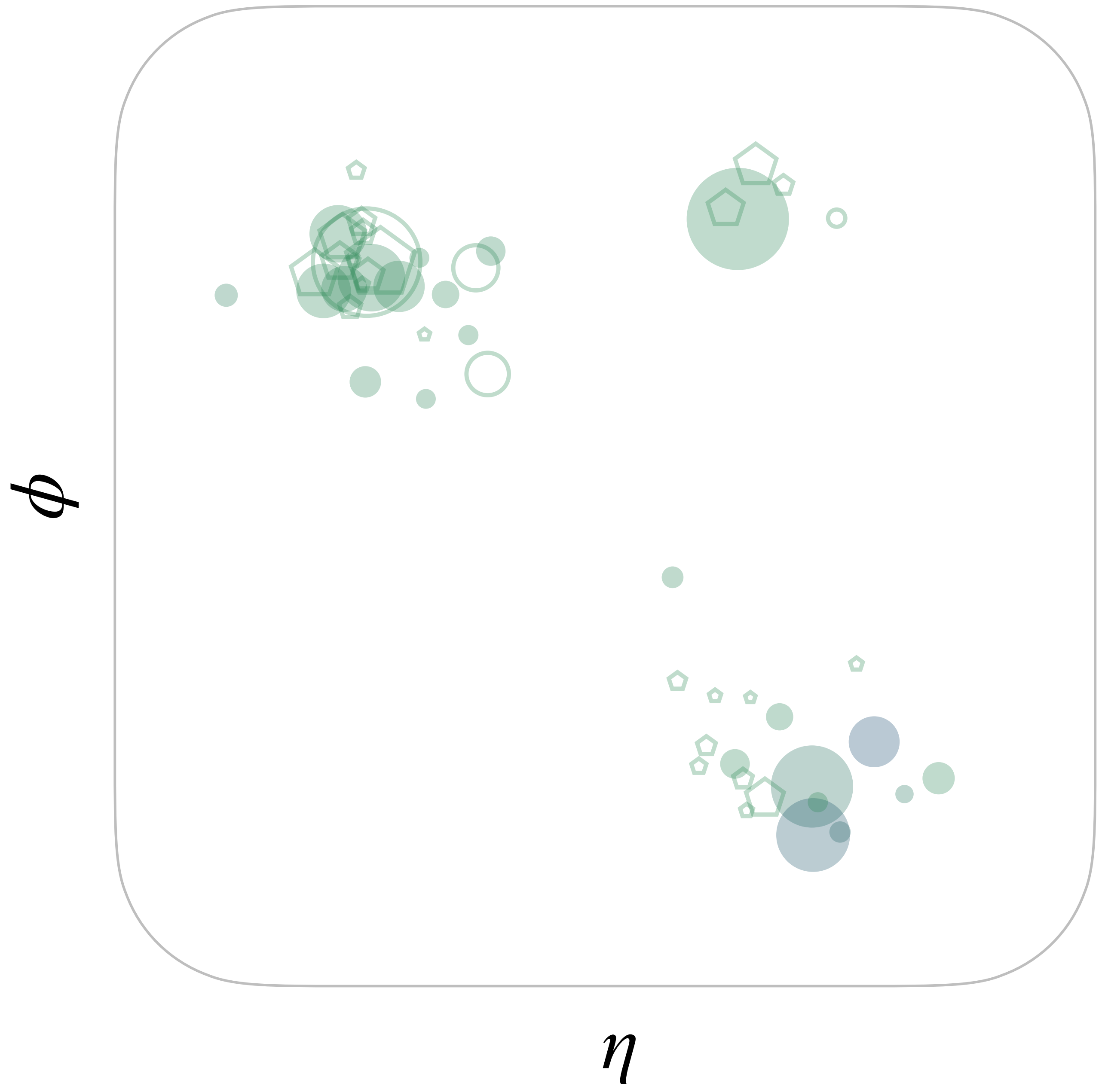
Data Representation: Point Cloud

HEP



Collision events, detector hits, sensor arrays, …

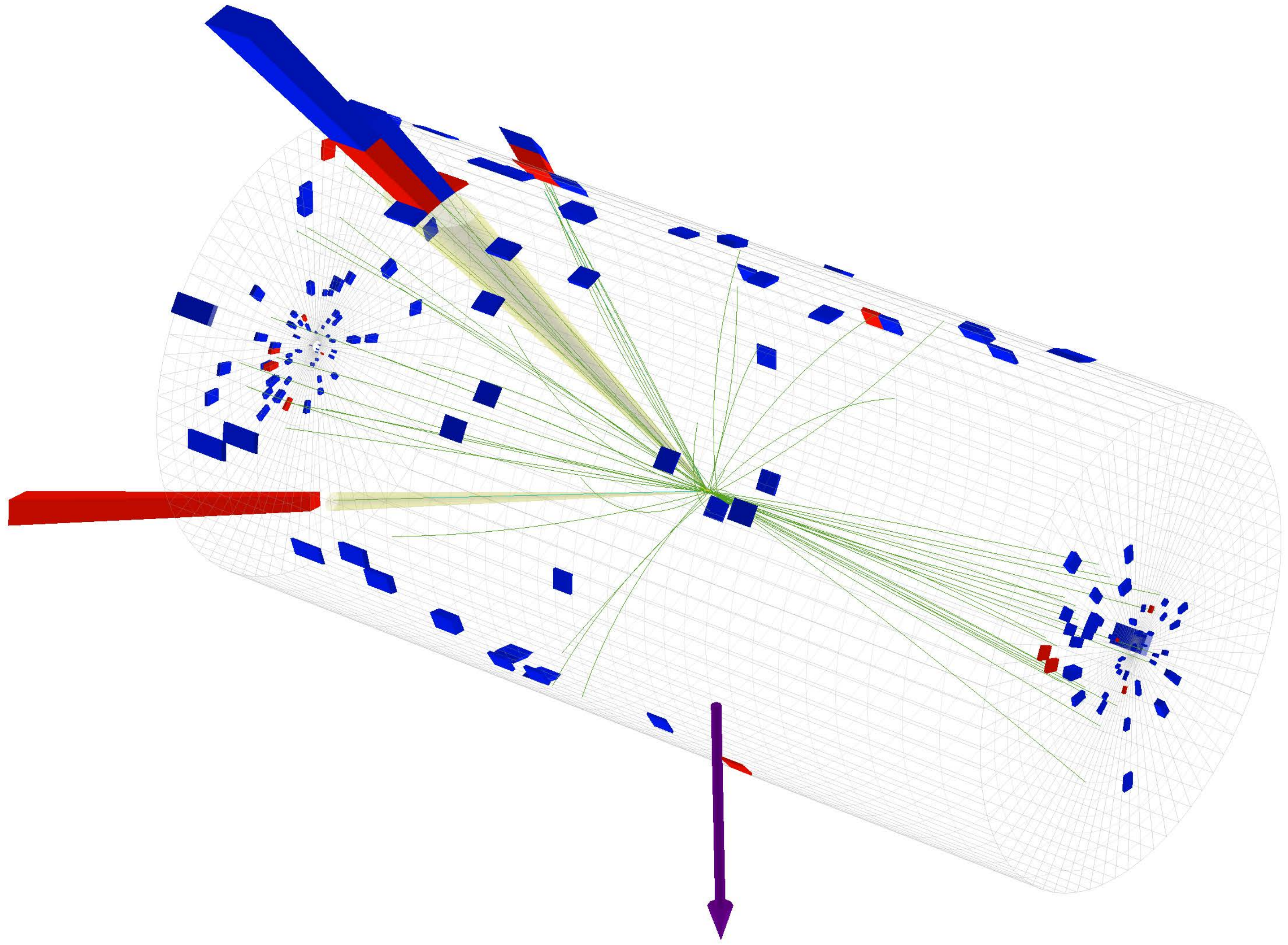
Point cloud



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

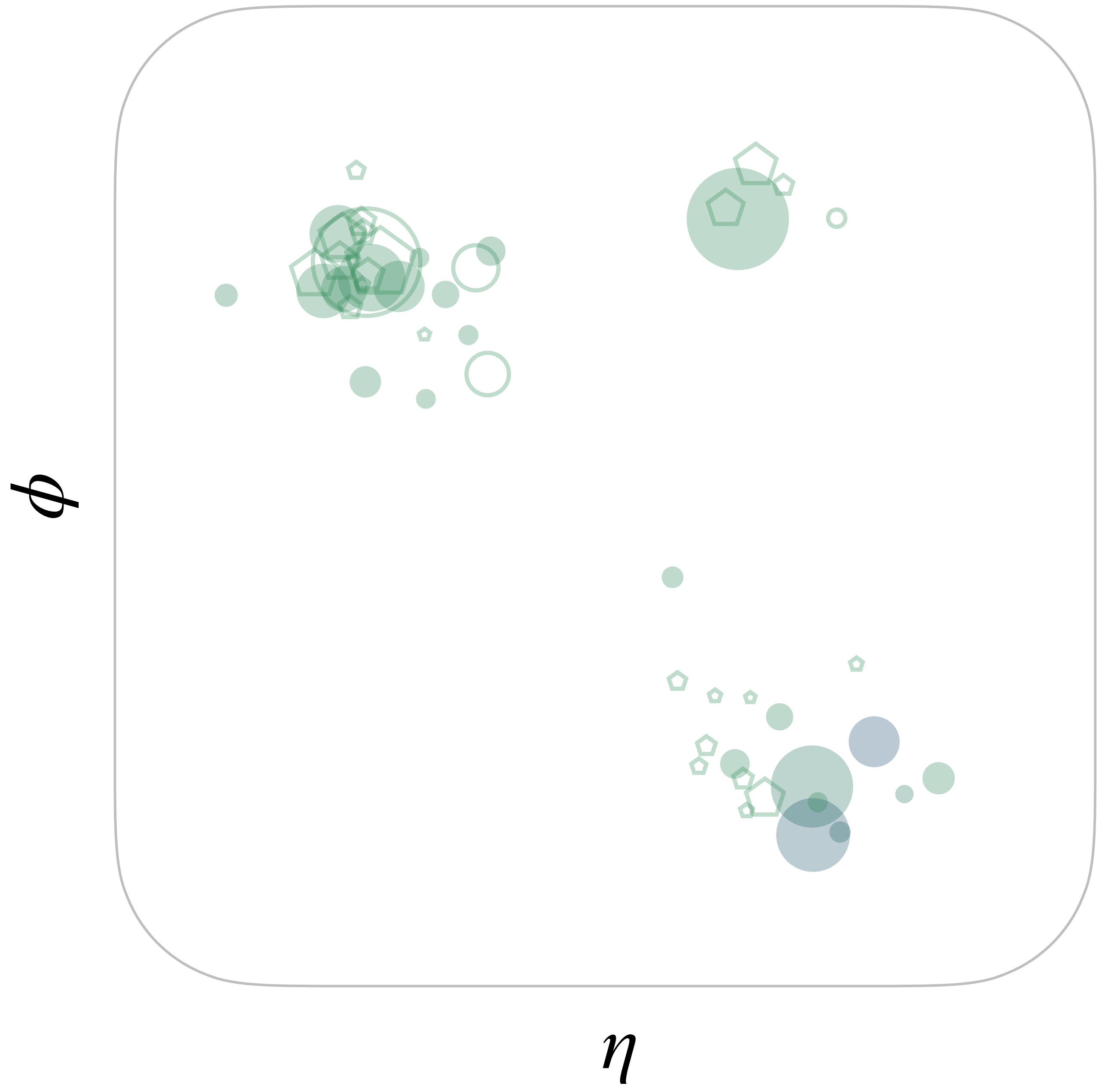
LEARNING ON POINT CLOUDS

HEP



Collision events, detector hits, sensor arrays, …

Point cloud

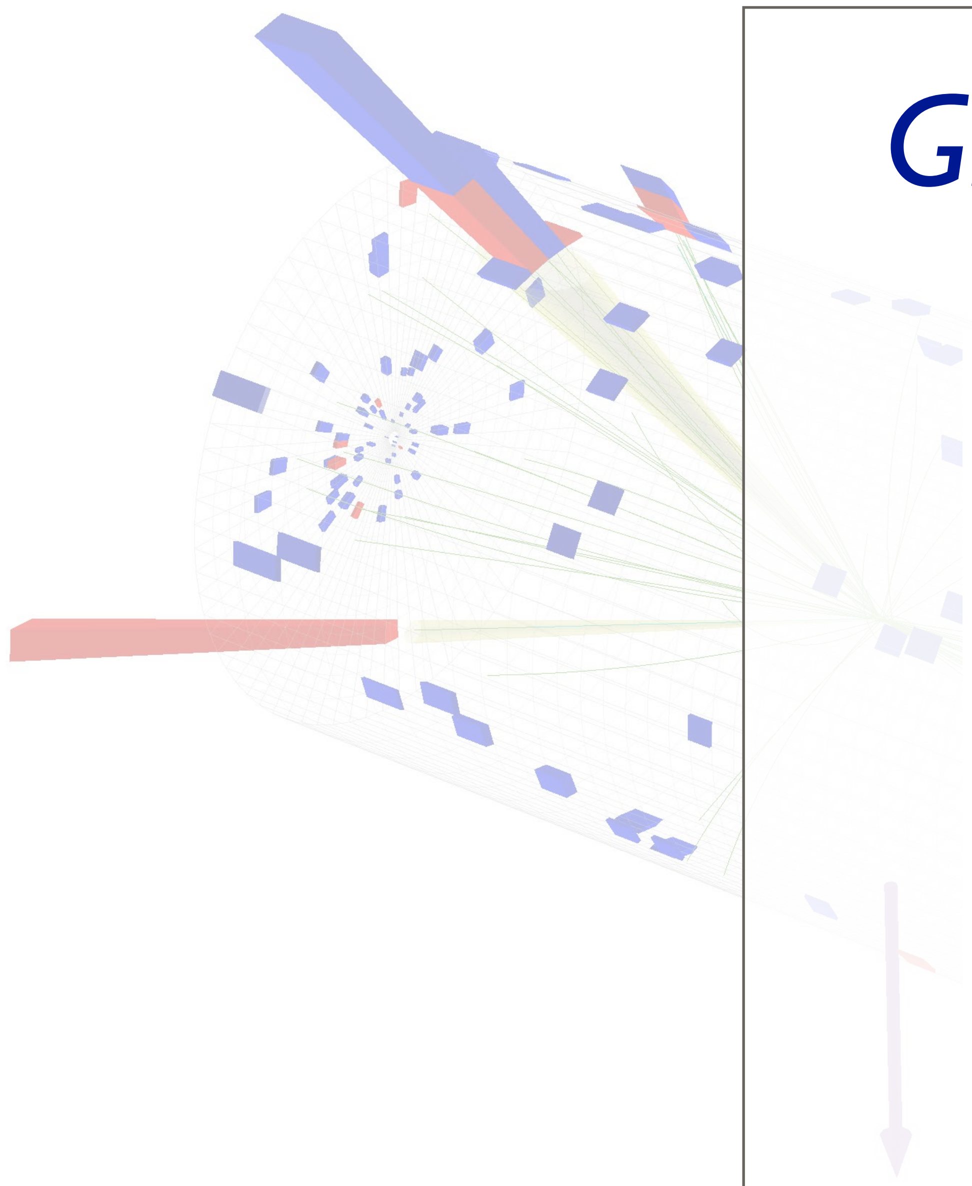


- Desired algorithms for learning on point cloud data
  - symmetry-preserving: the outputs should be invariant under permutation of the points
  - high expressiveness: capable of fully exploiting the correlations between points
  - low computational cost: scalable from O(10) to O(1000) points, and even up to O(1M) points in some cases

Learning on Point Clouds

HEP

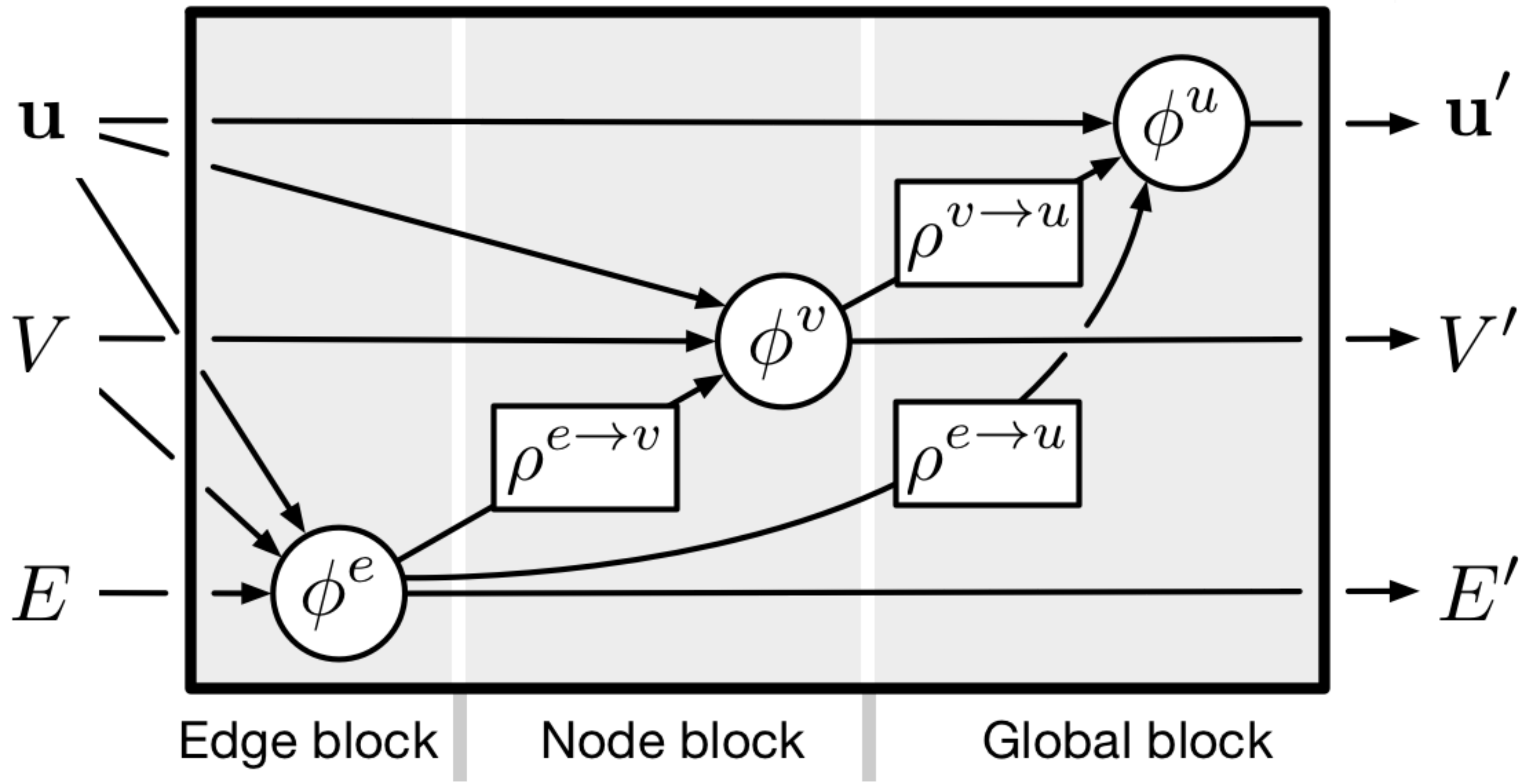
Point cloud



Collision events, detector hi

η

Graph neural network - A unified framework



Review in Shlomi, Battaglia, Vlimant, arXiv:2007.13681

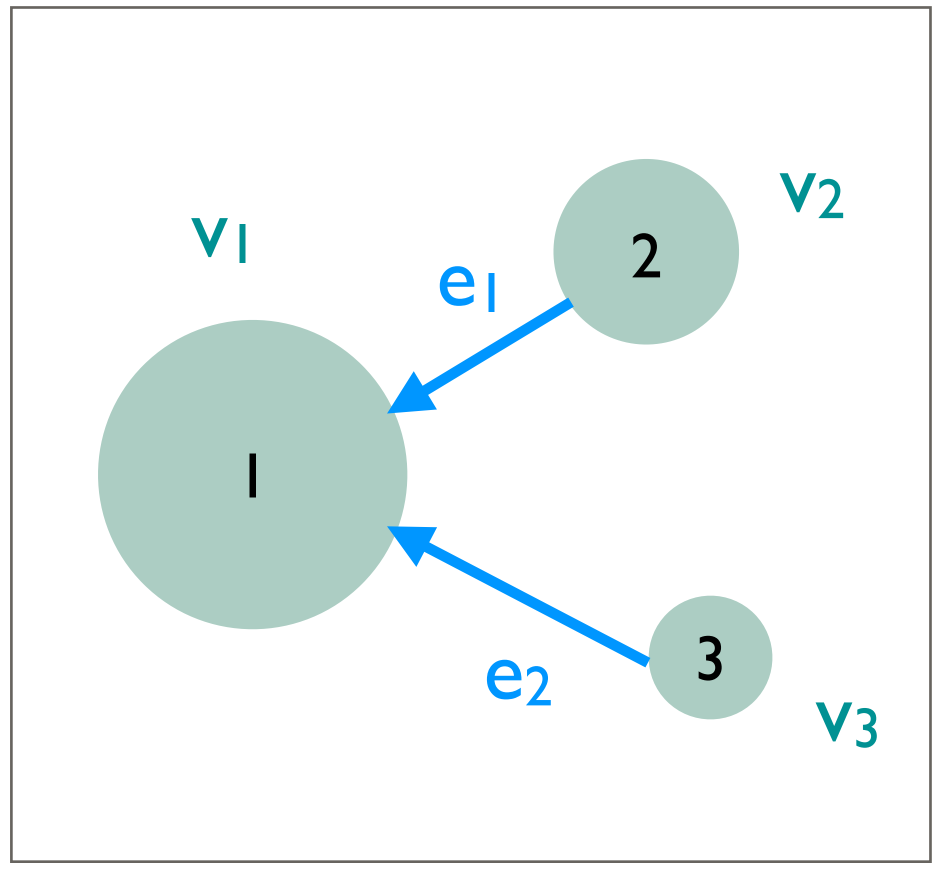
- Desired algorithms for learning on point cloud data
  - symmetry-preserving: the outputs should be invariant under permutation of the points
  - high expressiveness: capable of fully exploiting the correlations between points
  - low computational cost: scalable from O(10) to O(1000) points, and even up to O(1M) points in some cases

A Journey Through Graph Neural Networks

What is a 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)

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

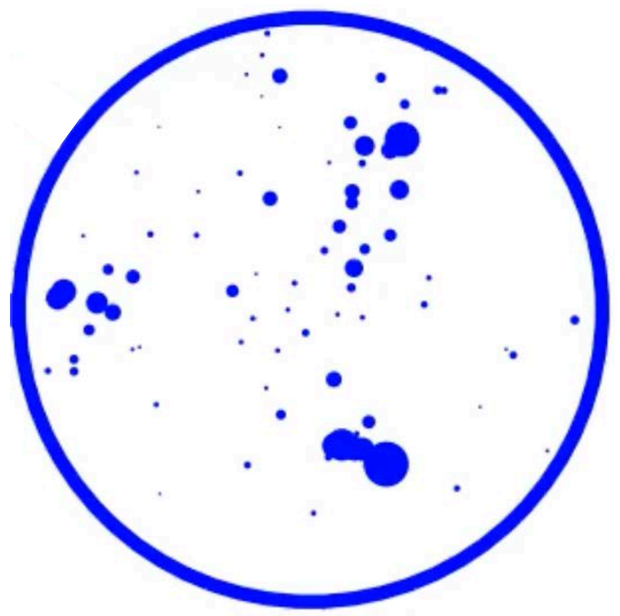
attributes of the k-th edge

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

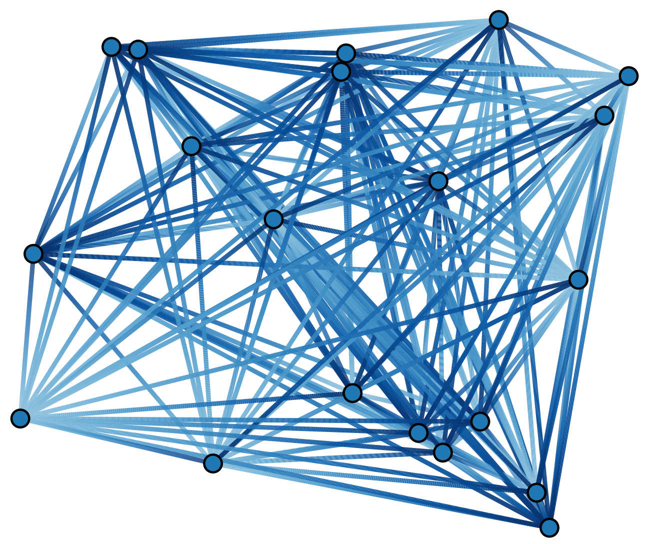
How to Build the Graph?

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

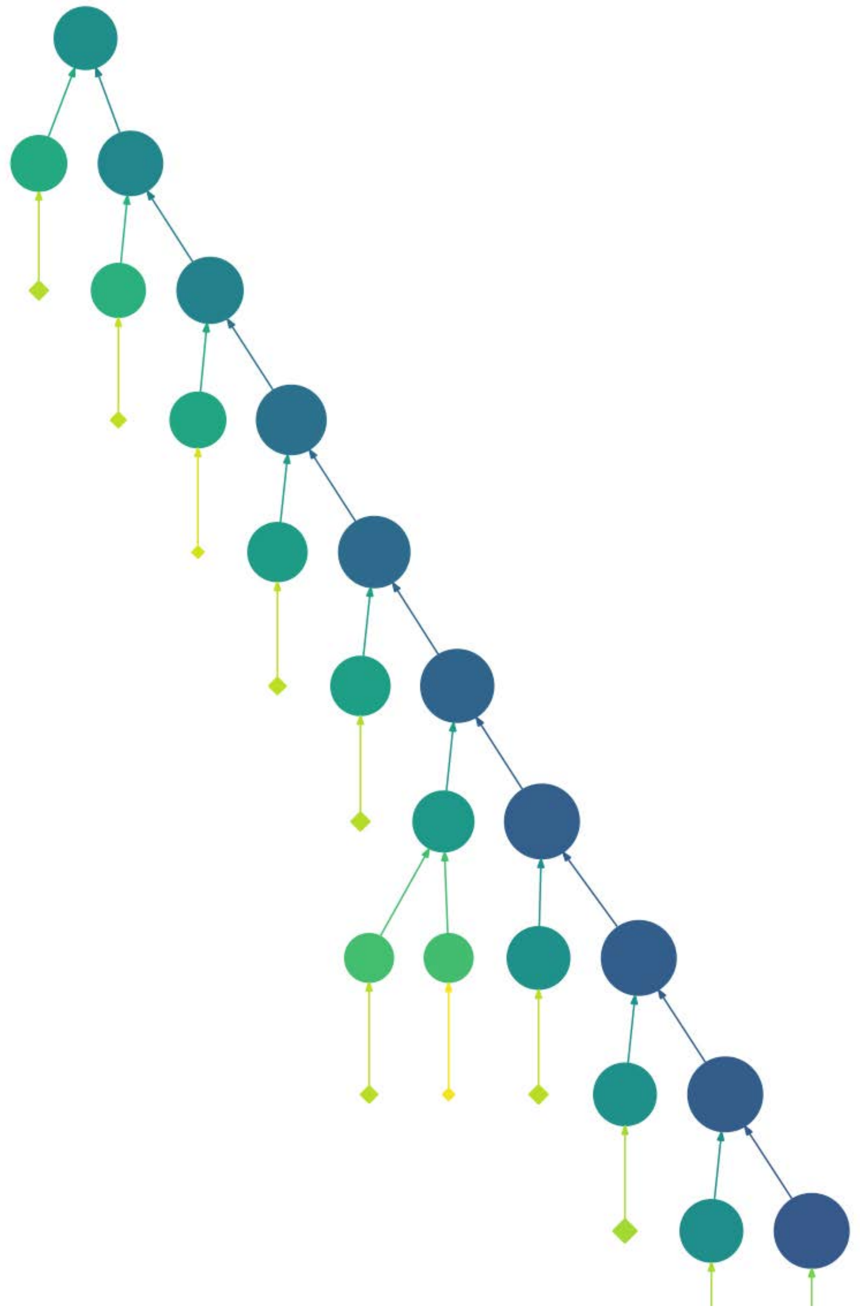
Set: no edges



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

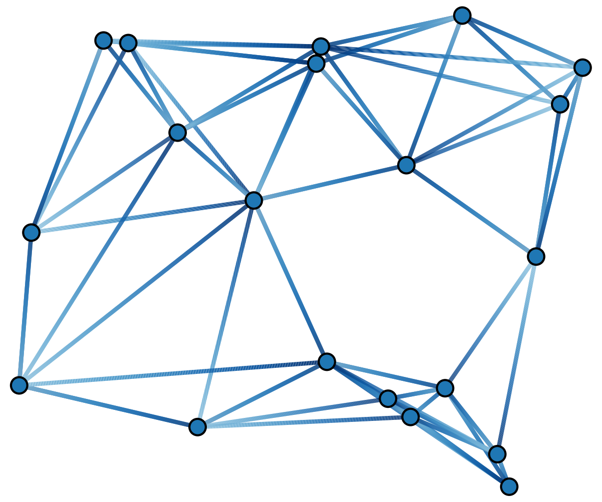


Hierarchical trees:
- *decay chain*
- *jet clustering history*

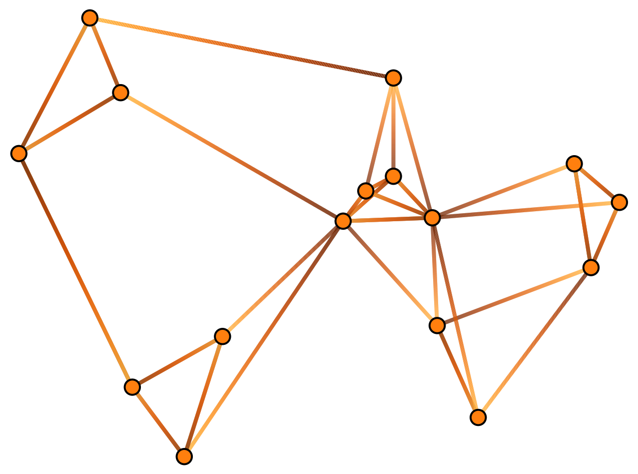


Locally connected graph
- *i.e., connect each node only to neighbor nodes*
  - *k-nearest neighbors*
  - *fixed radius*

static



(dynamically) learned



Graph Network Formalism

- Typical graph neural networks (GNNs) can be described in the "Message Passing" framework

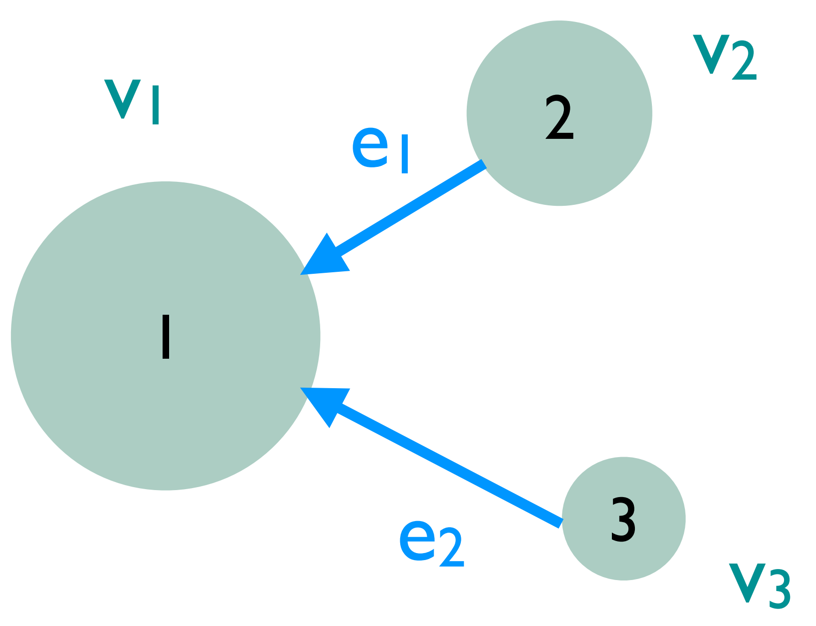
Graph: $G = (\mathbf{u}, V, E)$ with N_v vertices and N_e edges

Vertices (nodes)

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

Edges (links)

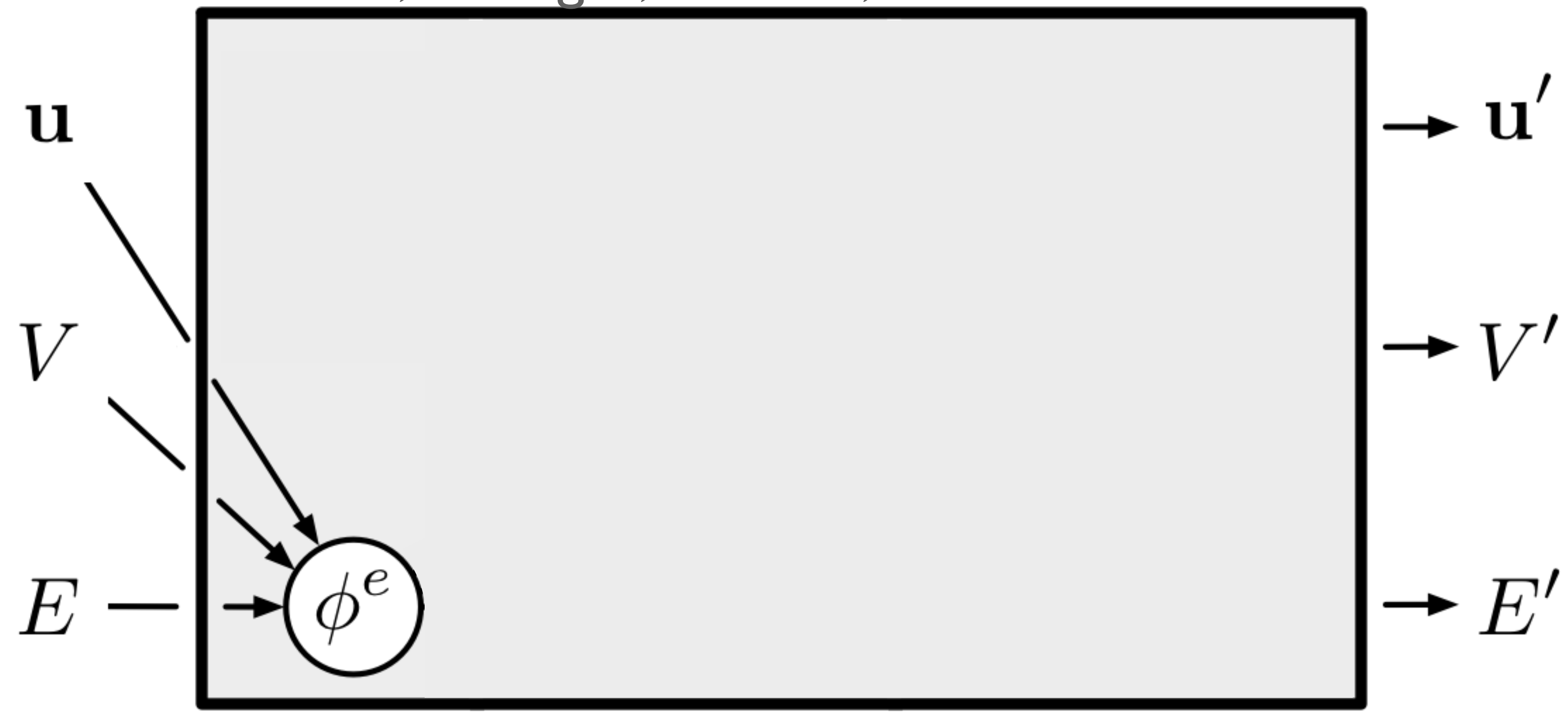
$$E = \{(\mathbf{e}_k, r_k, s_k)\}_{k=1:N_e}$$



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 graph neural networks (GNNs) can be described in the "Message Passing" framework

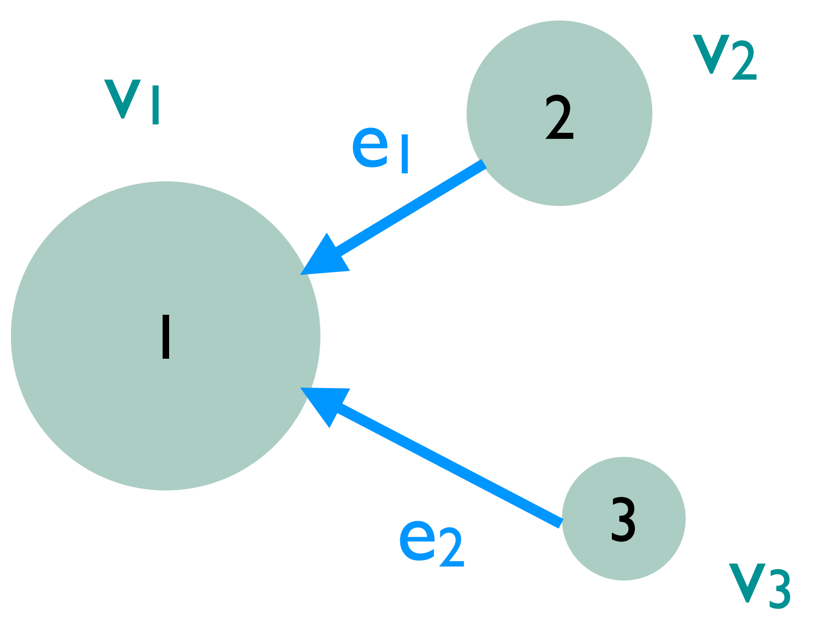
Graph: $G = (\mathbf{u}, V, E)$ with N_v vertices and N_e edges

Vertices (nodes)

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

Edges (links)

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



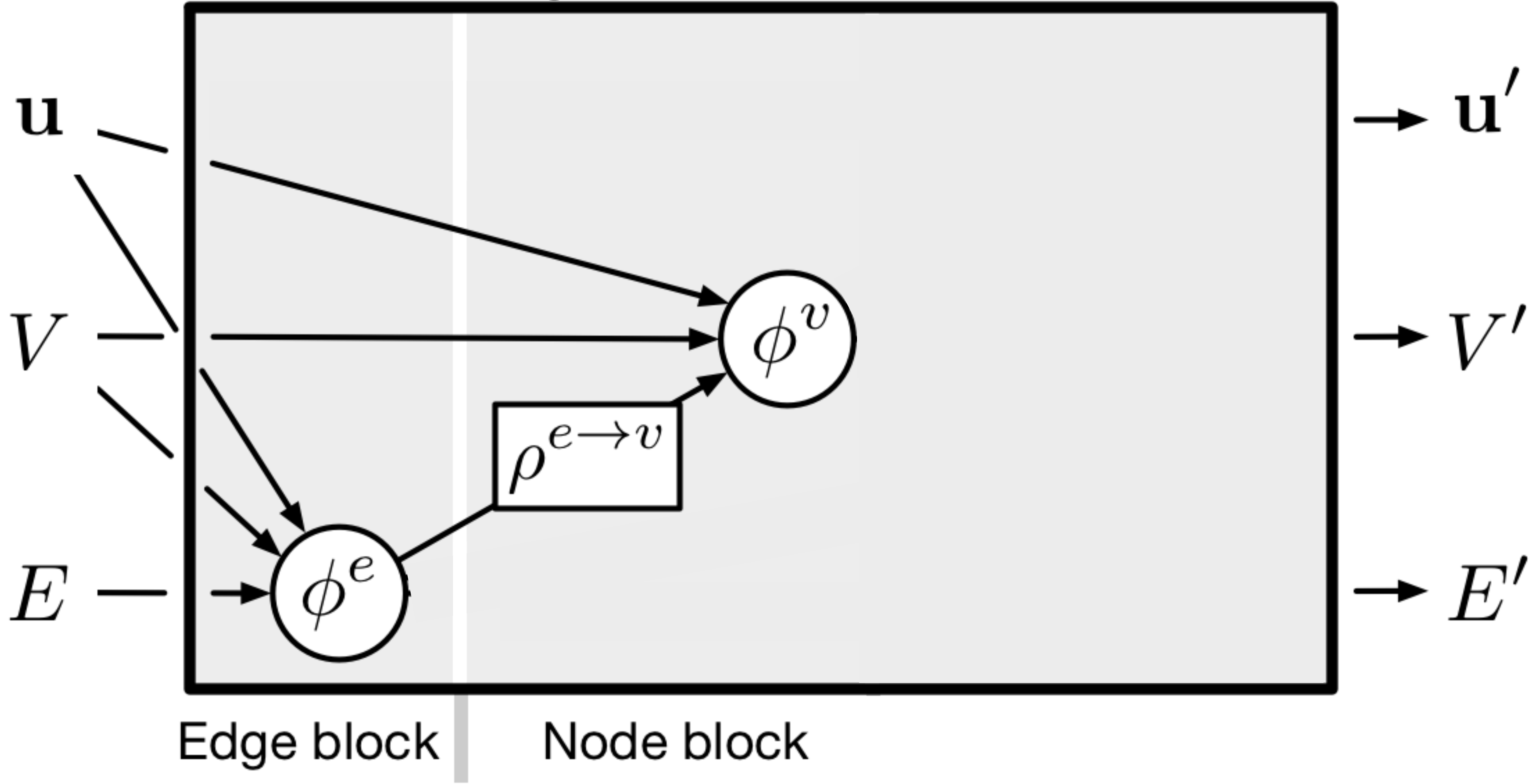
e'_k: message computed for edge k connecting nodes r_k, s_k

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

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

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

Shlomi, Battaglia, Vlimant, arXiv:2007.13681



Graph Network Formalism

- Typical graph neural networks (GNNs) can be described in the "Message Passing" framework

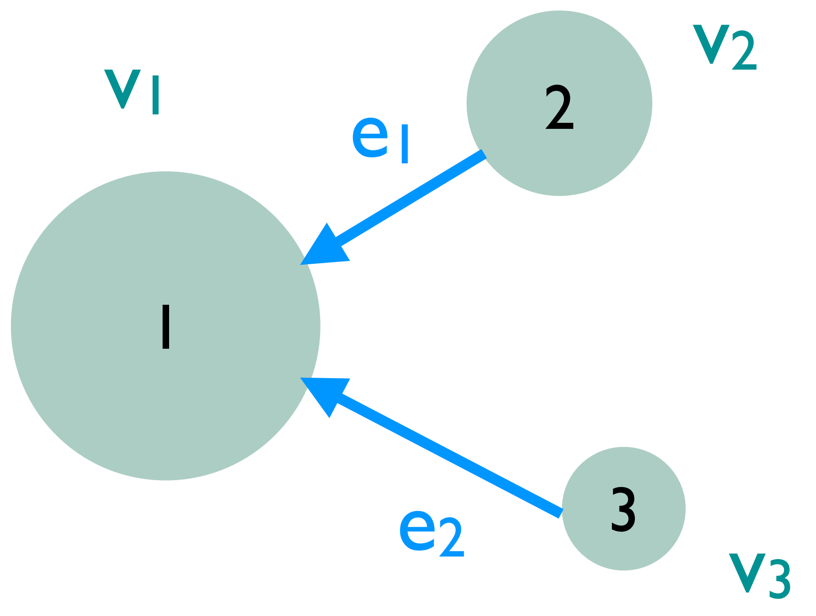
Graph: $G = (\mathbf{u}, V, E)$ with N_v vertices and N_e edges

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

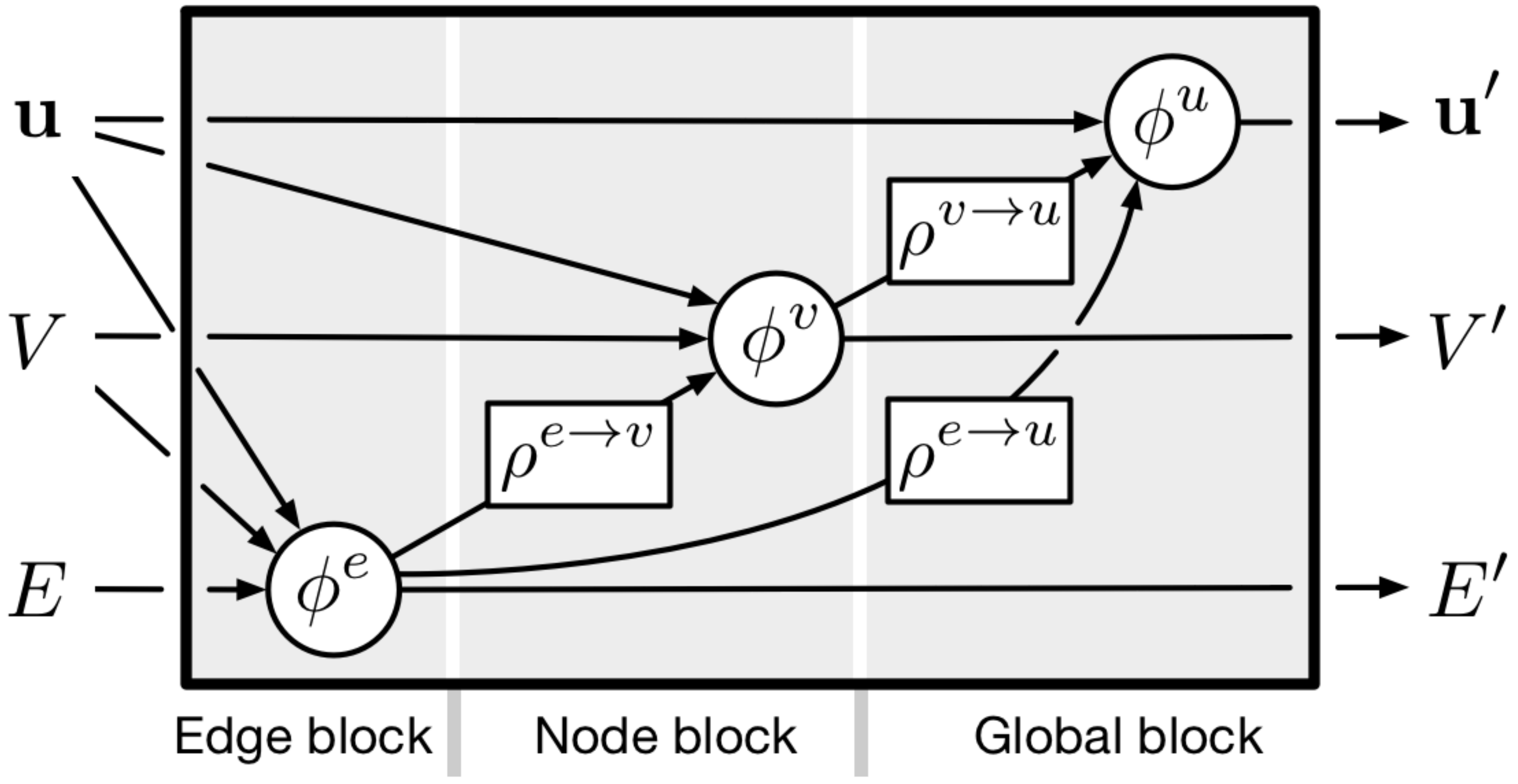
u': global feature update based on aggregated, updated node and edge features

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

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

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

Shlomi, Battaglia, Vlimant, arXiv:2007.13681



Graph Network Formalism

- Typical graph neural networks (GNNs) can be described in the "Message Passing" framework

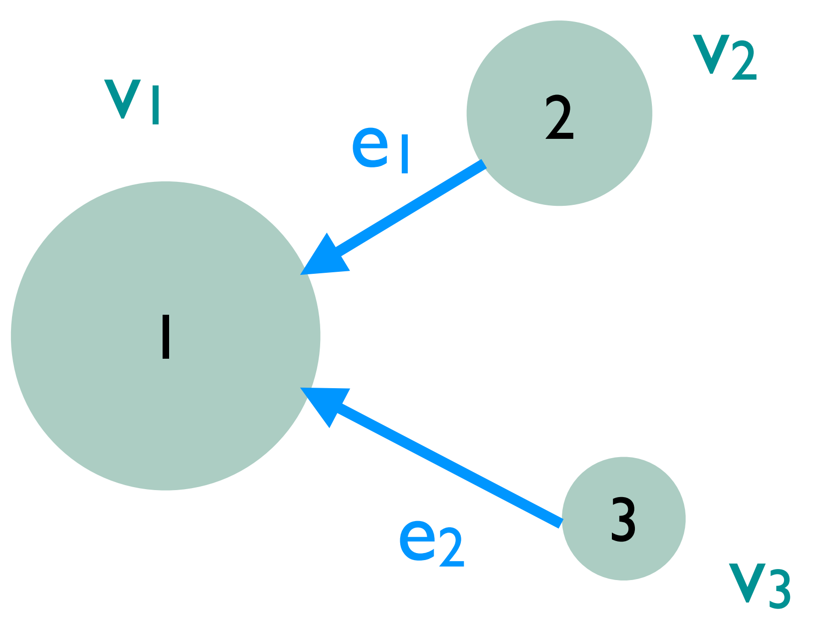
Graph: $G = (\mathbf{u}, V, E)$ with N_v vertices and N_e edges

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

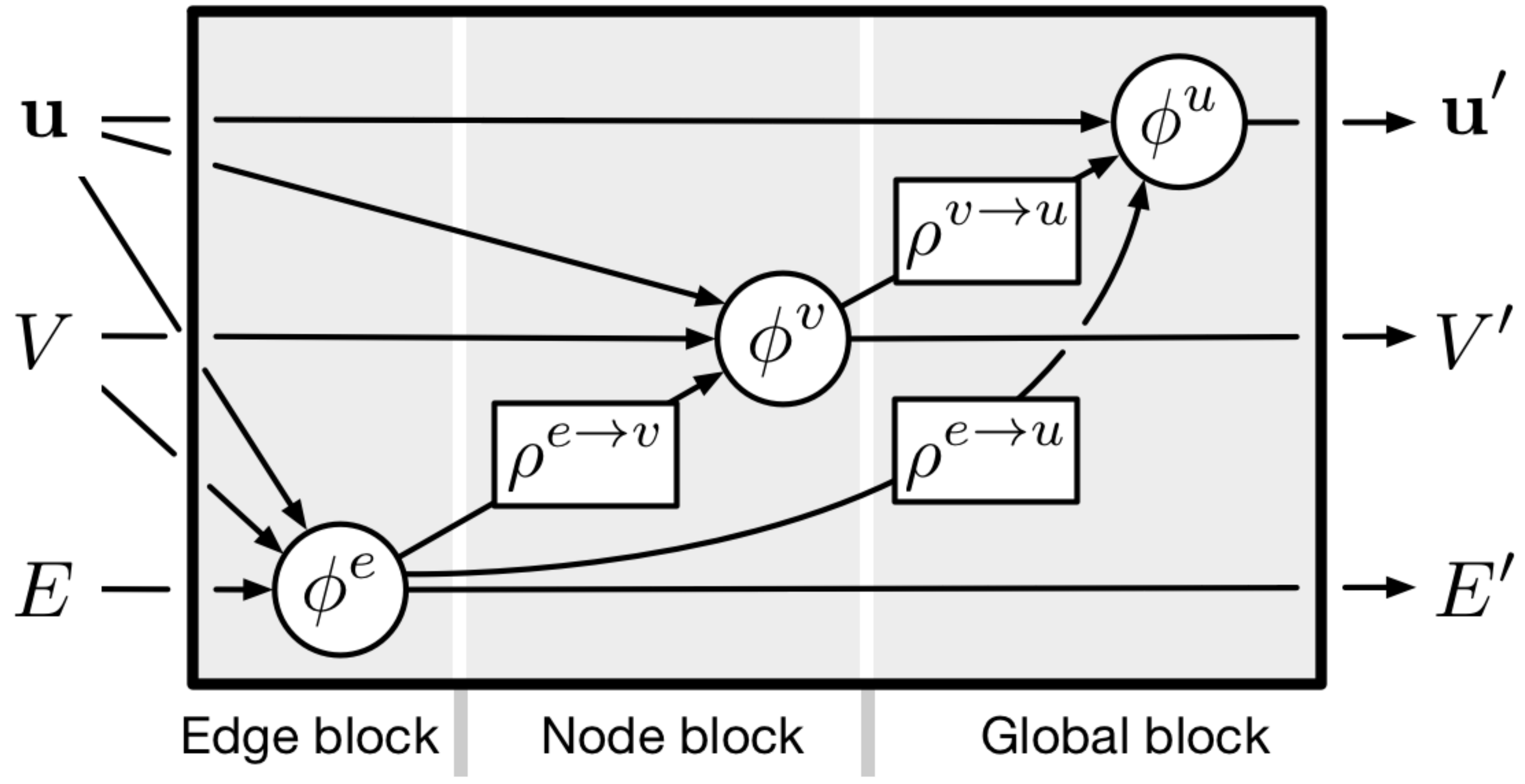
u': global feature update based on aggregated, updated node and edge features

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

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

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

Shlomi, Battaglia, Vlimant, arXiv:2007.13681



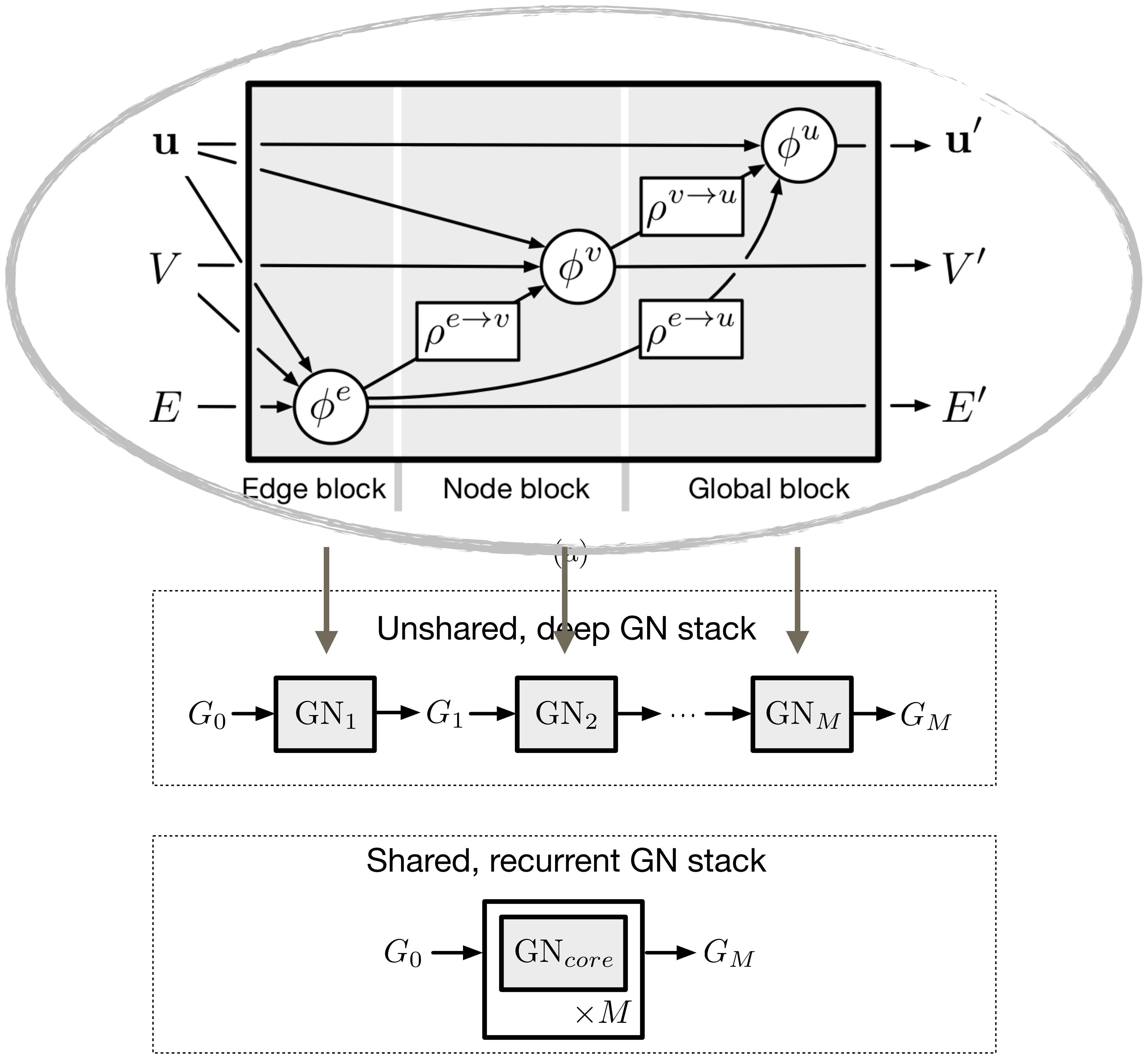
Shared-weight NN *Symmetric functions (e.g., sum, mean, max, etc.)* → *Permutation invariance*

Graph Network Formalism

- Typical graph neural networks (GNNs) can be described in the "Message Passing" framework

GN layer

Full GNN



Example: Deep Sets

Deep Sets

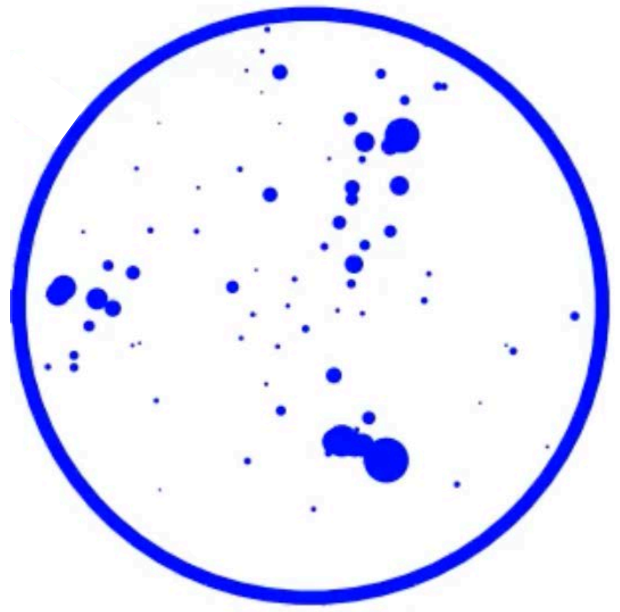
[1703.06114]

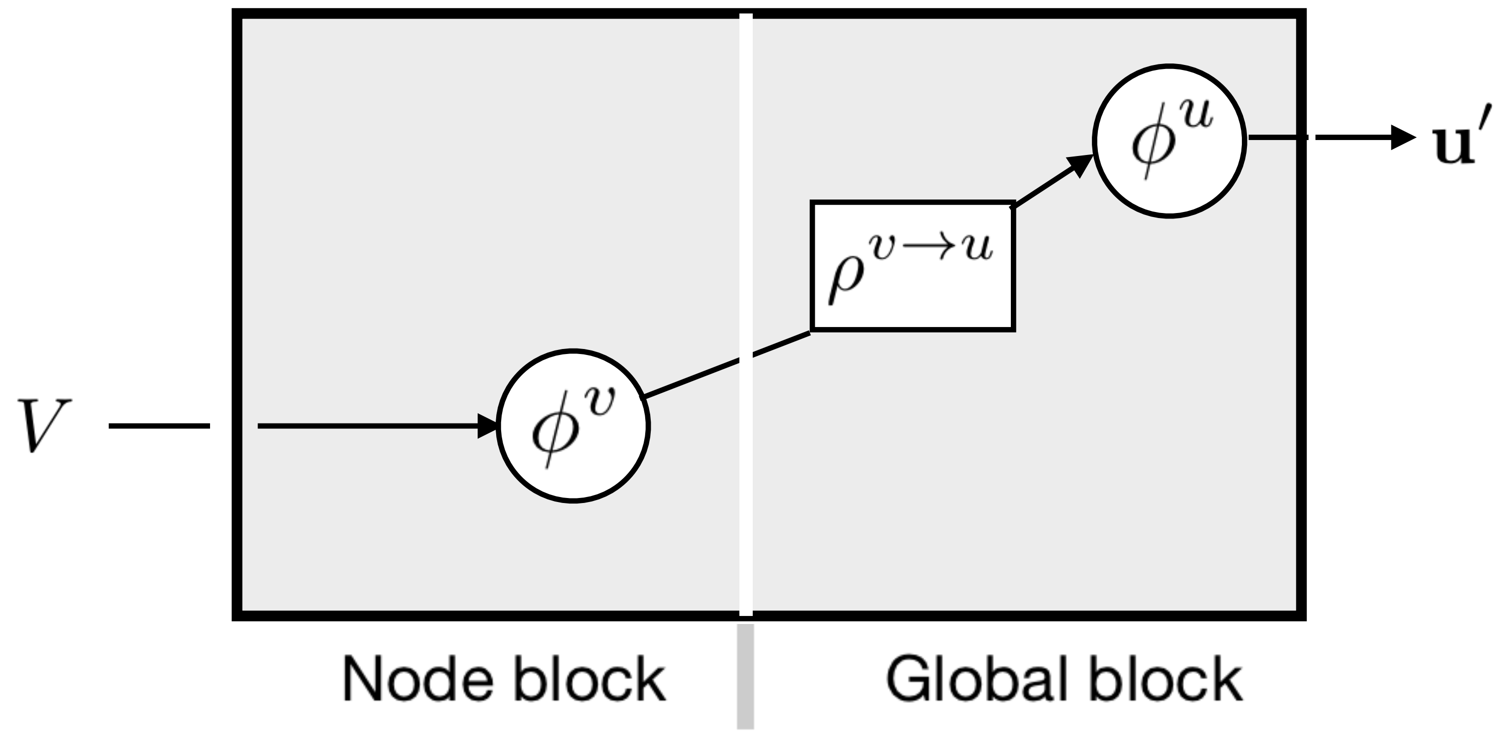
Manzil Zaheer[1,2], Satwik Kottur[1], Siamak Ravanbhakhsh[1],
Barnabás Póczos[1], Ruslan Salakhutdinov[1], Alexander J Smola[1,2]
[1] Carnegie Mellon University [2] Amazon Web Services

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]

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

Set: no edges



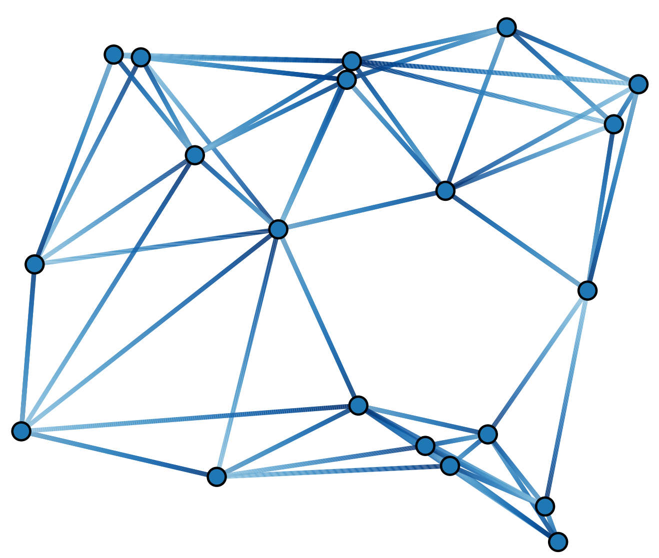


Example: Dynamic Graph CNN (DGCNN)

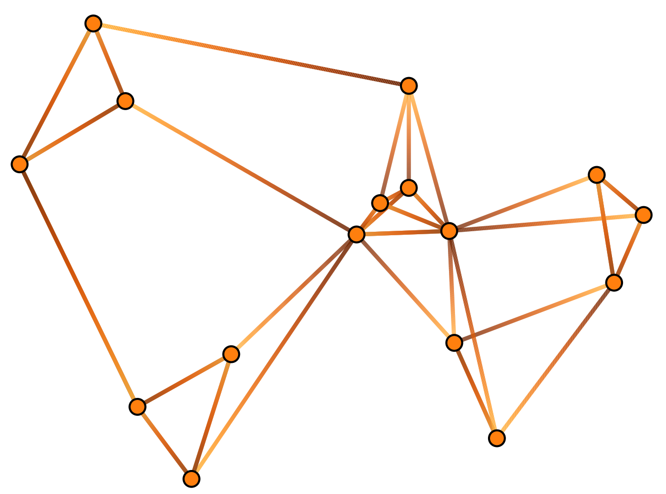
***Dynamic** locally connected graph*

- *k-nearest neighbors*

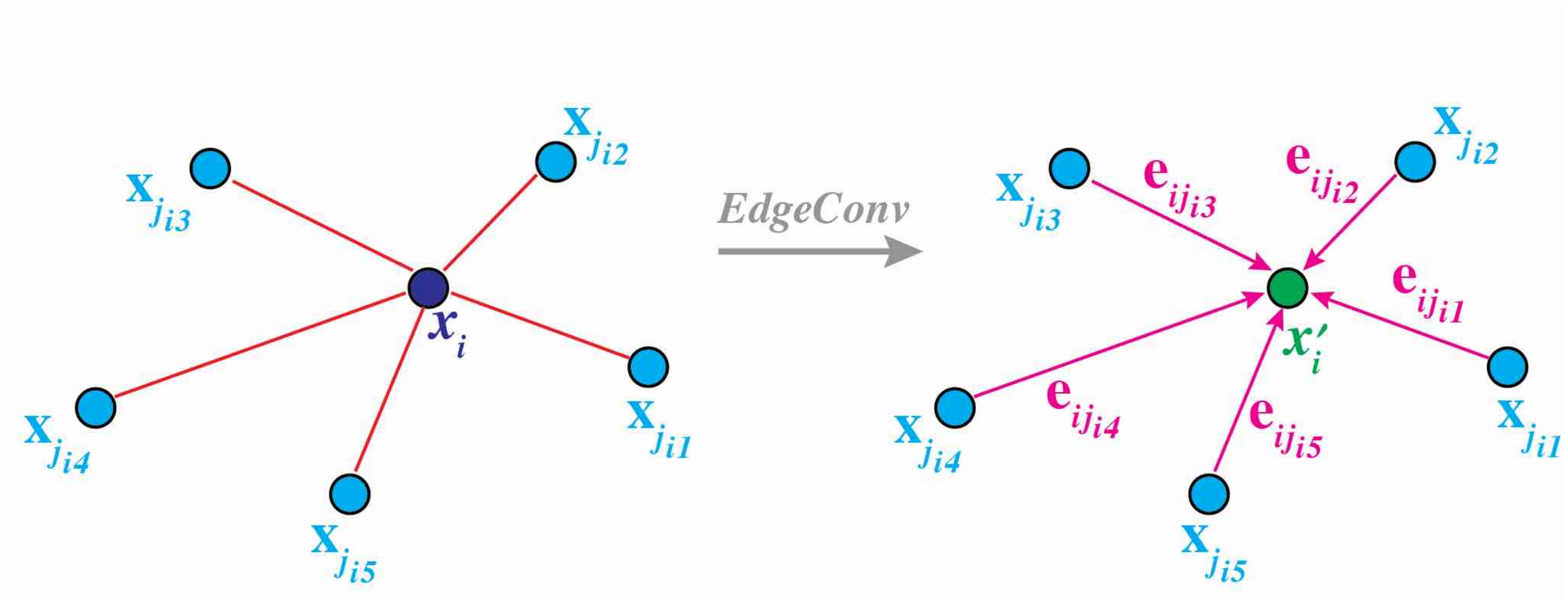
For the 1st layer:
kNN in input coordinates (xyz/η-ϕ)



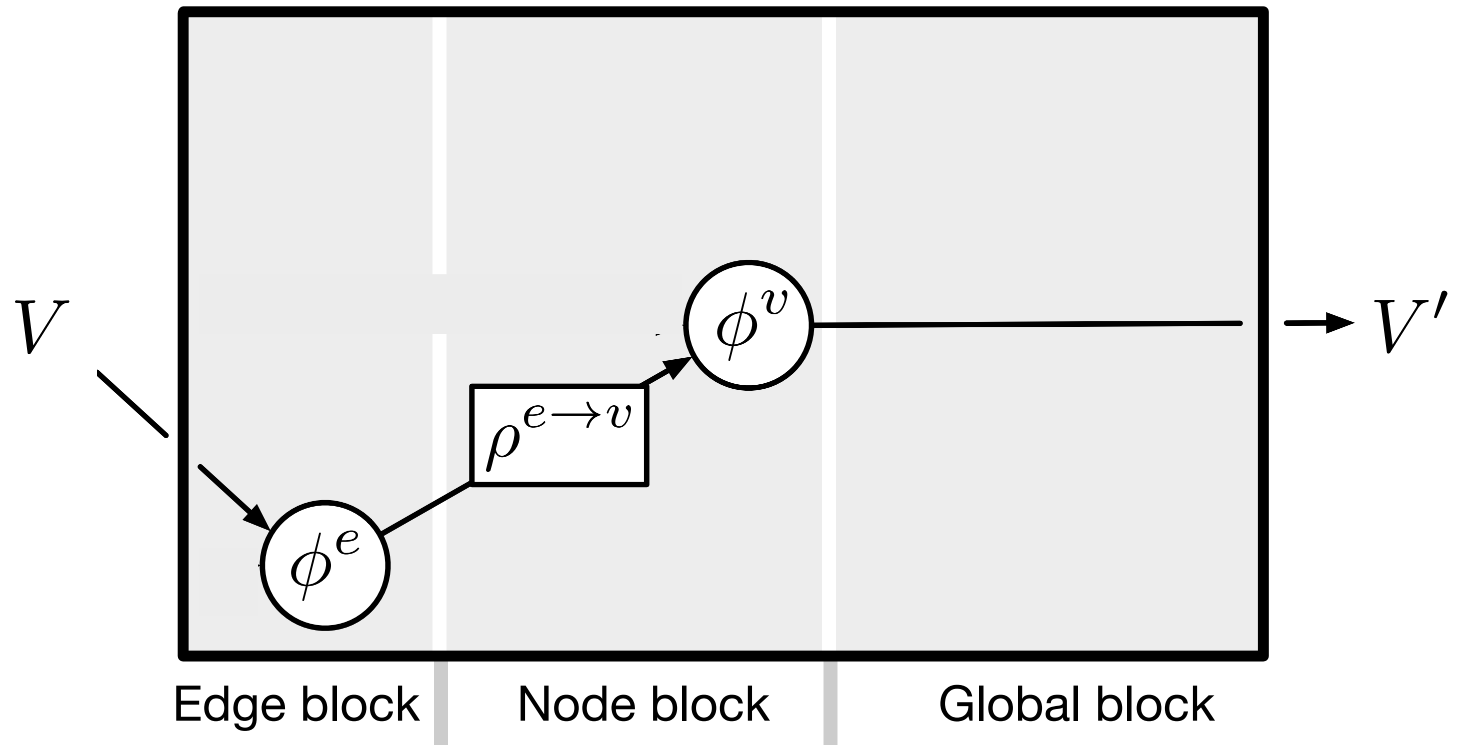
For subsequent layers:
kNN in learned latent space



Key building block: EdgeConv



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

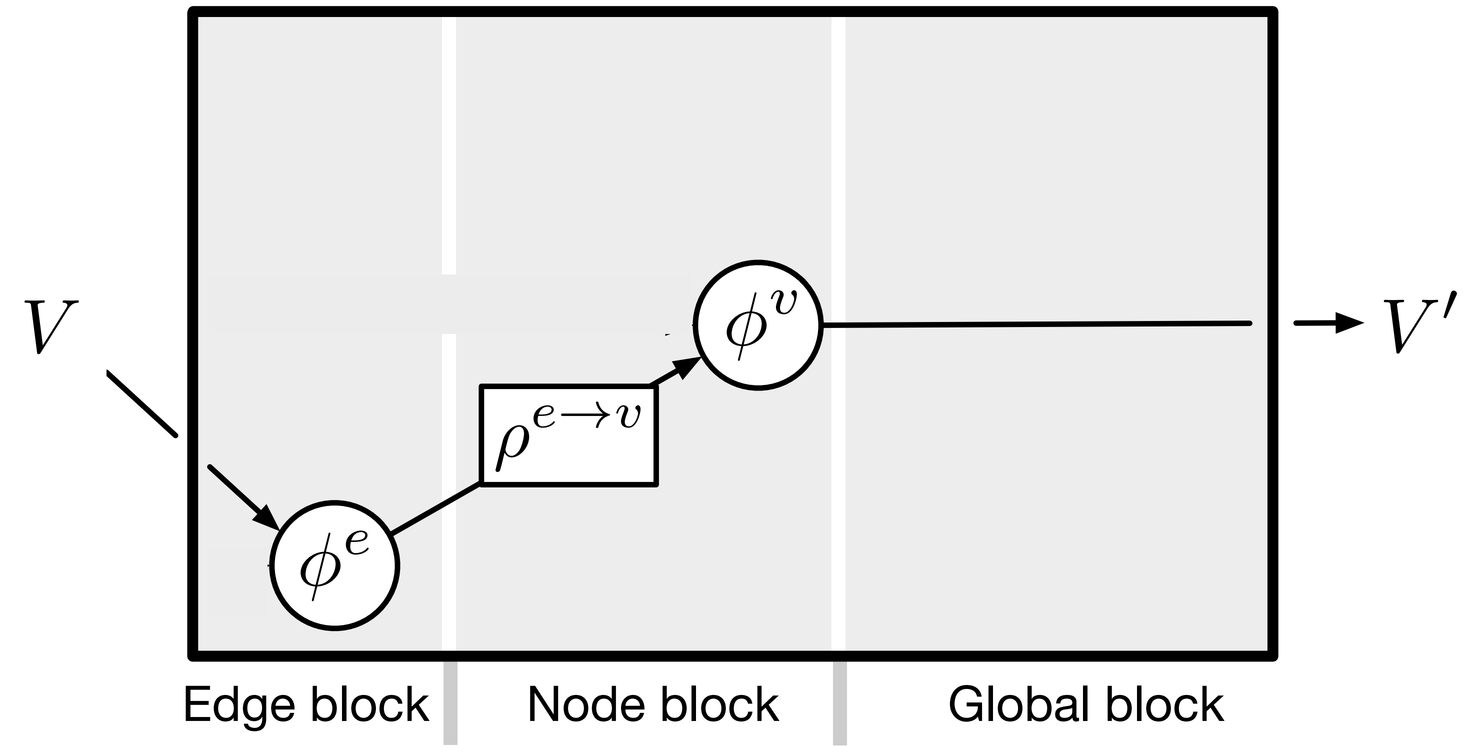


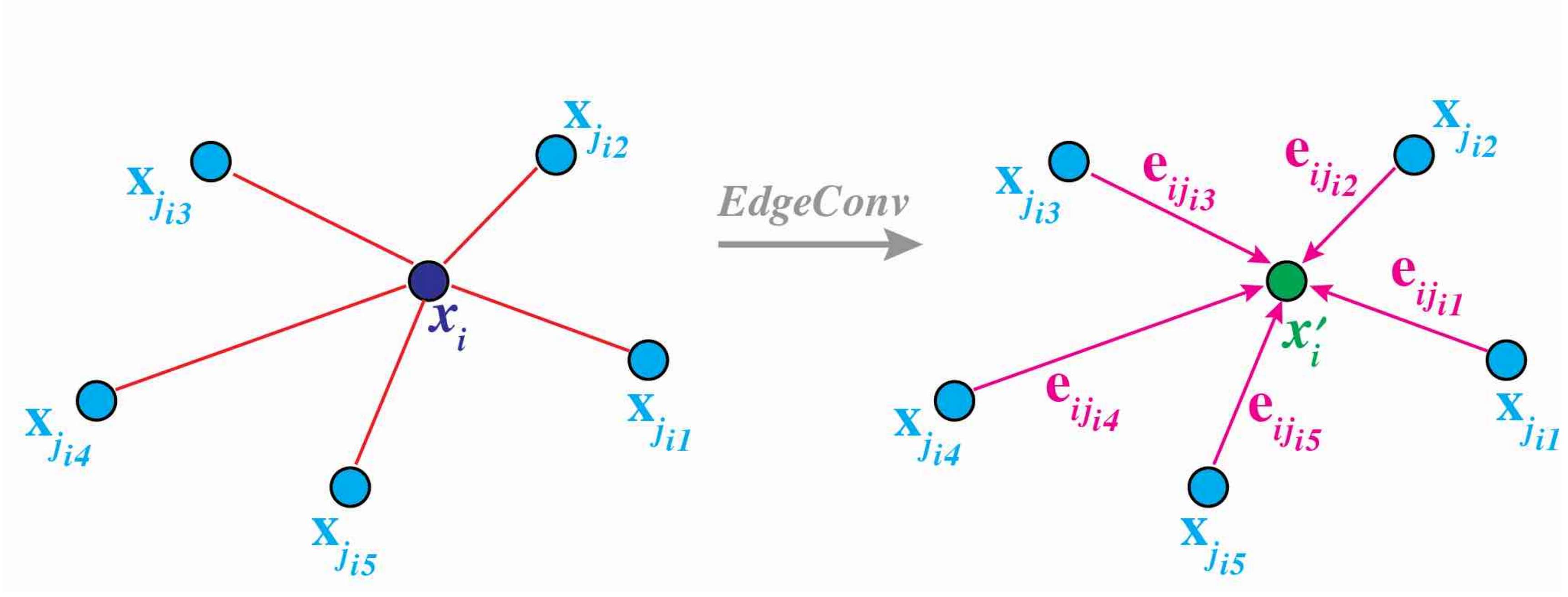
$$e'_{ij} = \phi^e(v_i, v_j) = \text{MLP}(v_i, v_j)$$

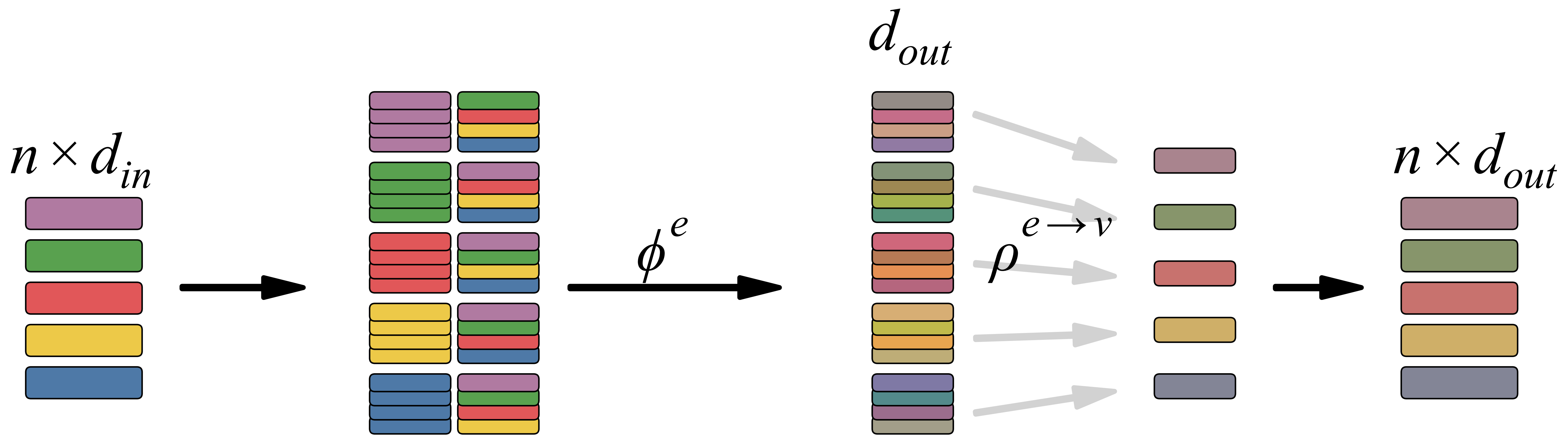
$$v'_i = \rho^{e\to v}(E'_i) = \square_j\, e'_{ij}$$

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

EXAMPLE: DYNAMIC GRAPH CNN (DGCNN)



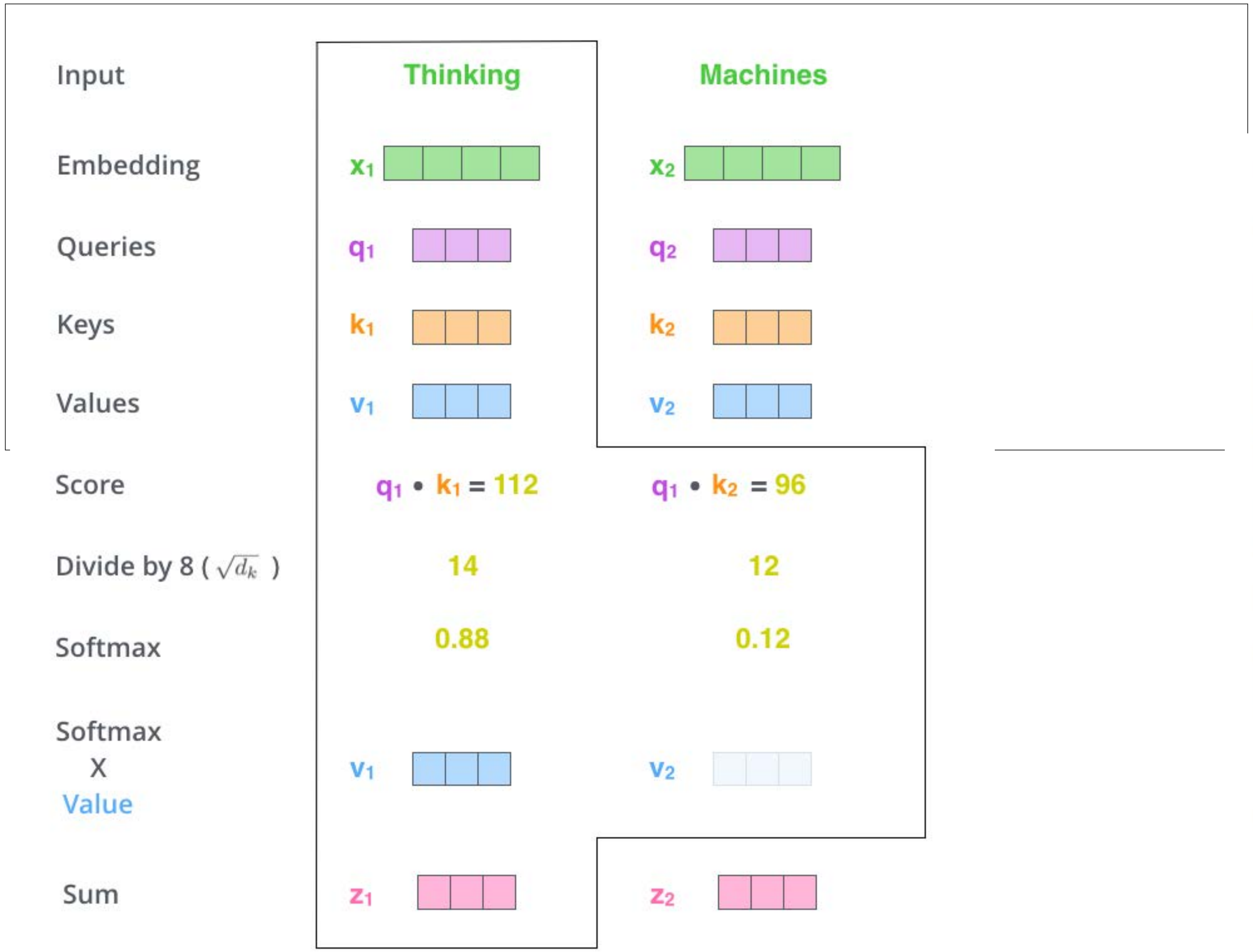


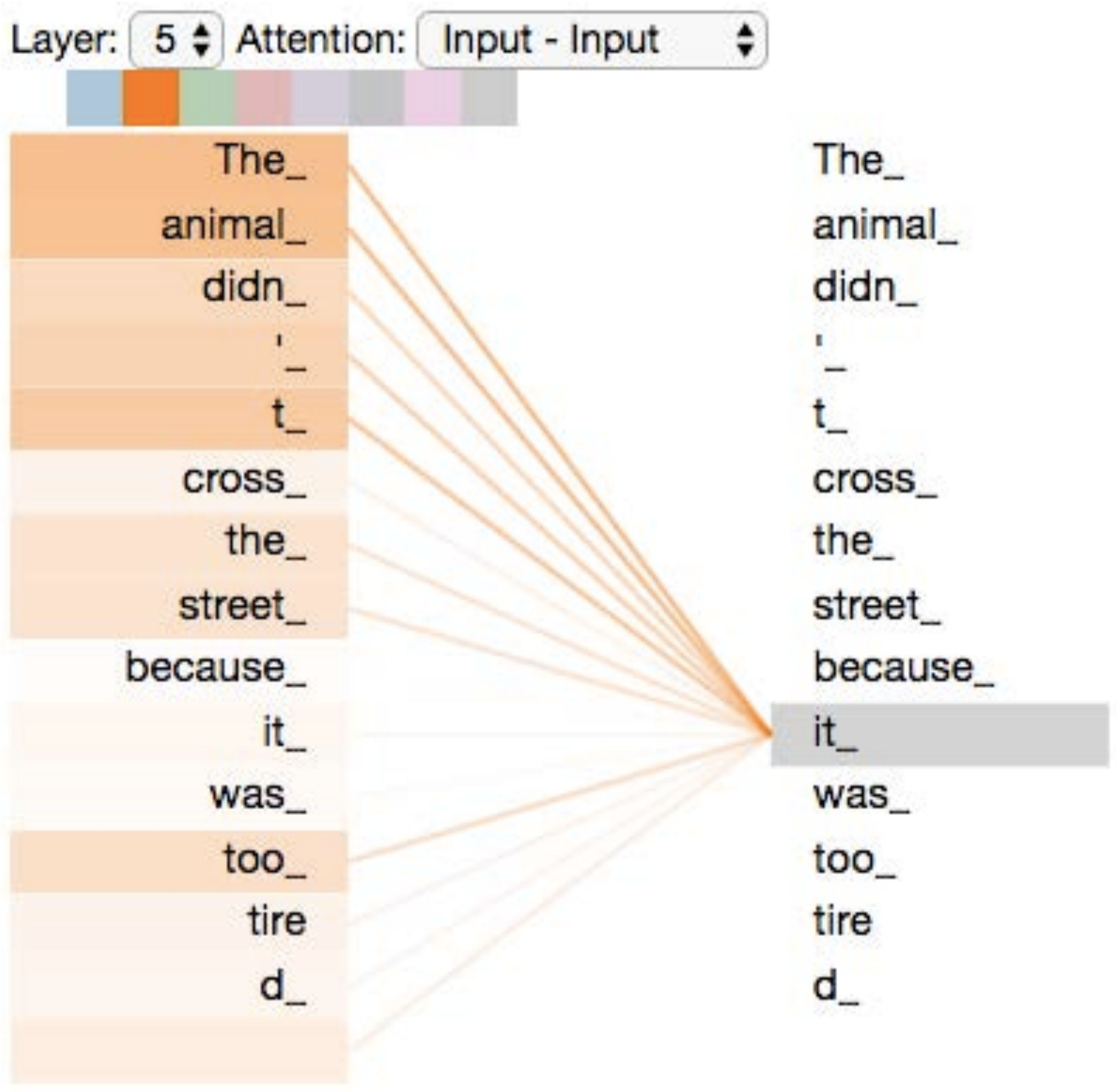


EXAMPLE: TRANSFORMER

- **Transformers**: the new state-of-the-art architecture in ML – foundation of LLM like BERT/GPT
  - core concept: self-attention mechanism

Vaswani, Shazeer, Parmar, Uszkoreit, Jones, Gomez, Kaiser, Polosukhin, arXiv:1706.03762 ["*Attention Is All You Need*"]

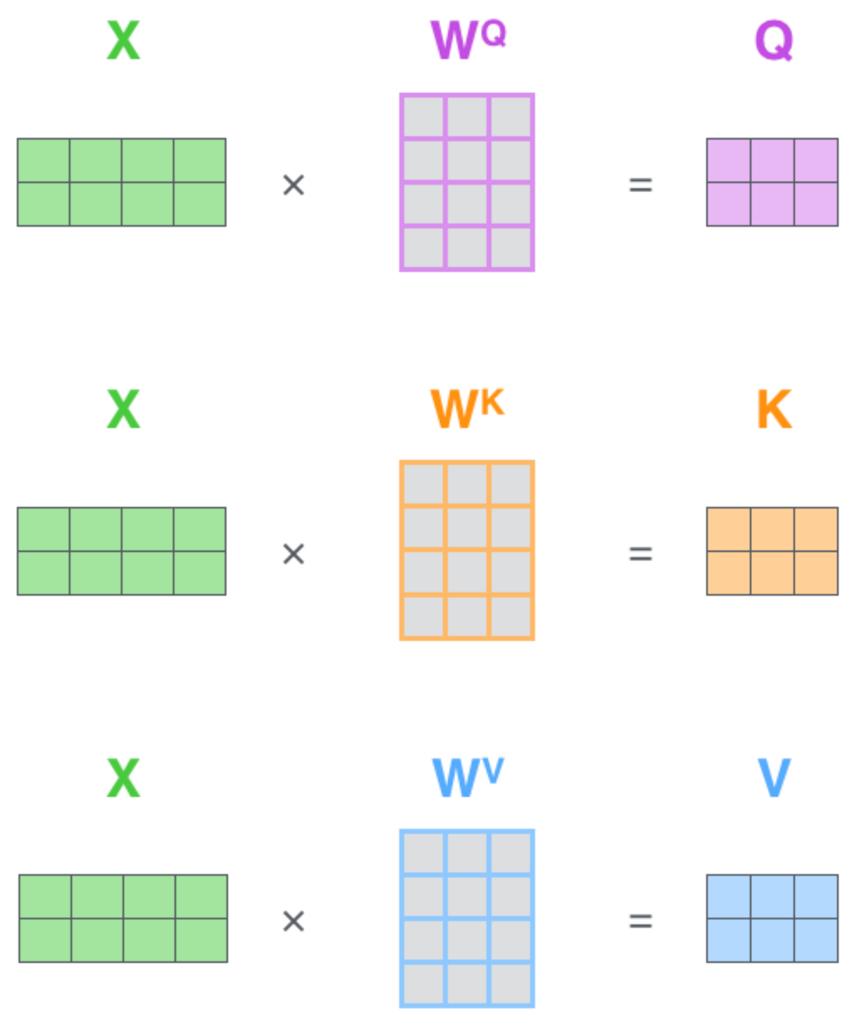


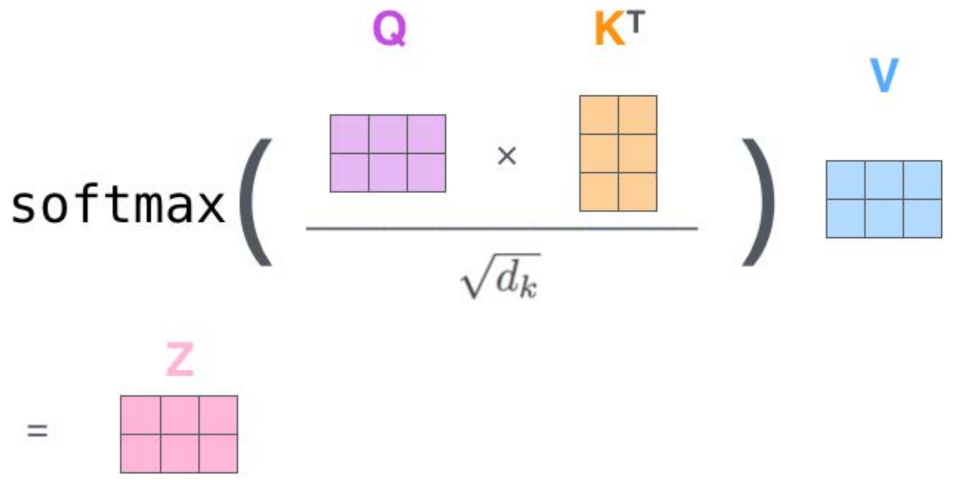


https://jalammar.github.io/illustrated-transformer/

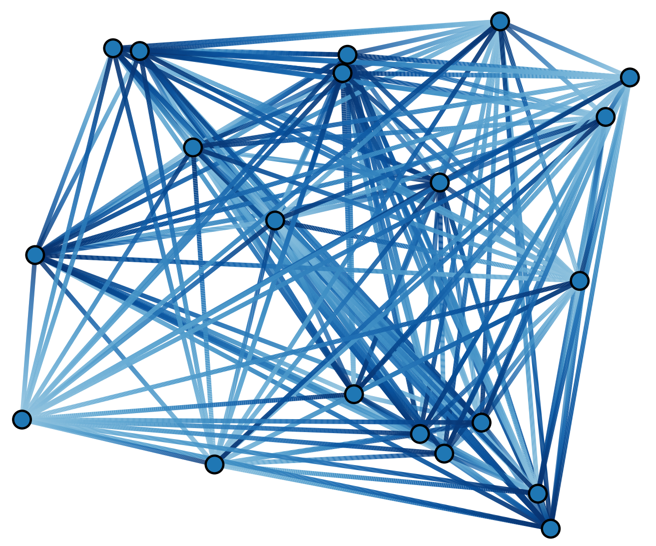
Example: Transformer

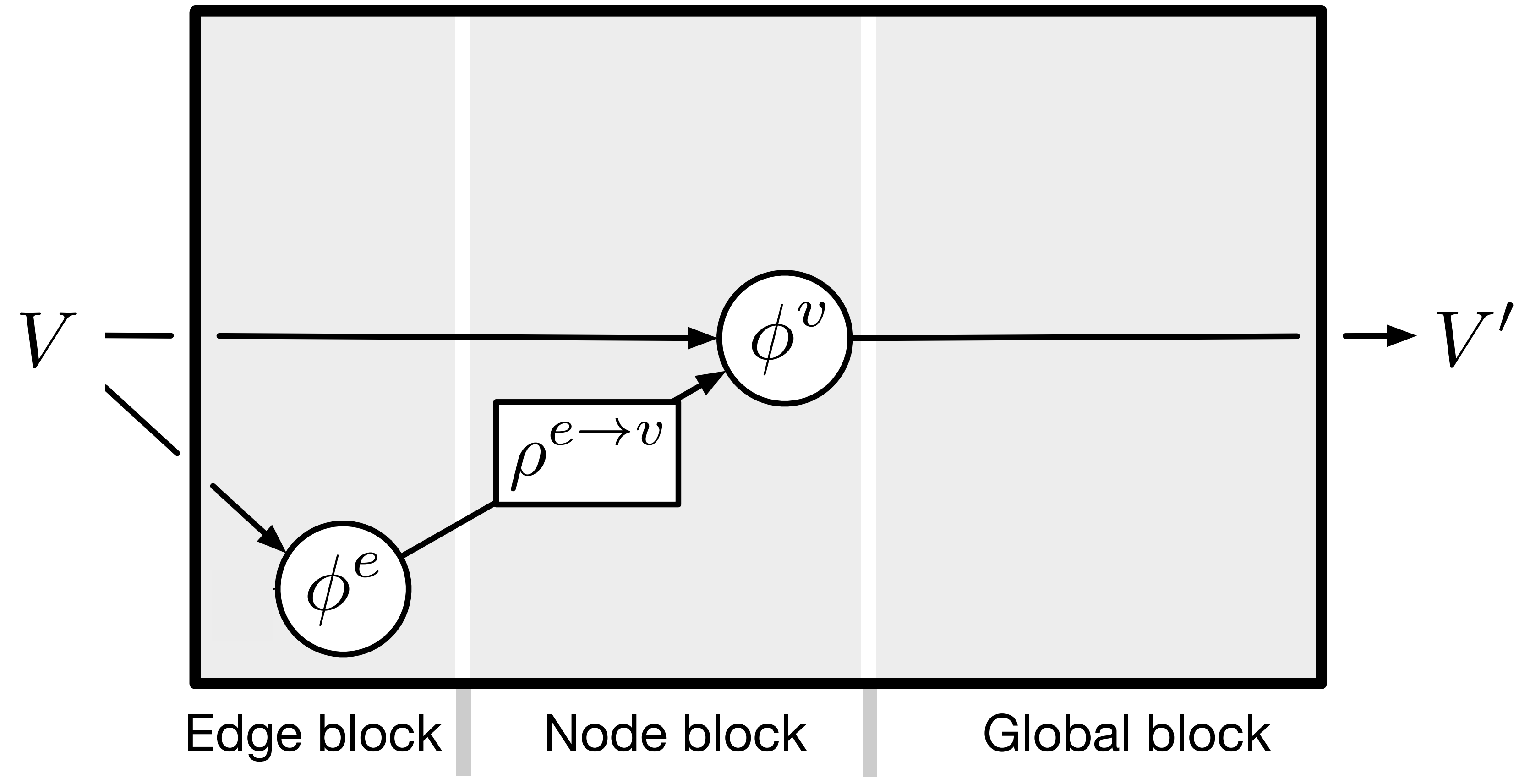
- The transformer architecture is also permutation-invariant as long as positional encoding is not used





Fully connected graph





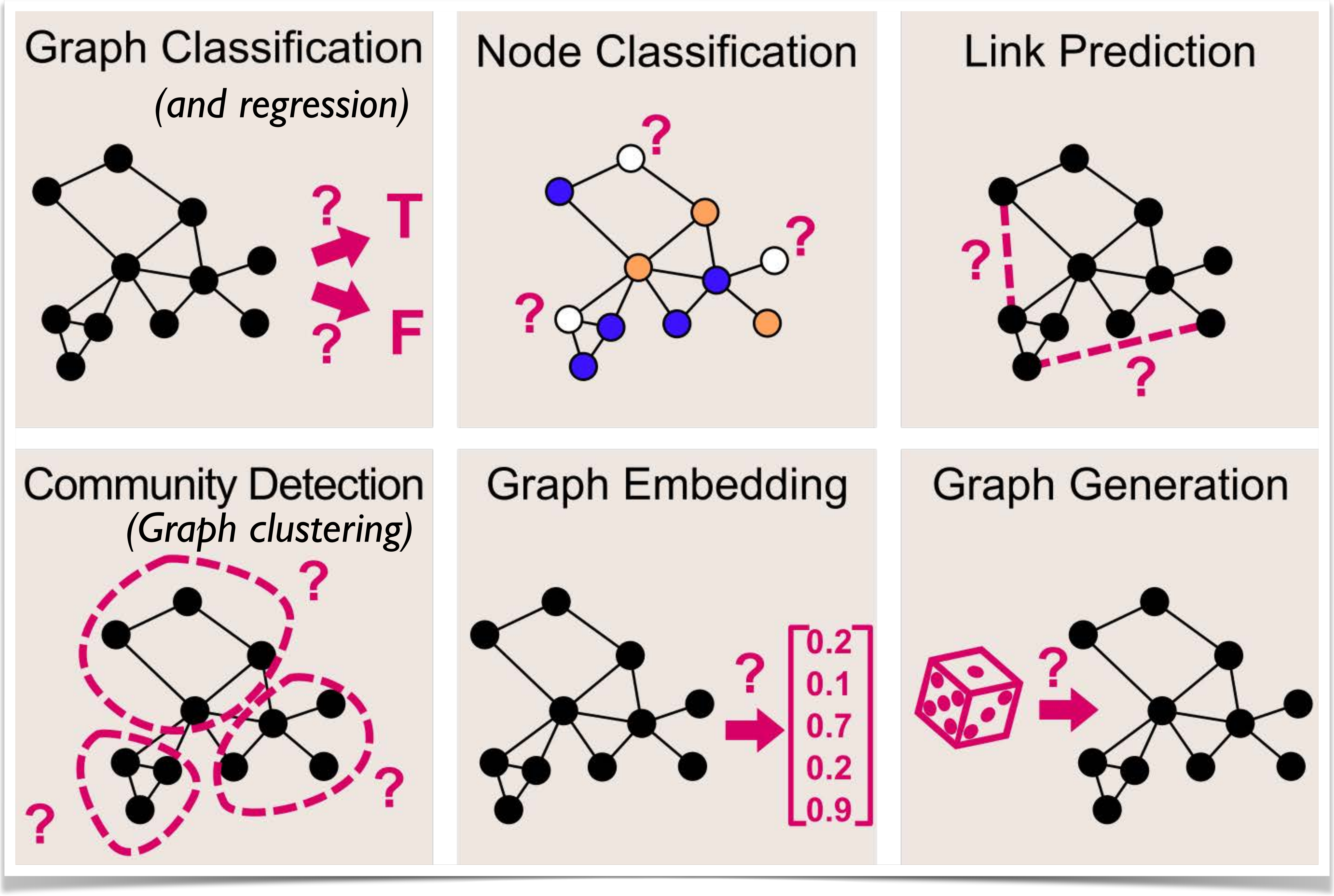
$$e'_{ij} = \phi^e(v_i, v_j) = (W_Q v_i)^T (W_K v_j)$$

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

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

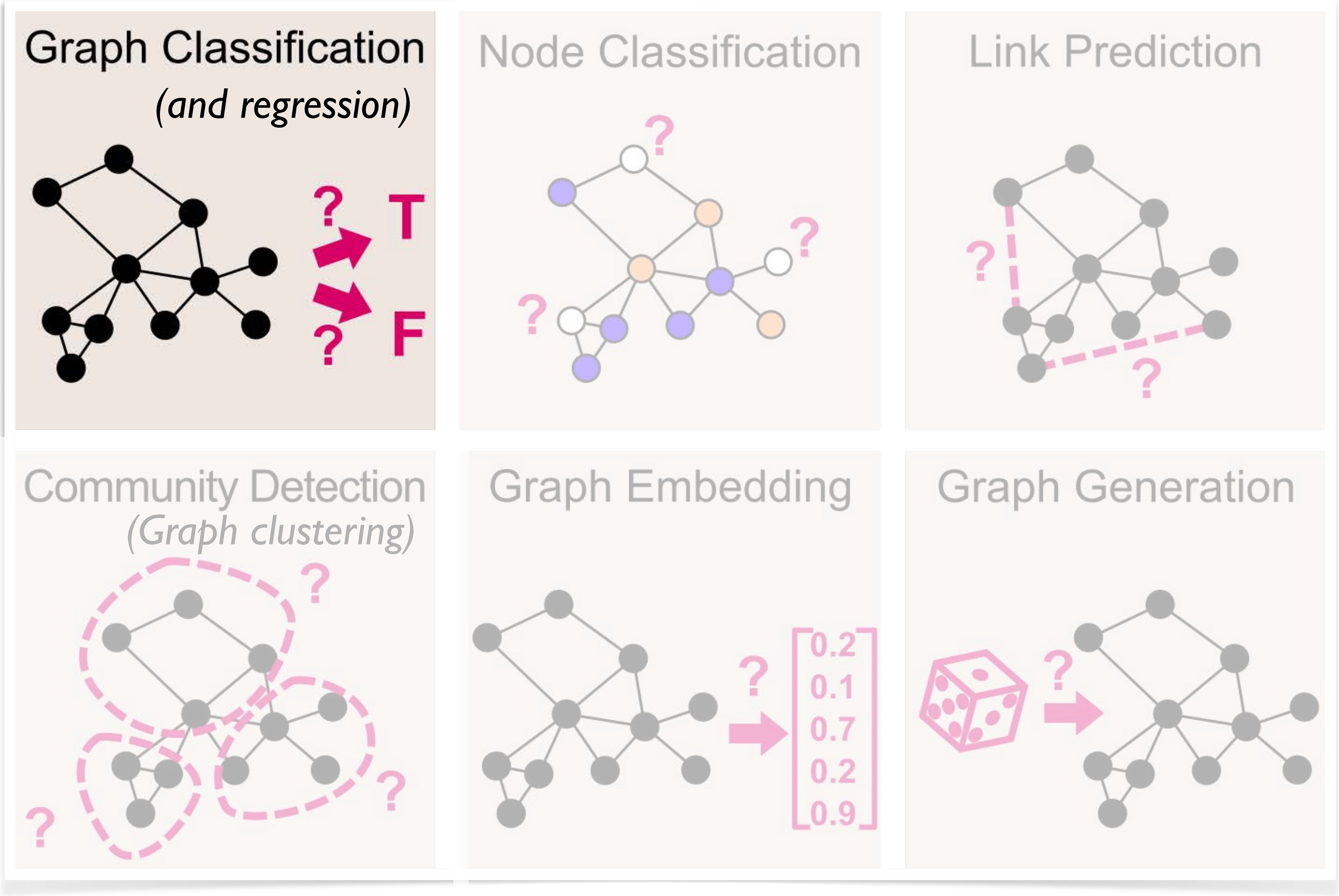
GRAPH NEURAL NETWORKS IN ACTION

Graph ML Tasks



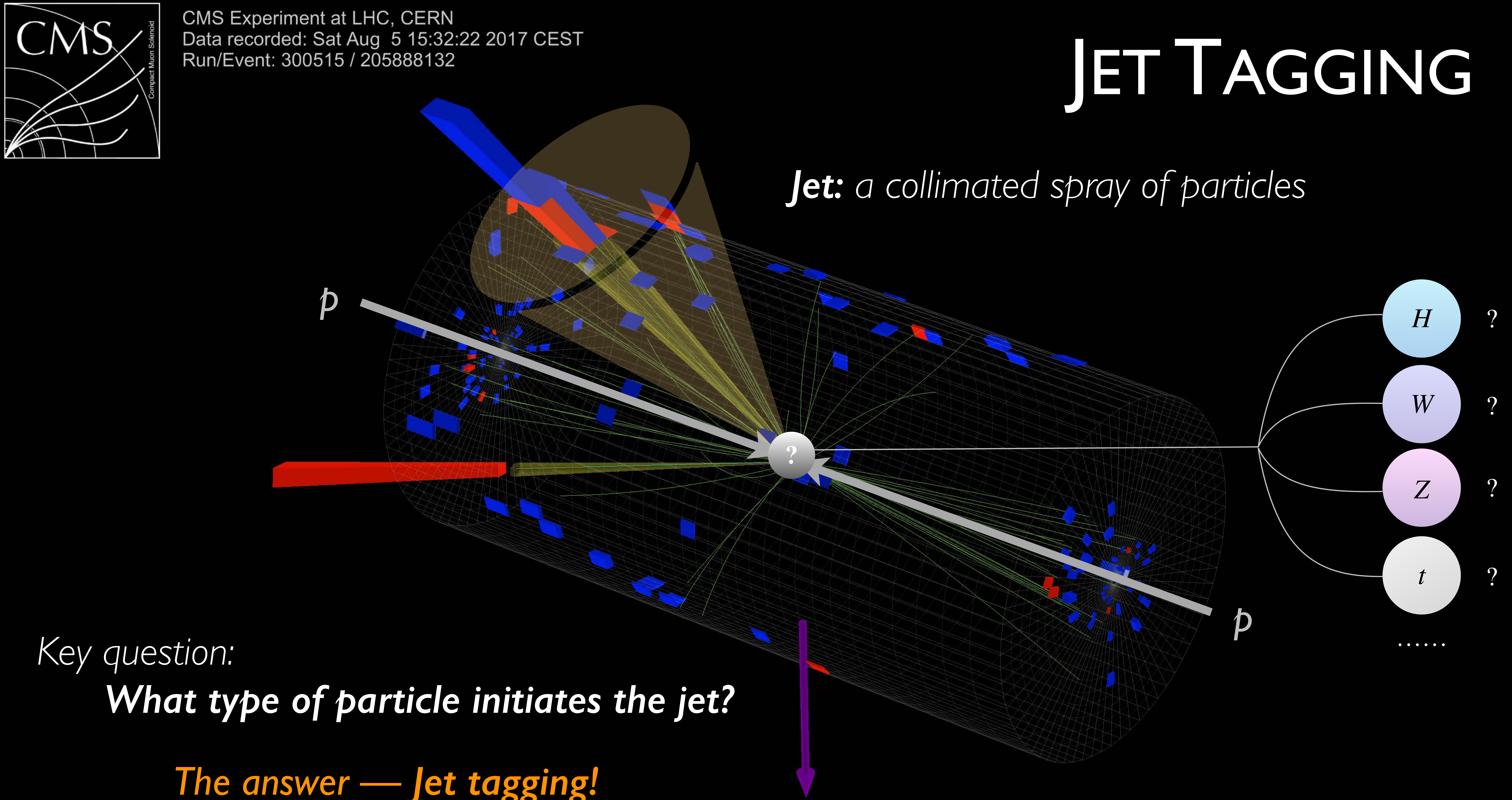
https://towardsdatascience.com/graph-convolutional-networks-deep-99d7fee5706f

GRAPH ML TASKS



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

JET TAGGING



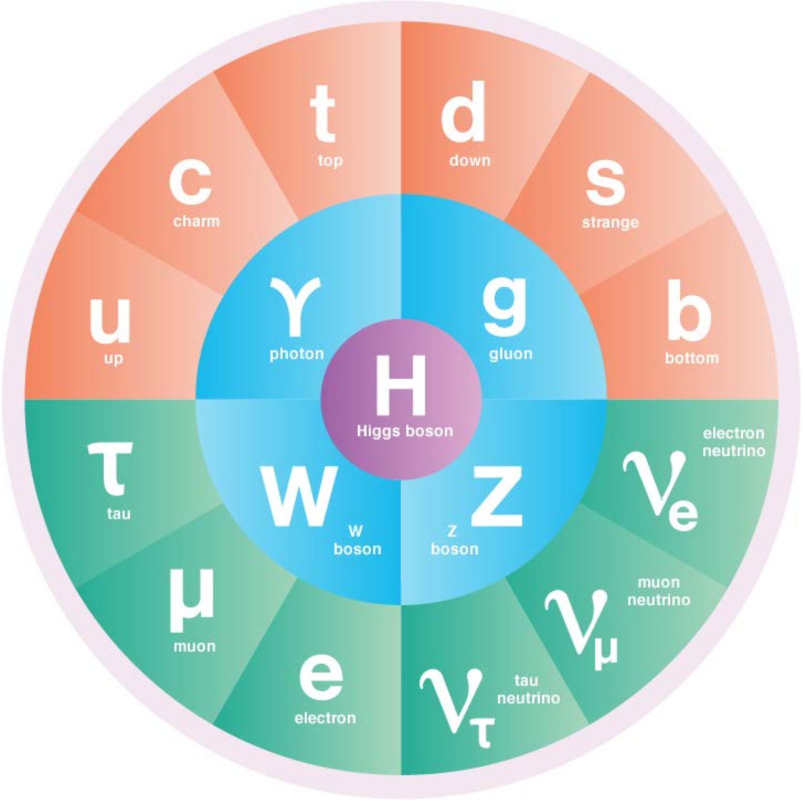
Jet: *a collimated spray of particles*

Key question:

What type of particle initiates the jet?

The answer — Jet tagging!

Jet Tagging



- Jet tagging: identifying the origin of a jet, i.e., what kind of particle initiates the jet
  - essentially a classification task from the machine learning perspective

Jet flavor tagging

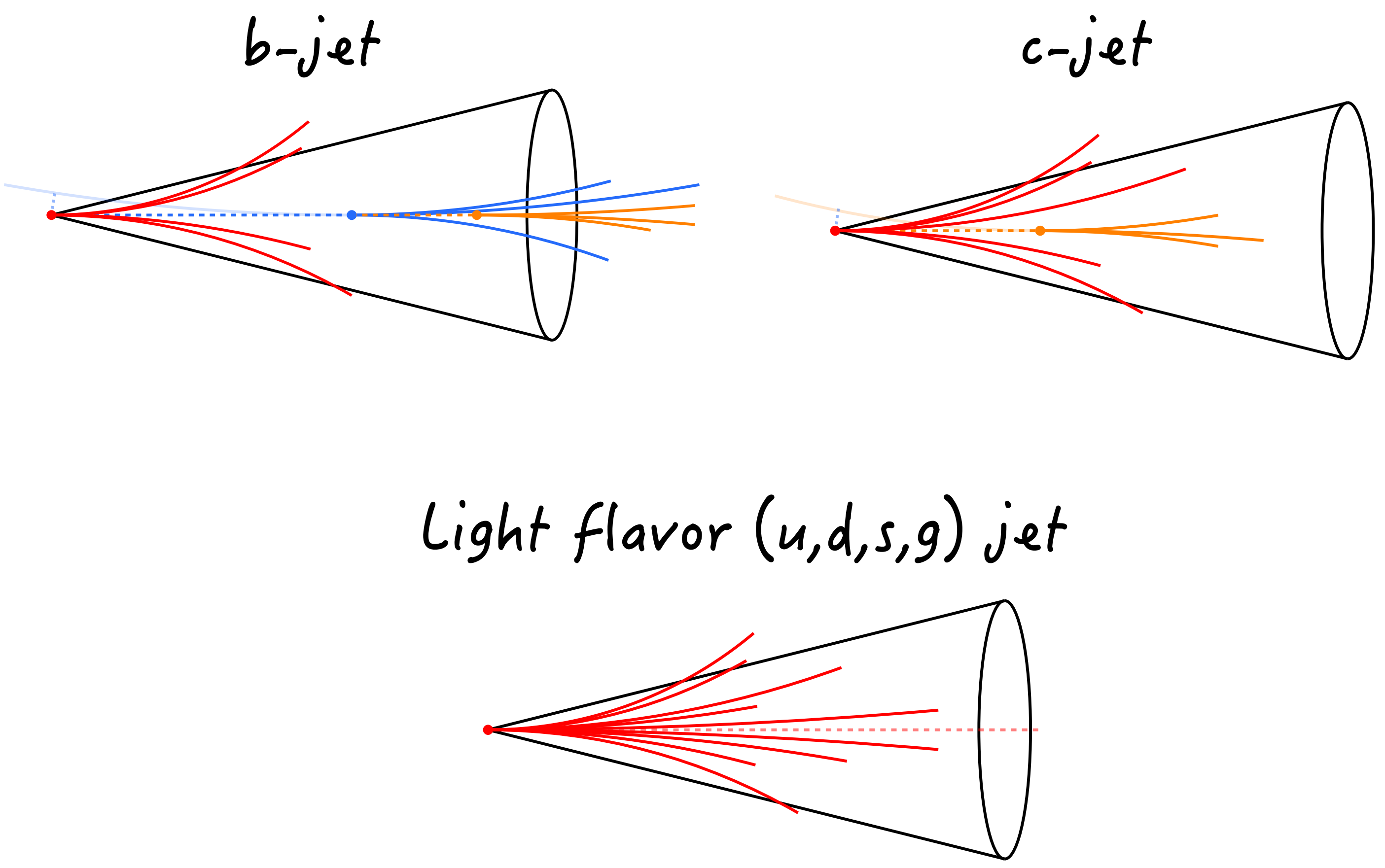


Image credit

Boosted jet tagging

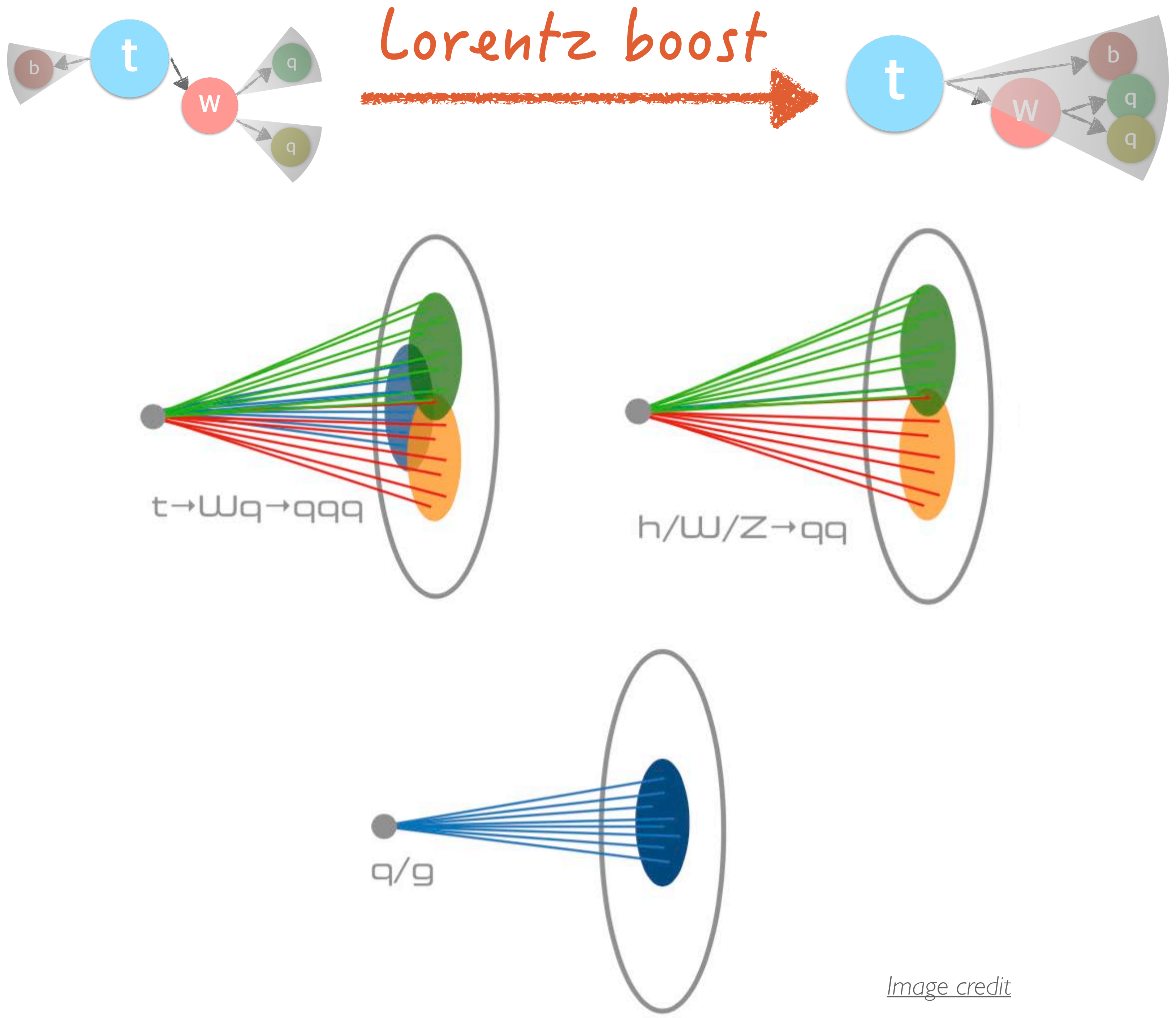


Image credit

Hadronic τ tagging

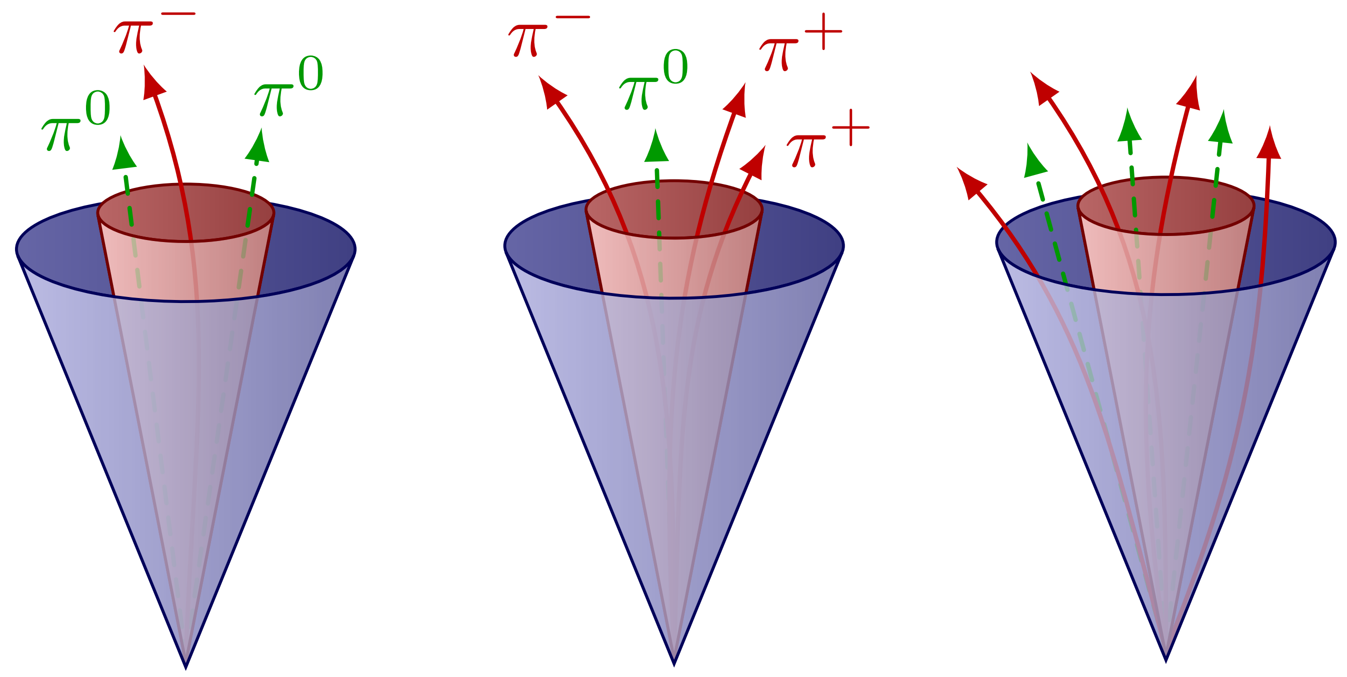
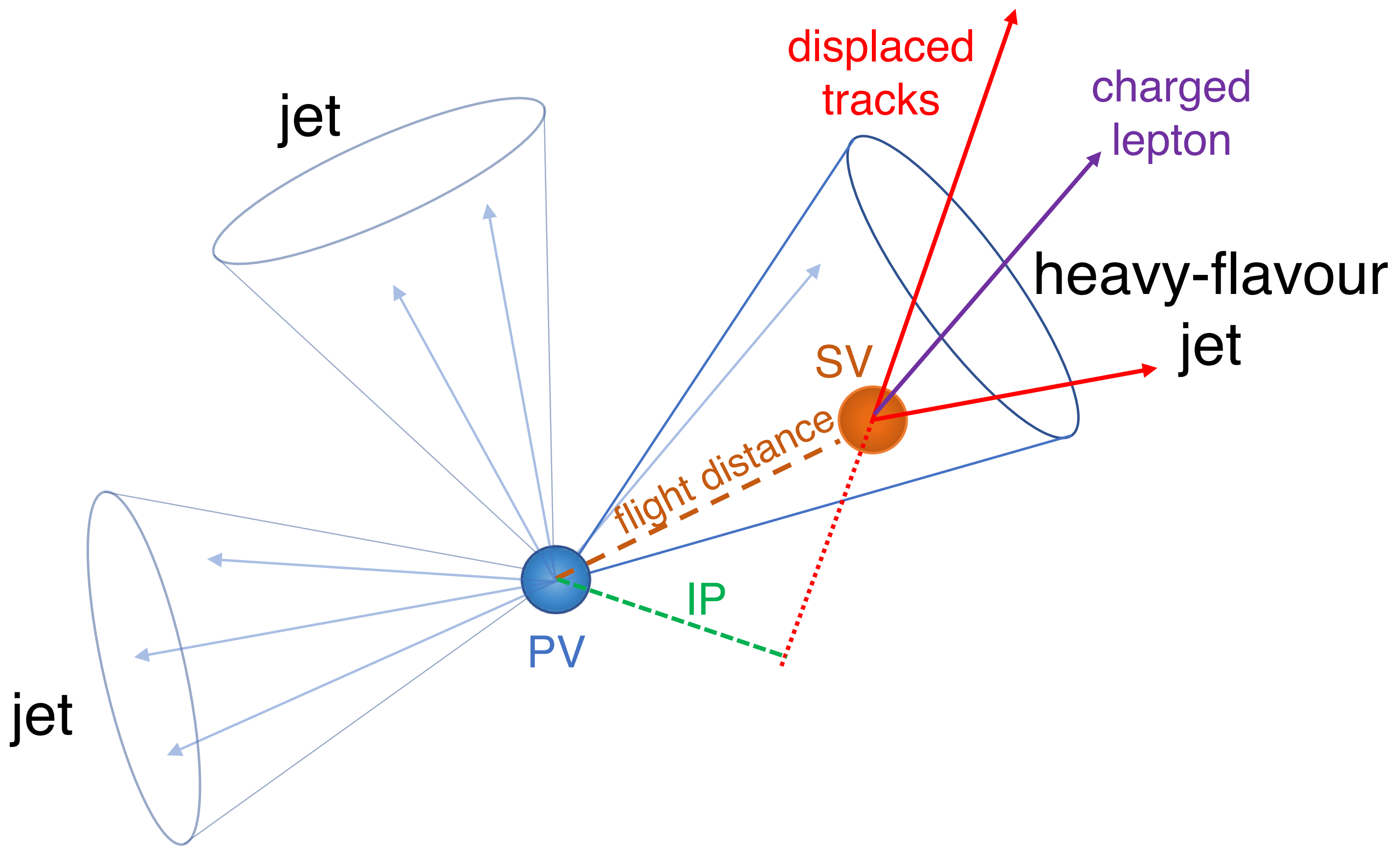


Image credit

Boosted Jet Tagging

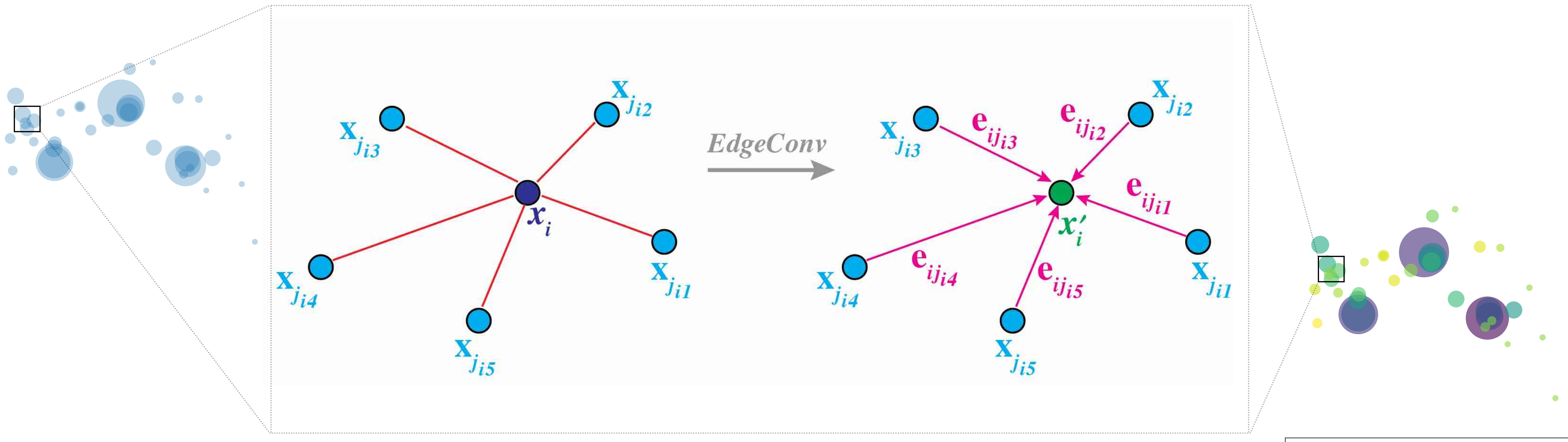
- Hadronic decays of highly Lorentz-boosted heavy particles (Higgs/W/Z/top) lead to large-radius jets with distinctive characteristics:
  - different radiation patterns ("**substructure**")
    - 3-prong (top), 2-prong (W/Z/H) vs 1-prong (gluon/light quark jet)
  - different **flavor** content: existence of one or more b-/c-quarks
- Boosted jet tagging:
  - simultaneously exploiting both **substructure** and **flavor** to maximize the performance
  - significant performance leap thanks to deep learning techniques



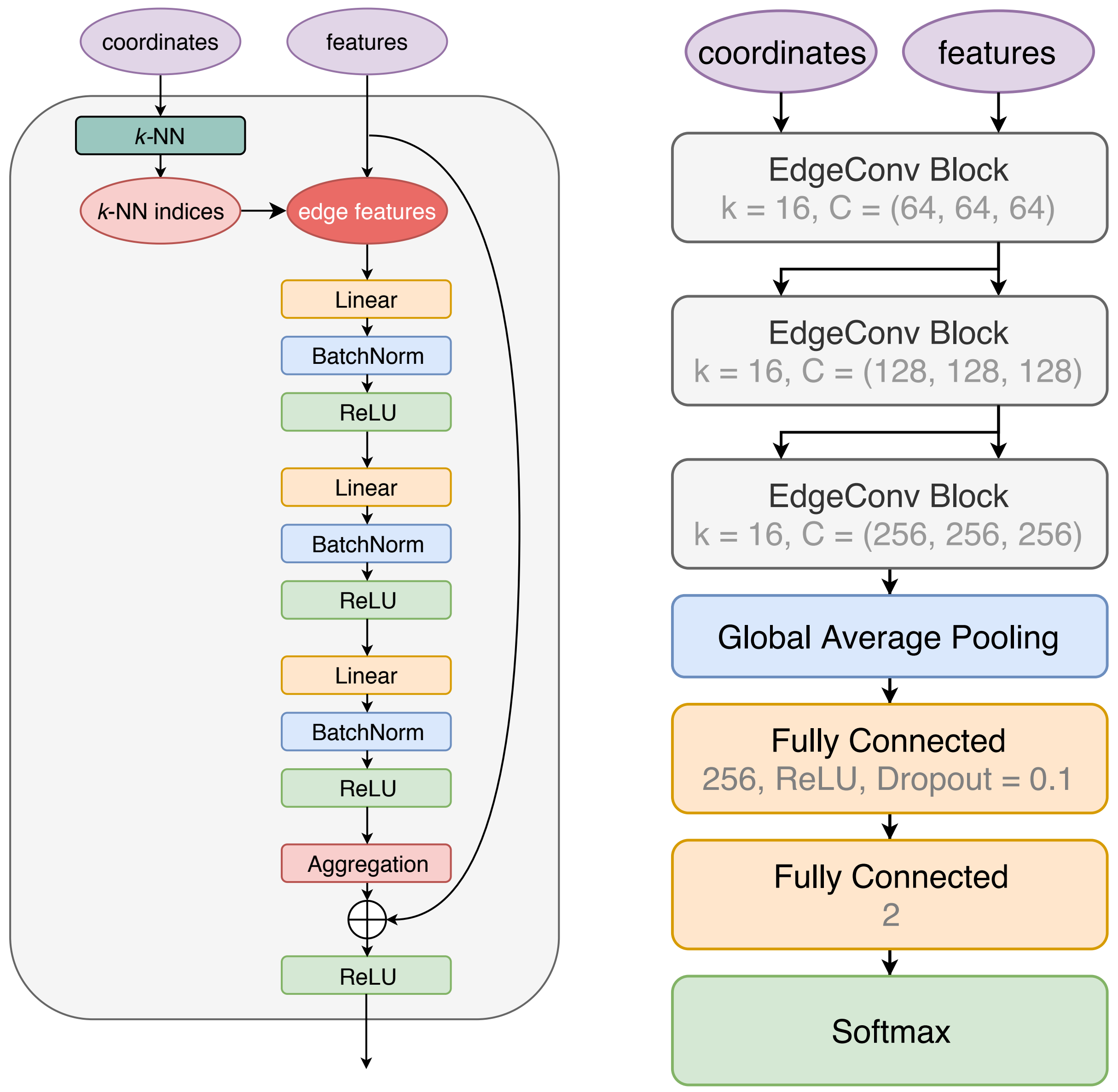
ParticleNet

HQ and L. Gouskos
Phys.Rev.D 101 (2020) 5, 056019

- 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: $e_{ij} = \mathrm{MLP}(x_i, x_j)$
      - aggregate the edge features in a symmetric way: $x_i' = \mathrm{mean}_j\, e_{ij}$



ParticleNet architecture



cf. P. T. Komiske, E. M. Metodiev and J. Thaler, JHEP 01 (2019) 121;
V. Mikuni and F. Canelli, Eur. Phys. J. Plus 135, 463 (2020); Mach.Learn.Sci.Tech. 2 (2021) 3, 035027.

ParticleNet: Performance

G. Kasieczka et al.
SciPost Phys. 7 (2019) 014

- Top performance among a variety of deep learning taggers on the community-wide top tagging benchmark

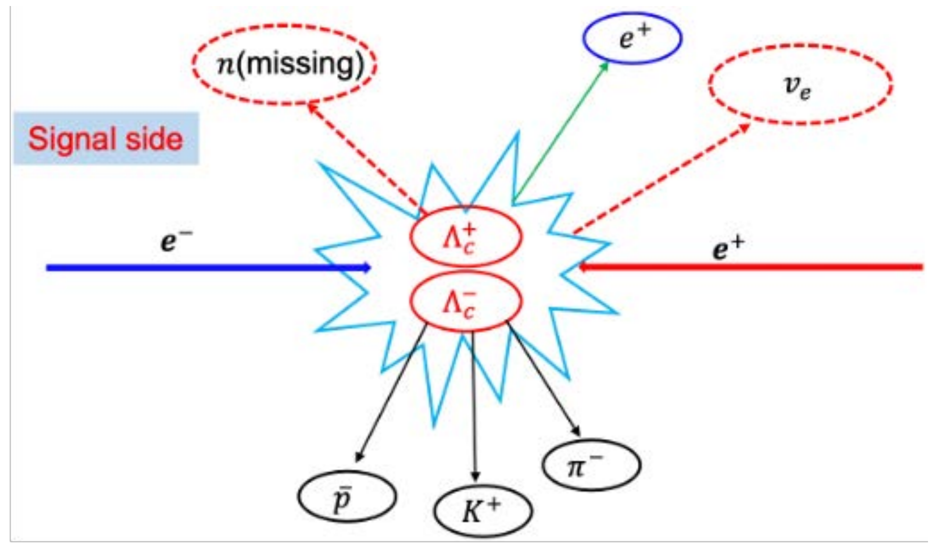
	AUC	Acc	$1/\epsilon_B$ ($\epsilon_S = 0.3$) single	mean	median	#Param
CNN [16]	0.981	0.930	914±14	995±15	975±18	610k
ResNeXt [30]	0.984	0.936	1122±47	1270±28	1286±31	1.46M
TopoDNN [18]	0.972	0.916	295±5	382± 5	378 ± 8	59k
Multi-body N-subjettiness 6 [24]	0.979	0.922	792±18	798±12	808±13	57k
Multi-body N-subjettiness 8 [24]	0.981	0.929	867±15	918±20	926±18	58k
TreeNiN [43]	0.982	0.933	1025±11	1202±23	1188±24	34k
P-CNN	0.980	0.930	732±24	845±13	834±14	348k
ParticleNet [47] (Preliminary ver.)	0.985	0.938	1298±46	1412±45	1393±41	498k
LBN [19]	0.981	0.931	836±17	859±67	966±20	705k
LoLa [22]	0.980	0.929	722±17	768±11	765±11	127k
Energy Flow Polynomials [21]	0.980	0.932	384			1k
Energy Flow Network [23]	0.979	0.927	633±31	729±13	726±11	82k
Particle Flow Network [23]	0.982	0.932	891±18	1063±21	1052±29	82k
GoaT	0.985	0.939	1368±140		1549±208	35k
ParticleNet-Lite	0.984	0.937	1262±49			26k
ParticleNet	**0.986**	**0.940**	**1615±93**			366k

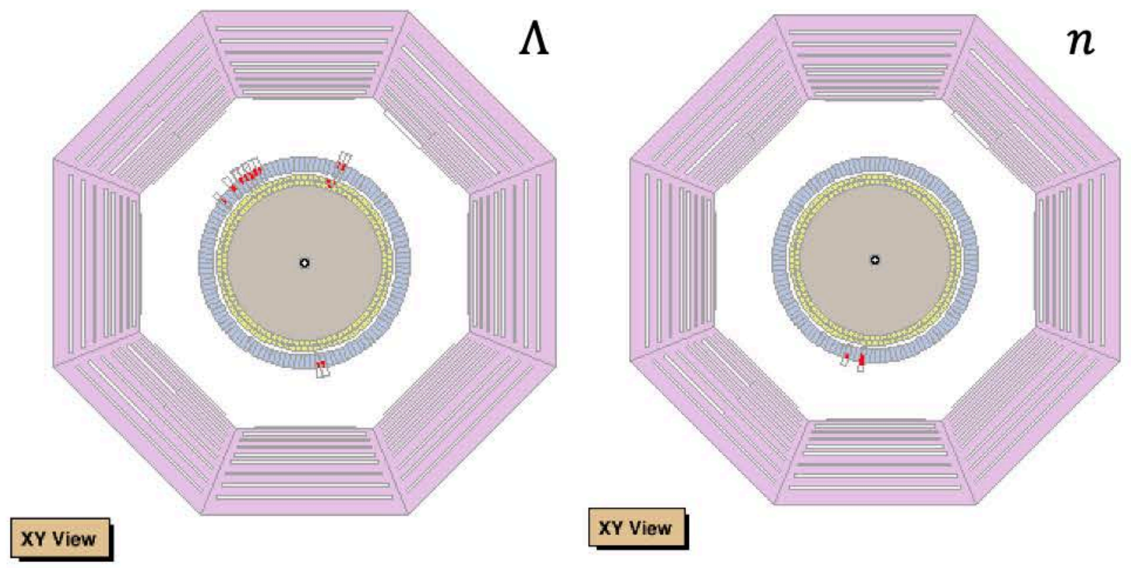
Ensemble of all taggers

ParticleNet: Beyond Jet Tagging

$\Lambda_c^+ \to n e^+ \nu$ search

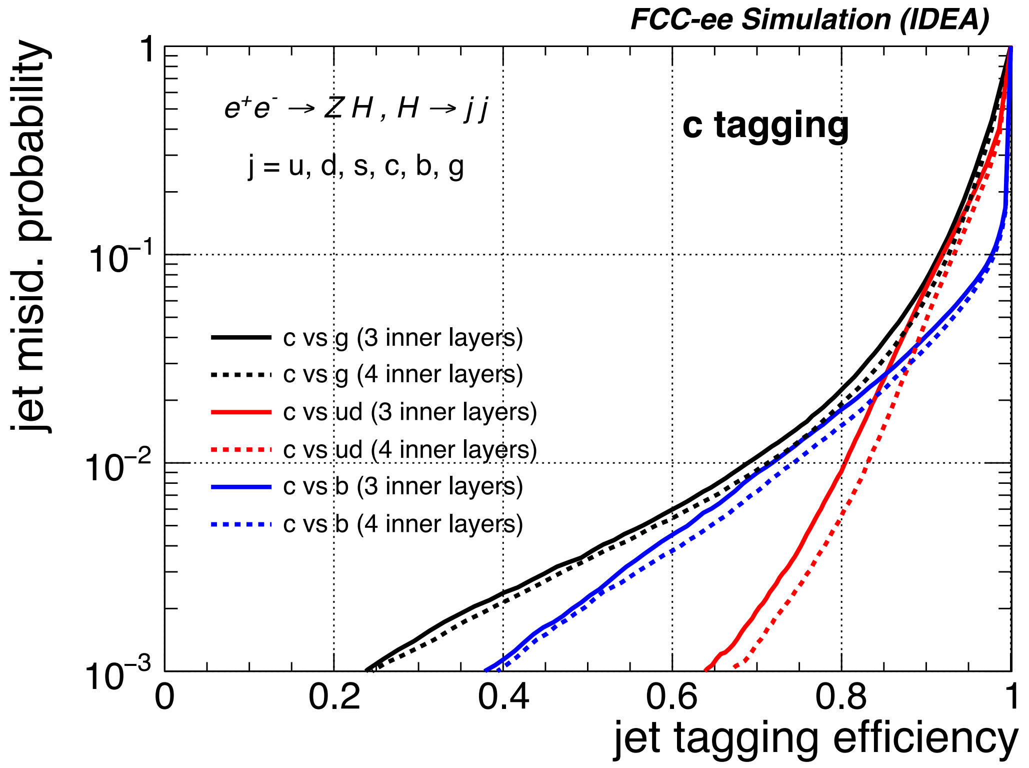
Yunxuan Song, Yangu Li et al.

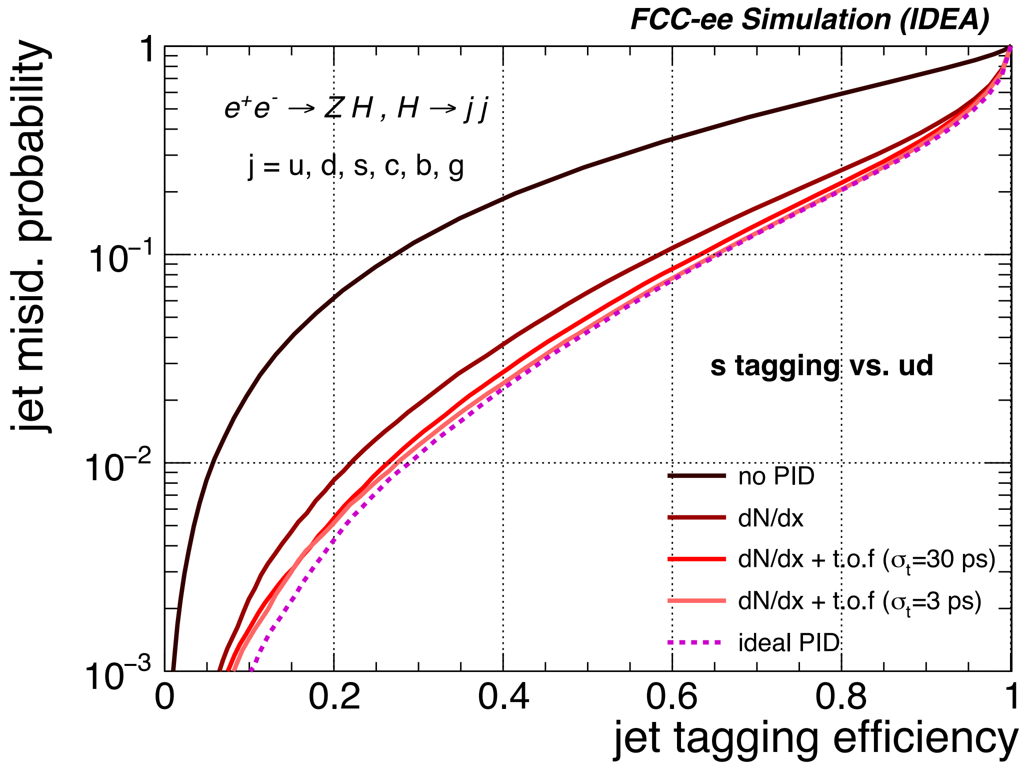




Particle identification

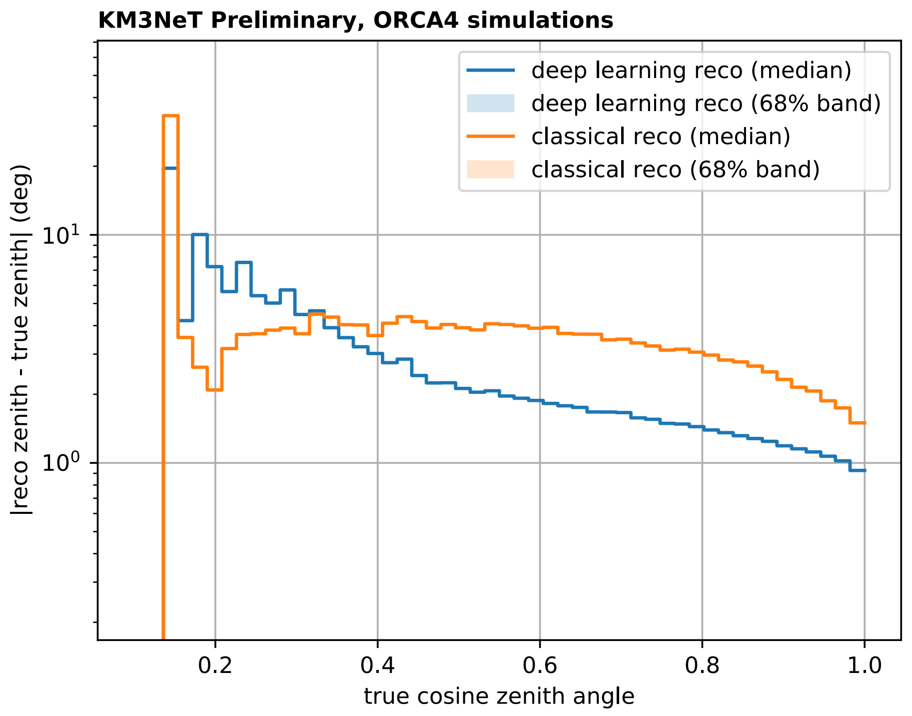
Eur.Phys.J.Plus 137 (2022) 1, 39
Eur.Phys.J.C 82 (2022) 7, 646



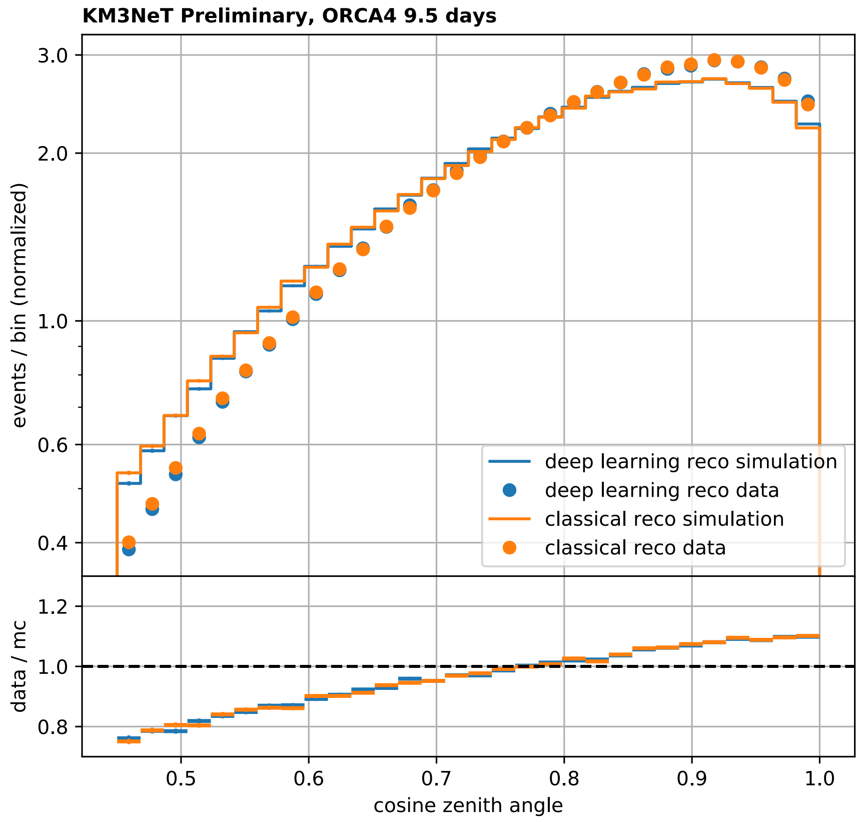


Muon bundle reconstruction

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

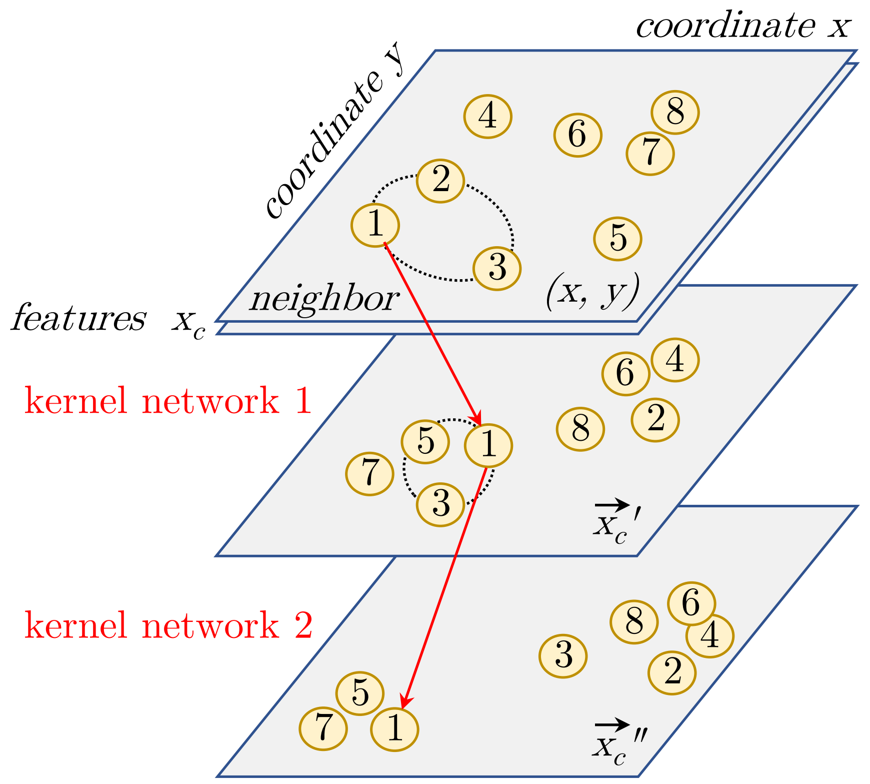


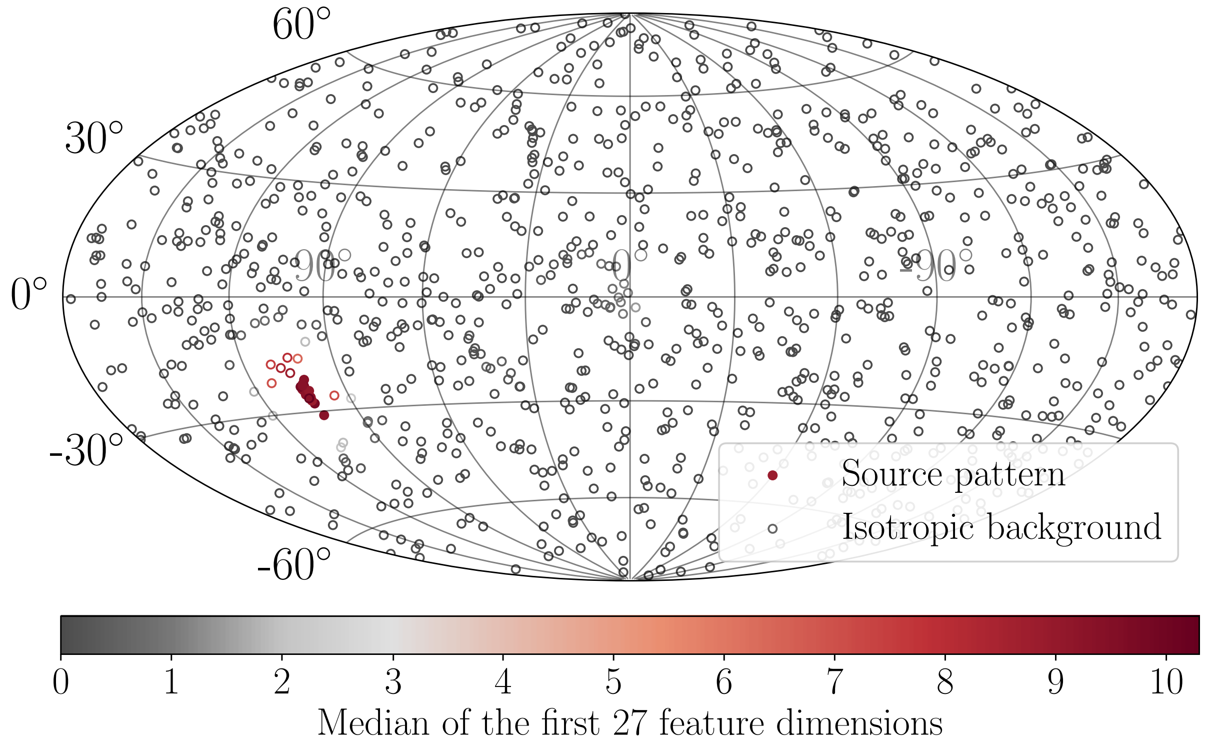
(b) events with two or more muons



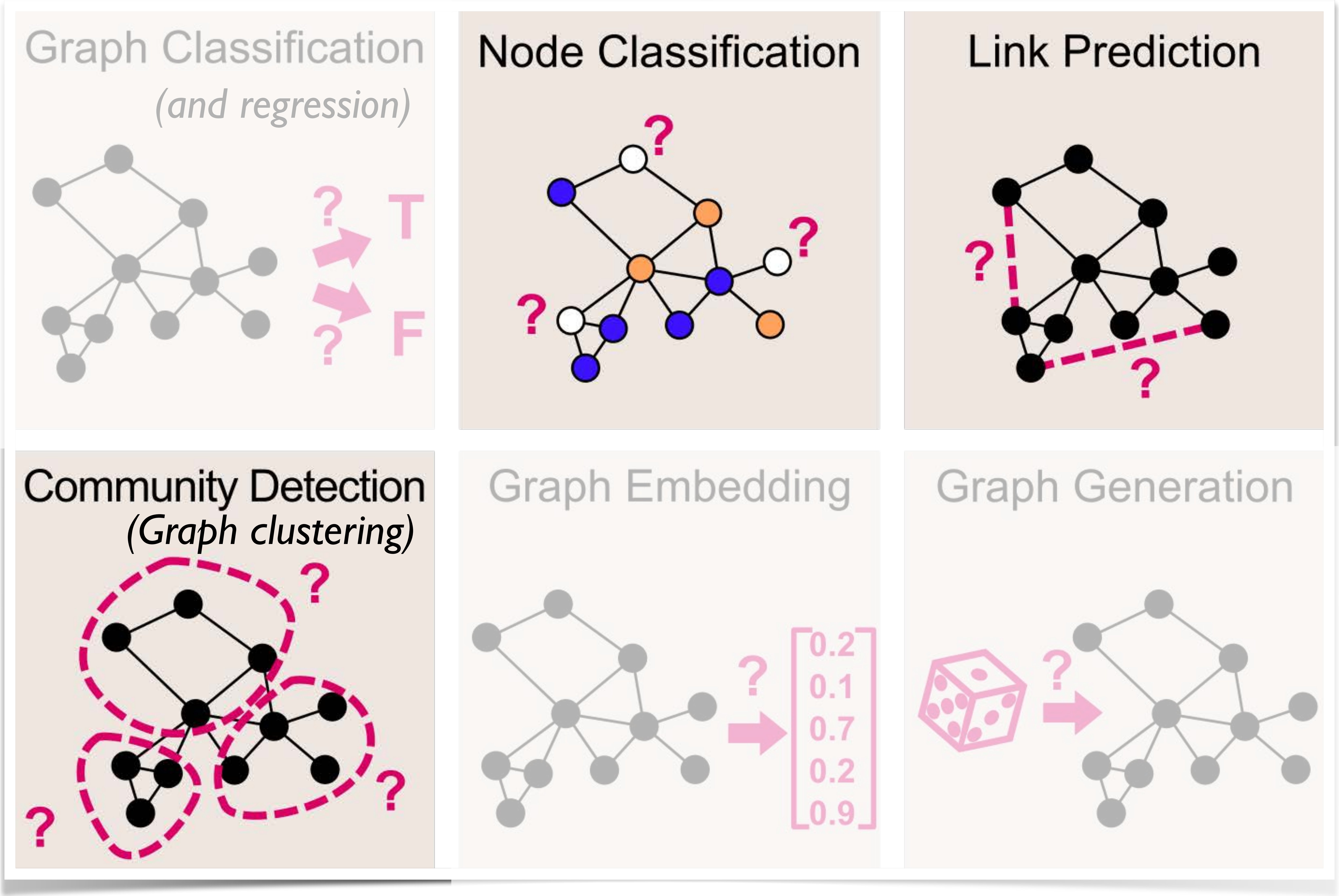
Cosmic ray pattern identification

Astropart.Phys. 126 (2021) 102527





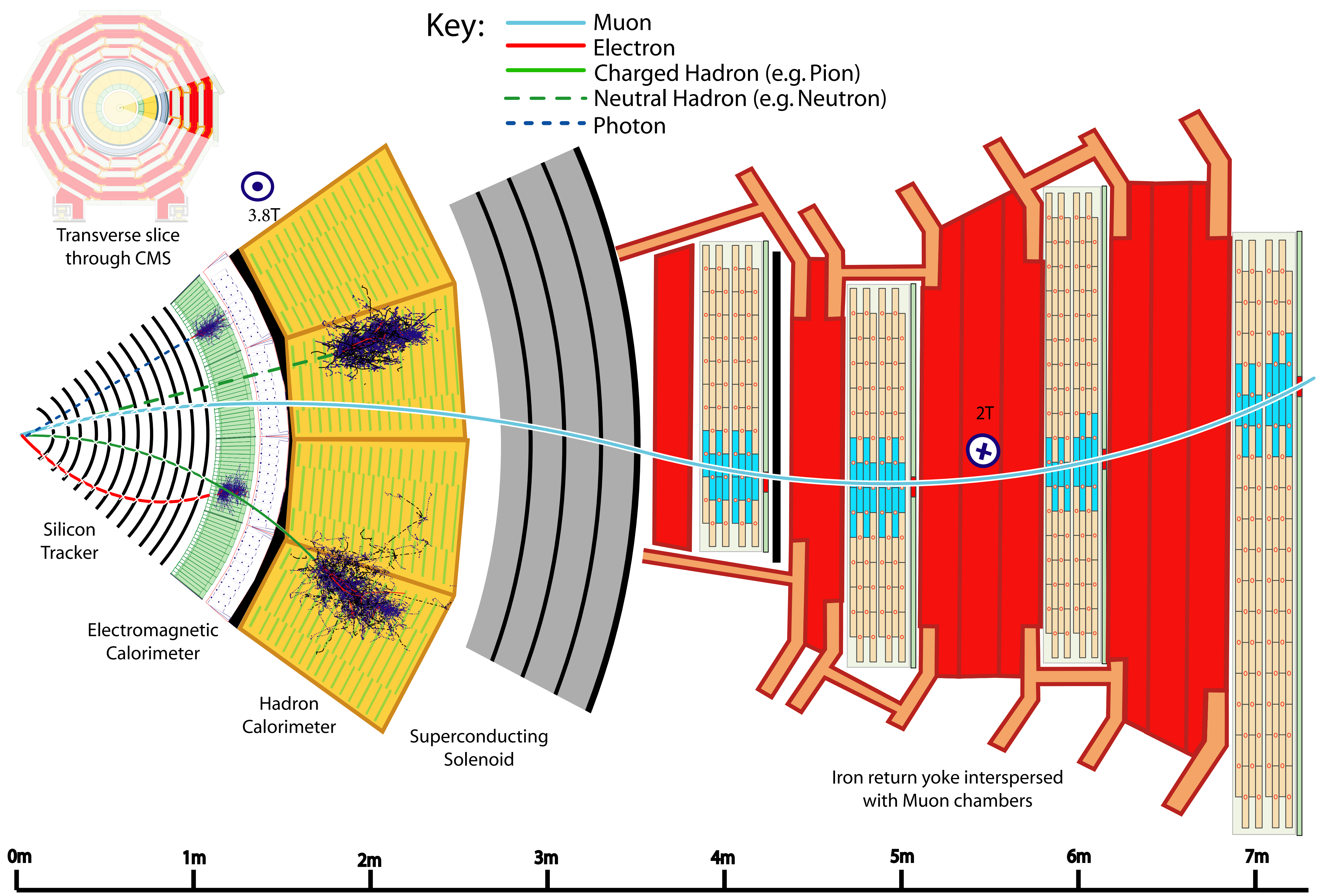
GNNs for Event Reconstruction

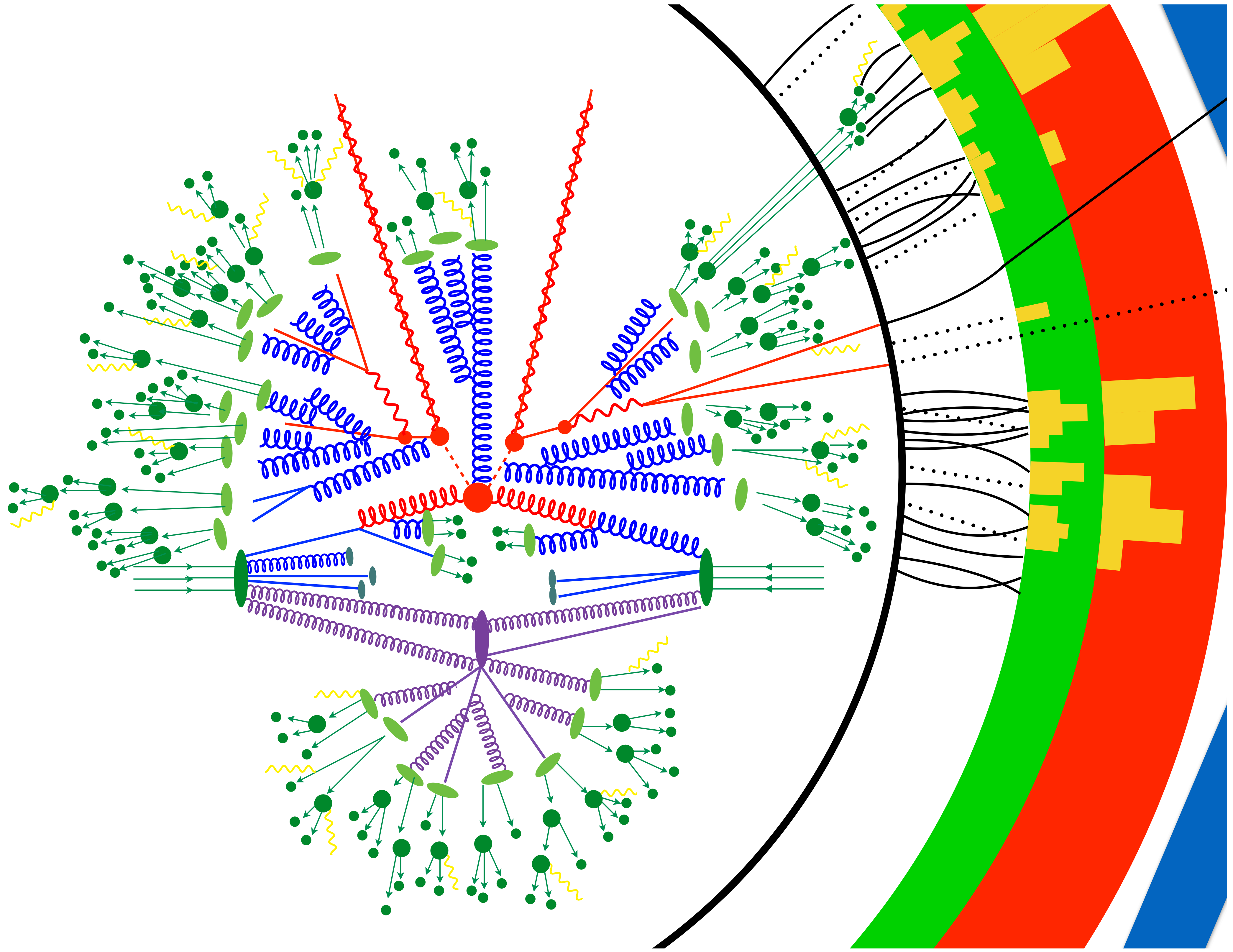


https://towardsdatascience.com/graph-convolutional-networks-deep-99d7fee5706f

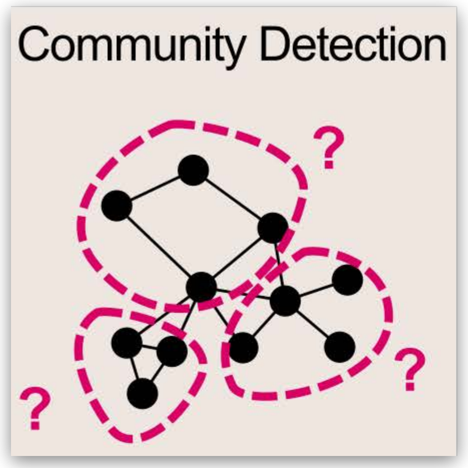
Event Reconstruction

- Event reconstruction: deciphering the detector signals
  - what are the outgoing particles?
  - what are their momenta, energy, ...?





CALORIMETER RECONSTRUCTION

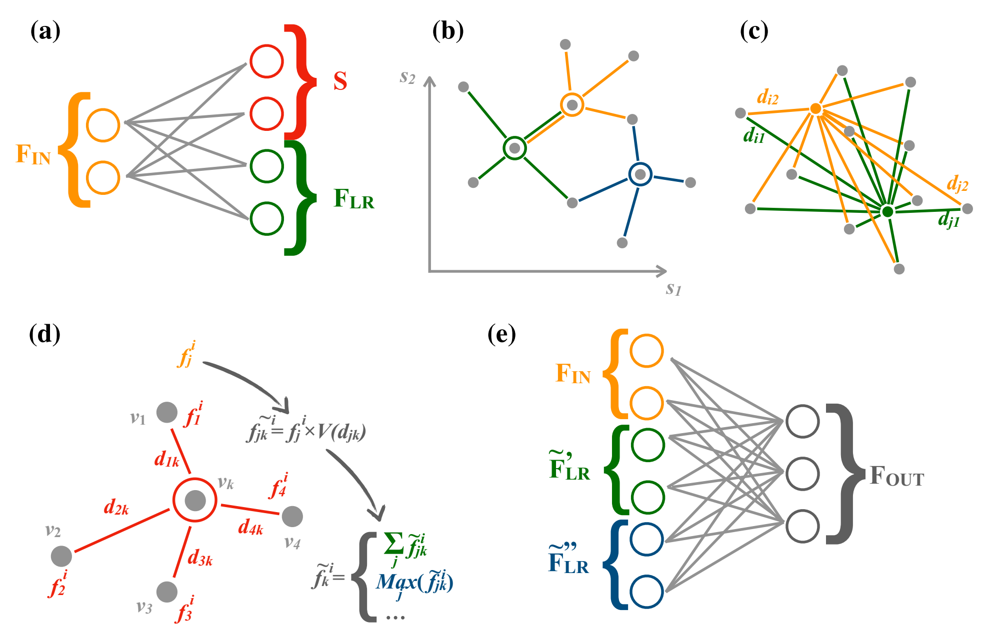


- GNNs also powerful tools for event reconstruction, particularly for non-uniform detector geometry

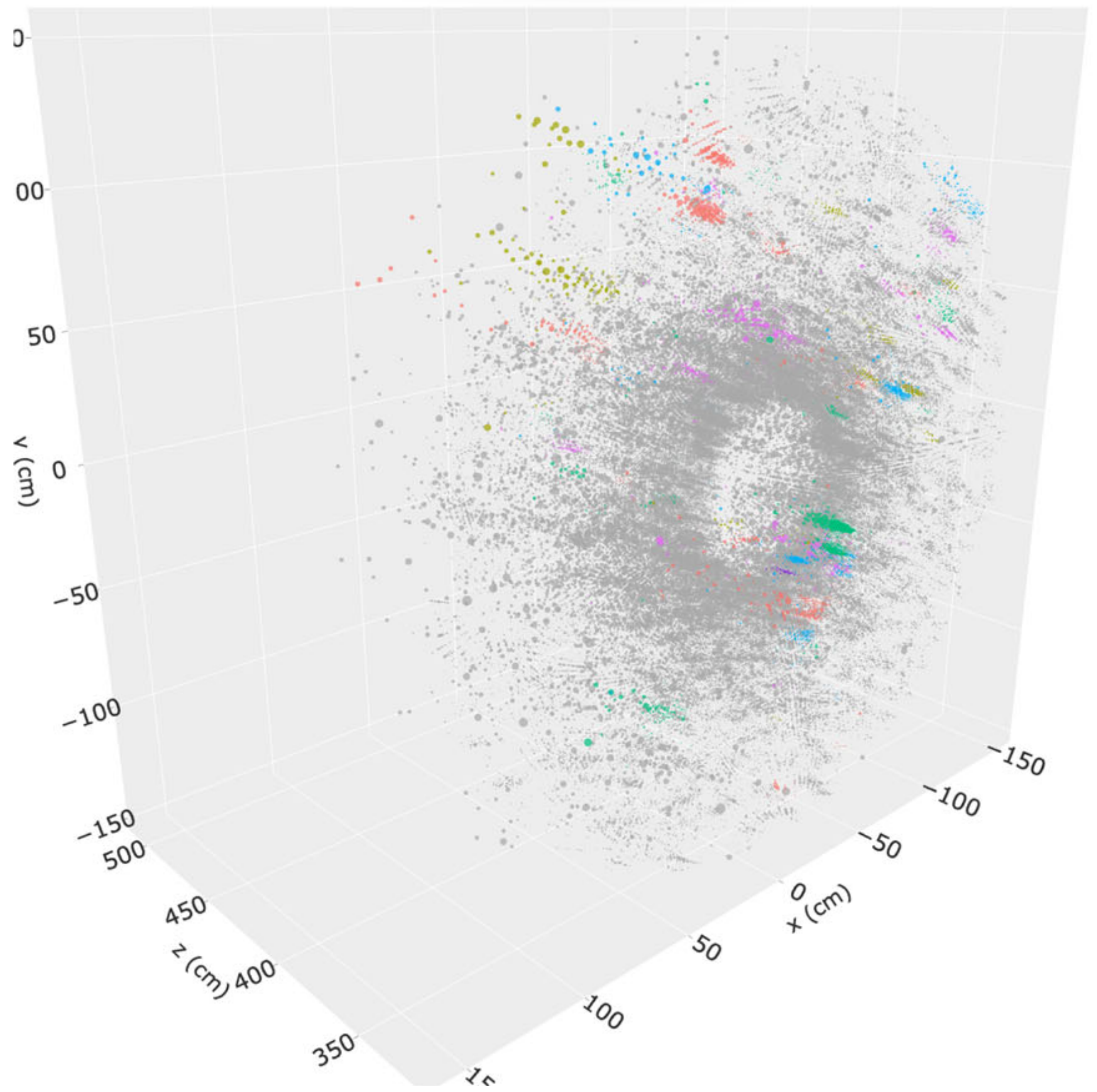
- Distance-weighted GNNs: GarNet/GravNet
  - much lower computational cost than DGCNN
  - GarNet: lightweight, can be implemented on FPGA for e.g., event triggering

- 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

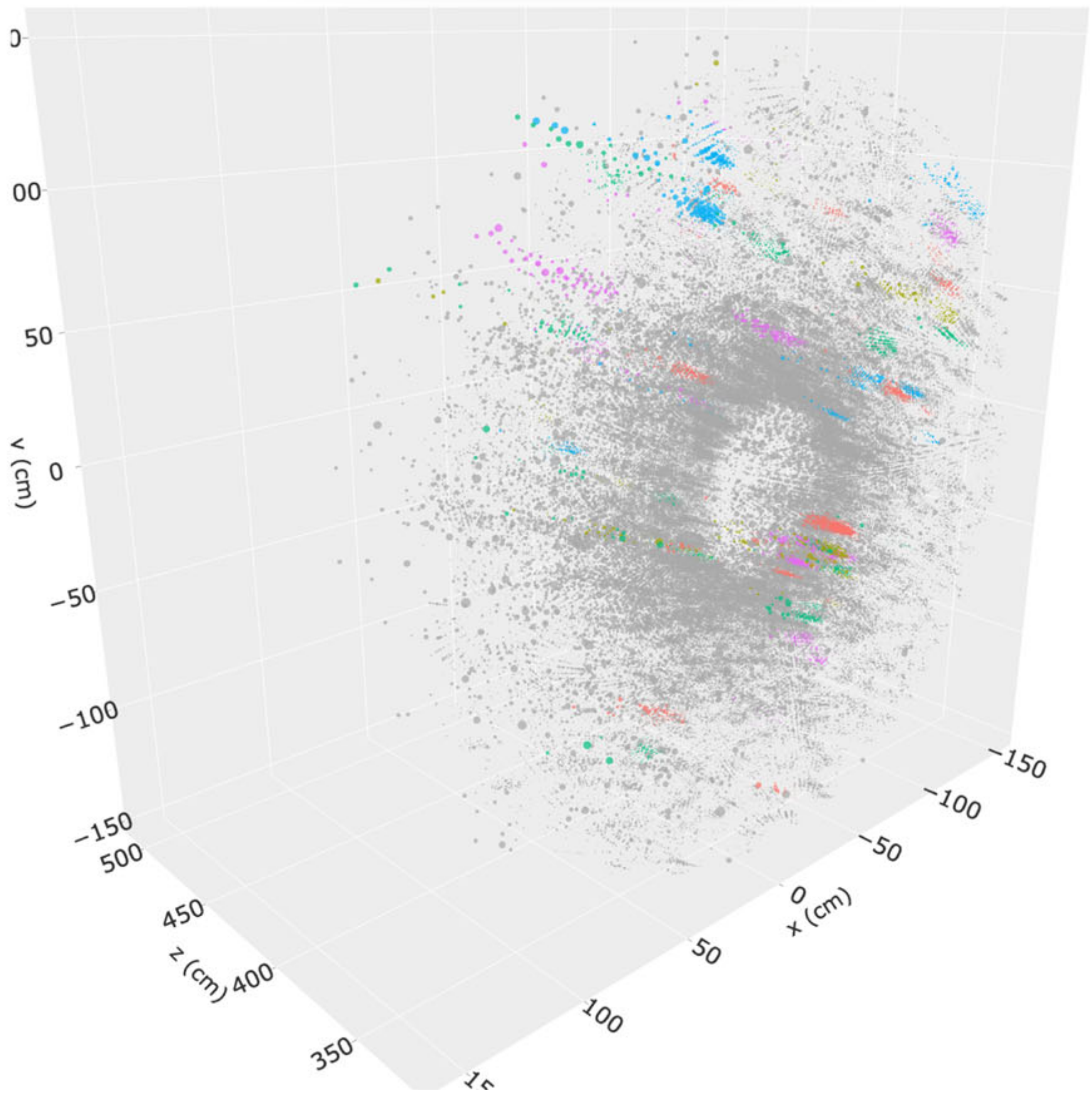
GarNet/GravNet



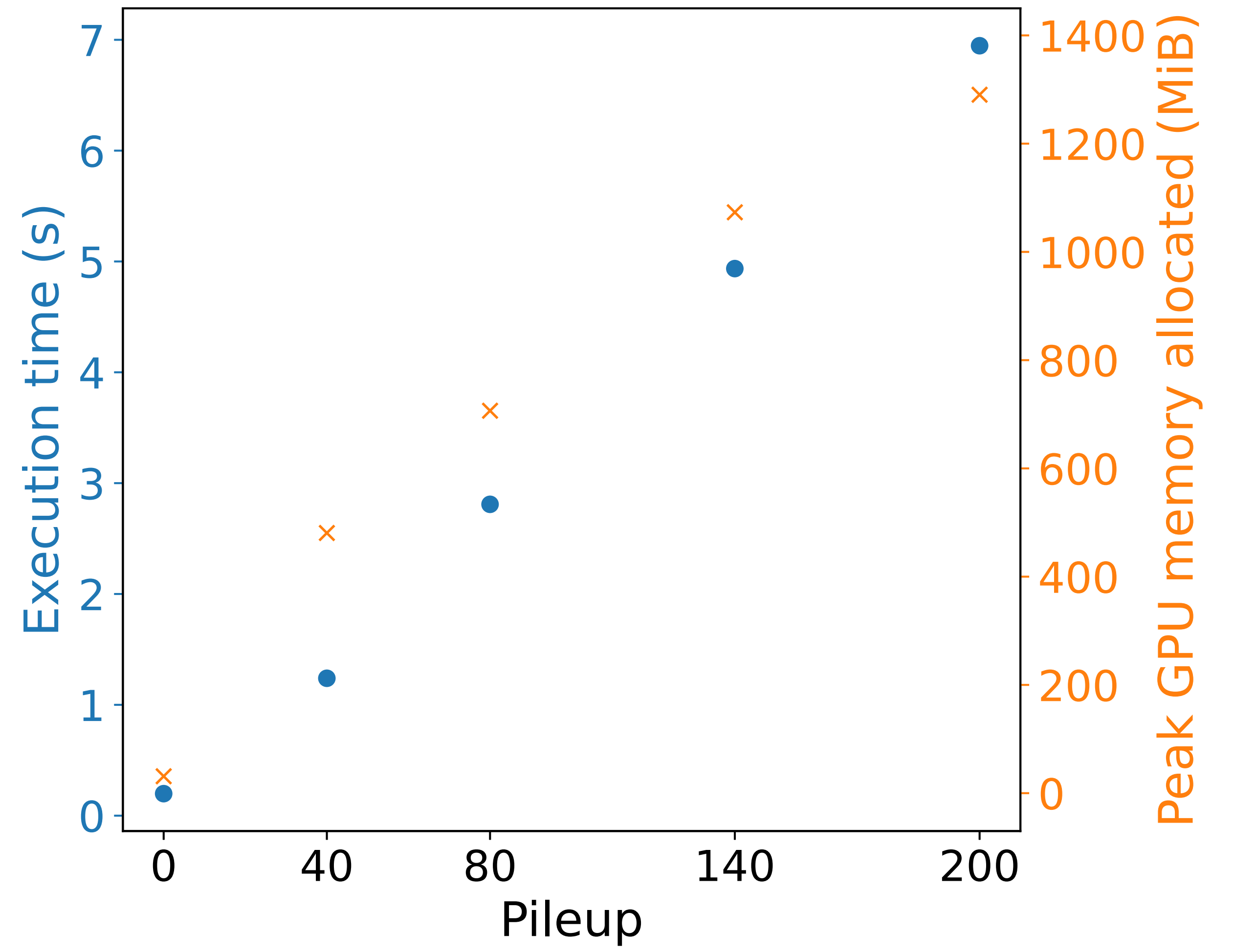
Truth



Reconstructed



Time and memory usage



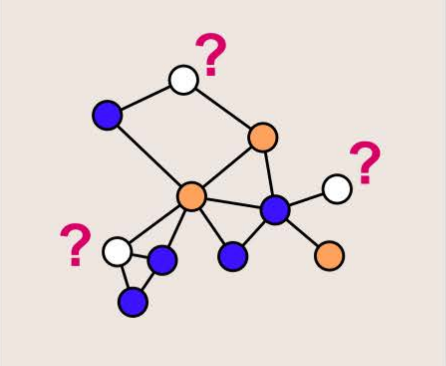
S. R. Qasim, J. Kieseler, Y. Iiyama and M. Pierini [EPJC 79 (2019) 7, 608]; J. Kieseler [EPJC 80 (2020) 9, 886]; S. R. Qasim et. al., [EPJC 82, 753 (2022)]

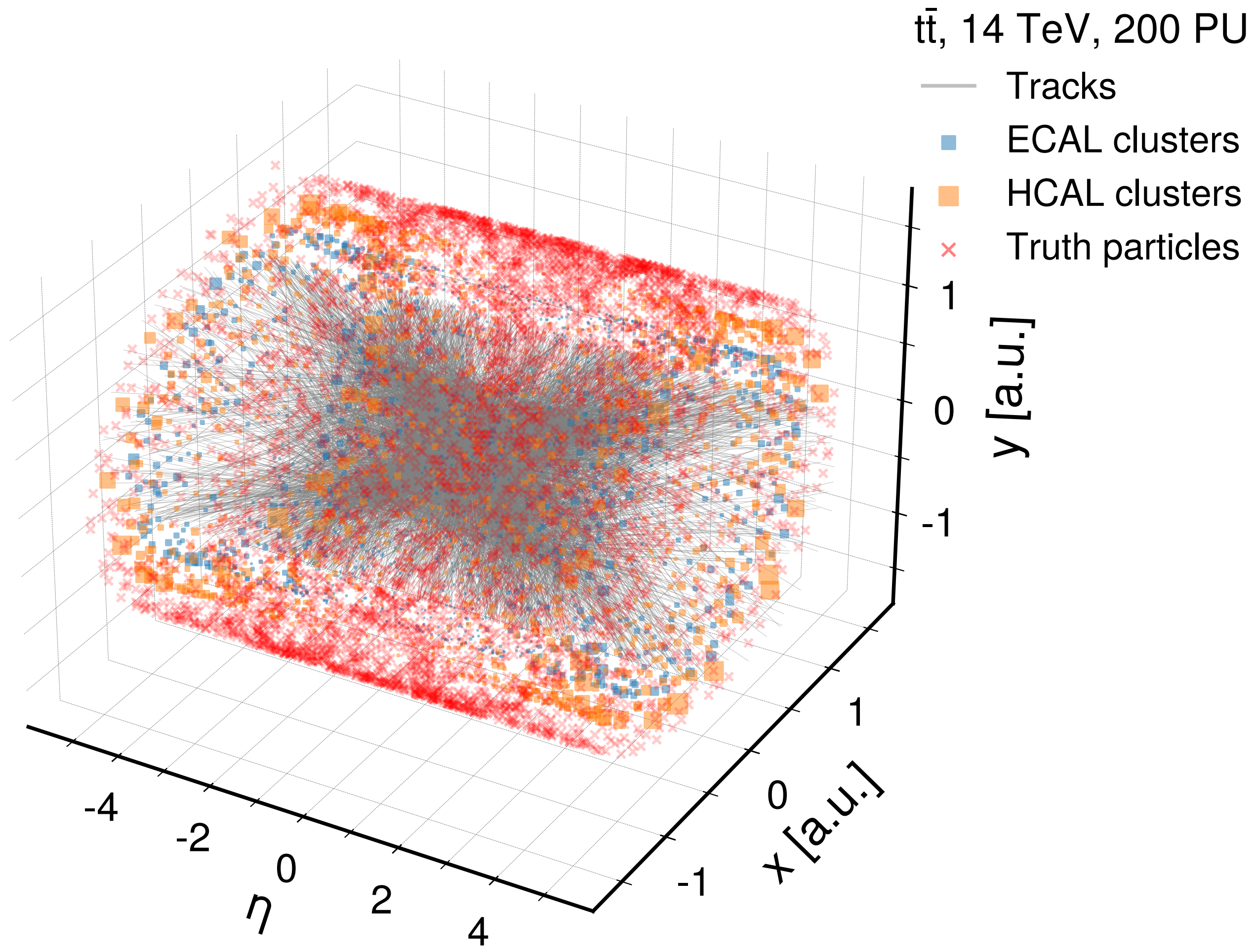
Particle Flow

J. Pata, J. Duarte, J. R. Vlimant, M. Pierini and M. Spiropulu [EPJC (2021) 81, 381]

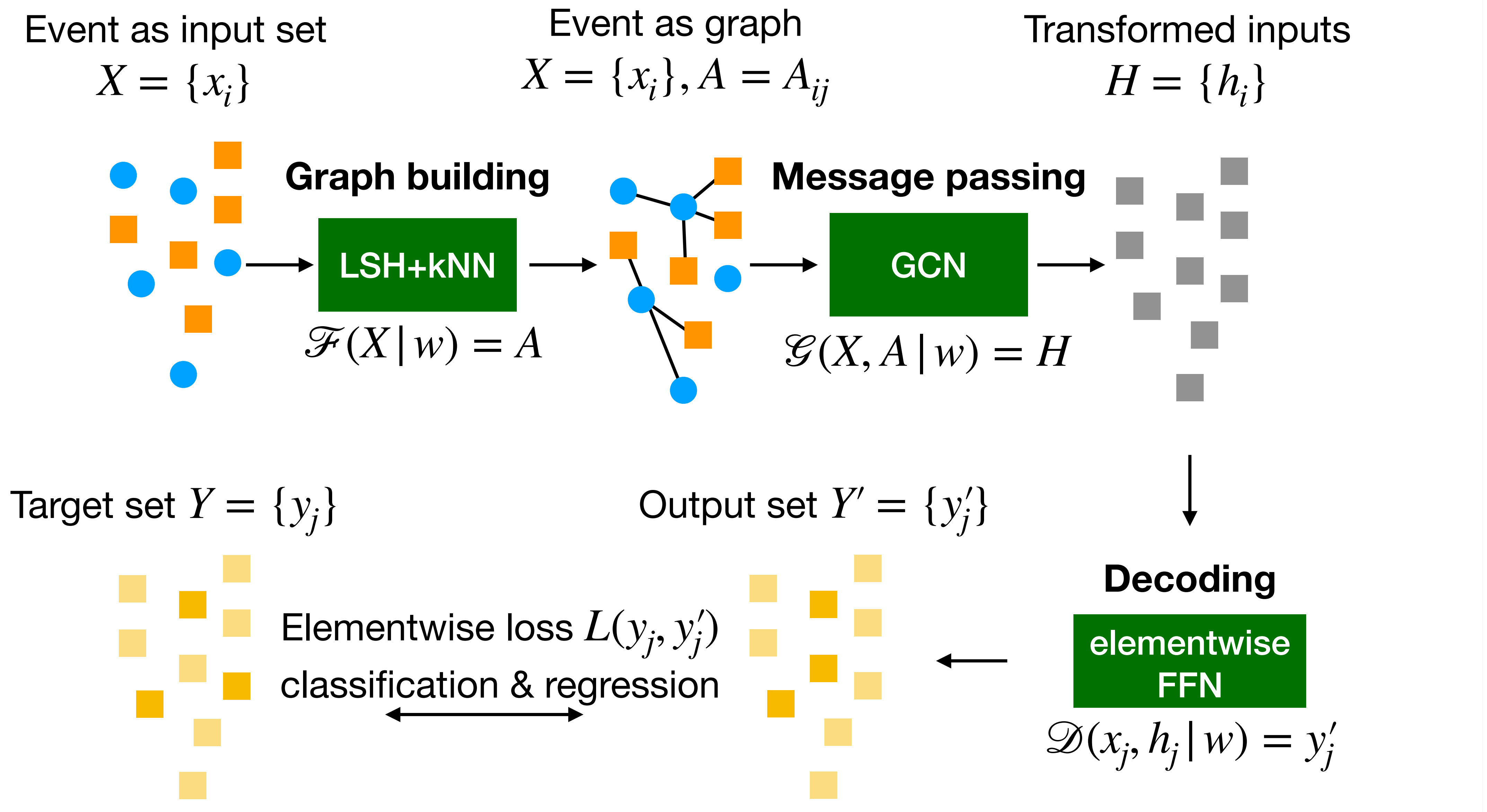
- Use GNNs to directly perform end-to-end particle flow reconstruction
  - comparable/better performance than rule-based PF on Delphes dataset
  - runtime scales linearly with input size, no quartic explosion

Node Classification





Delphes simulation



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

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

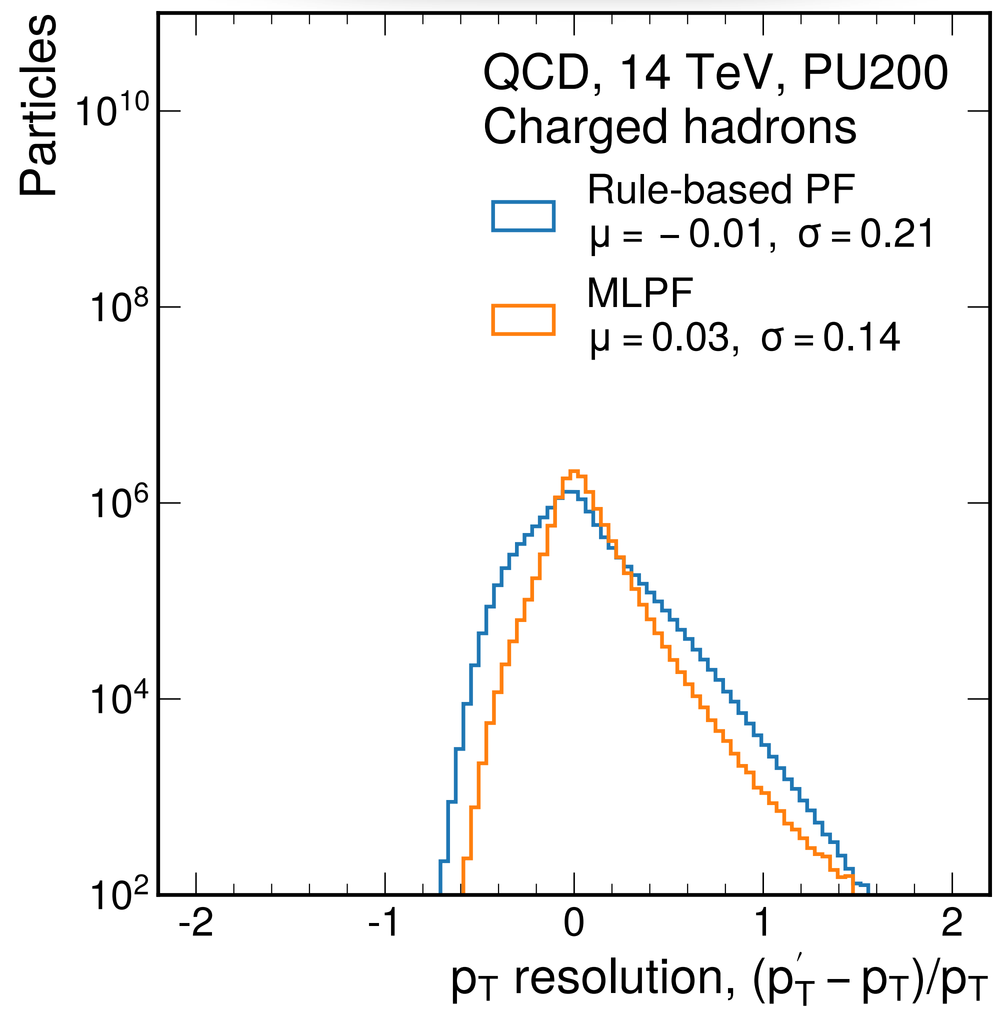
$h_i \in \mathbb{R}^{256}$

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

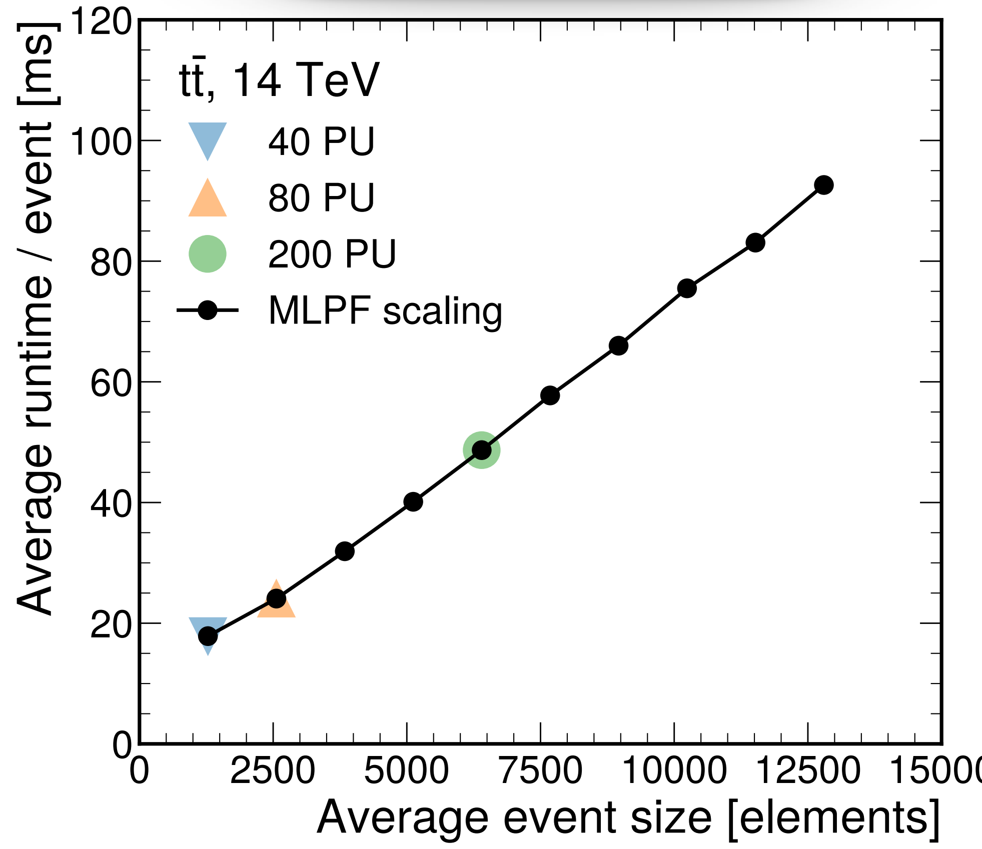
● - track, ■ - calorimeter cluster, ■ - encoded element

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

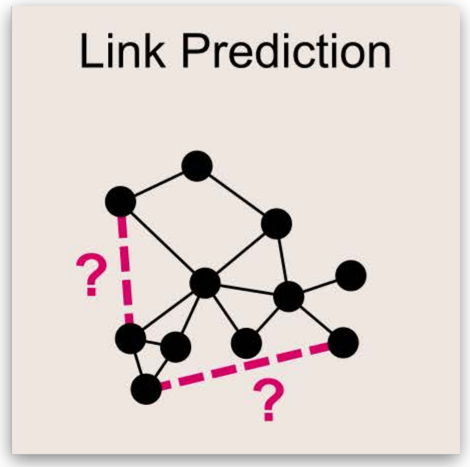
Resolution



Inference time

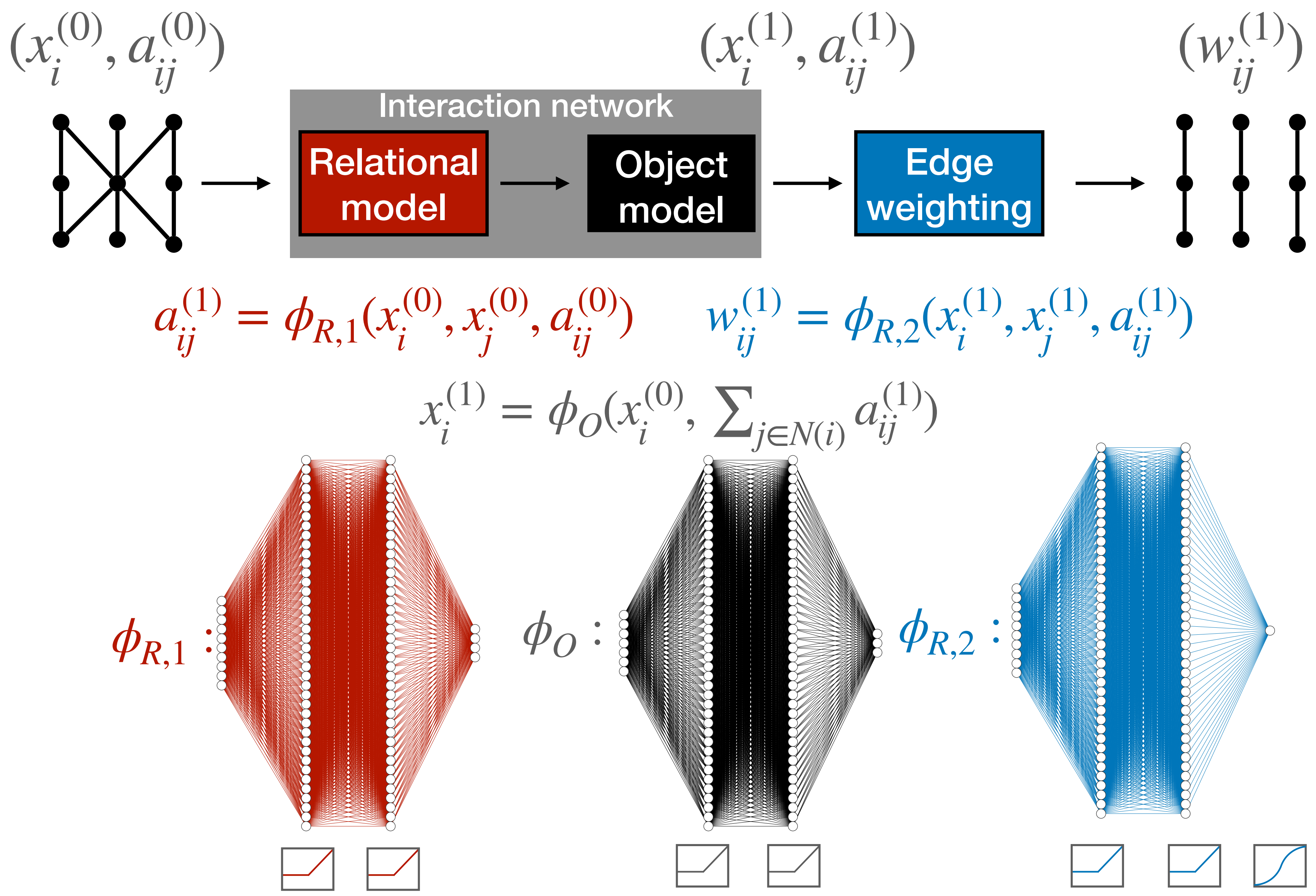


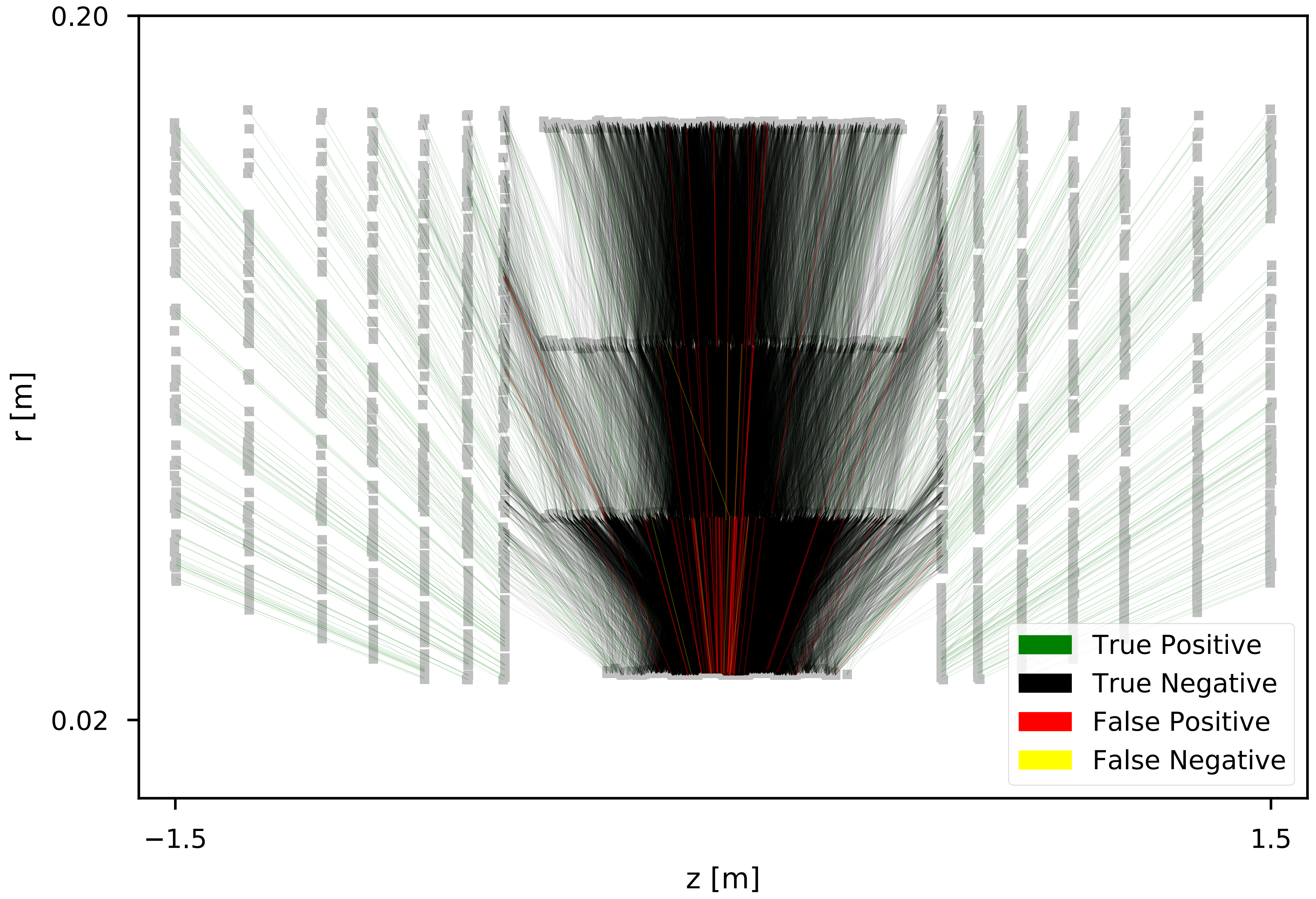
GNNs for Tracking



- 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

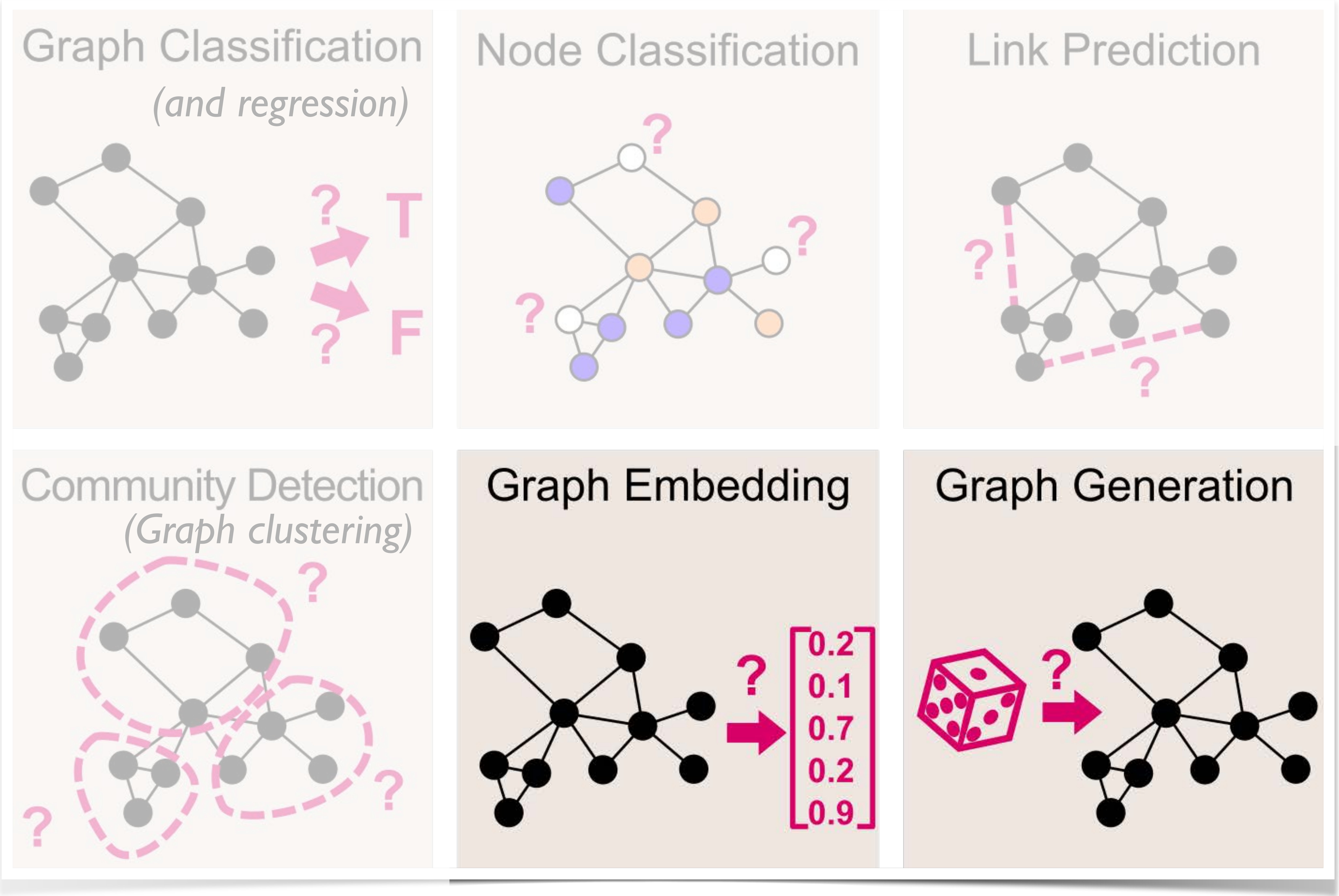
G. DeZoort et al. [Comput. Softw. Big Sci. 5, 26 (2021)]





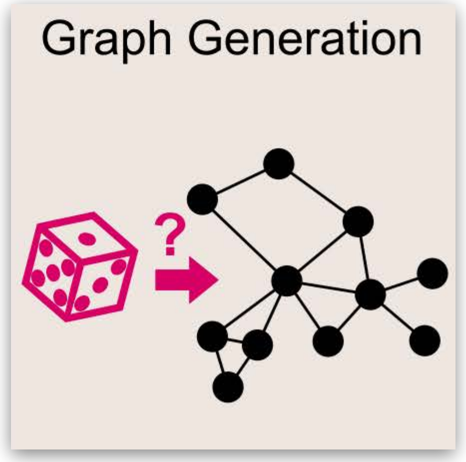
See also: S. Farrell et al. [1810.06111]; X. Ju et al. [2003.11603]; C. Biscarat, S. Caillou, C. Rougier, J. Stark and J. Zahreddine [2103.00916]; X. Ju et al. [2103.06995]; etc.

Graph Generative Models



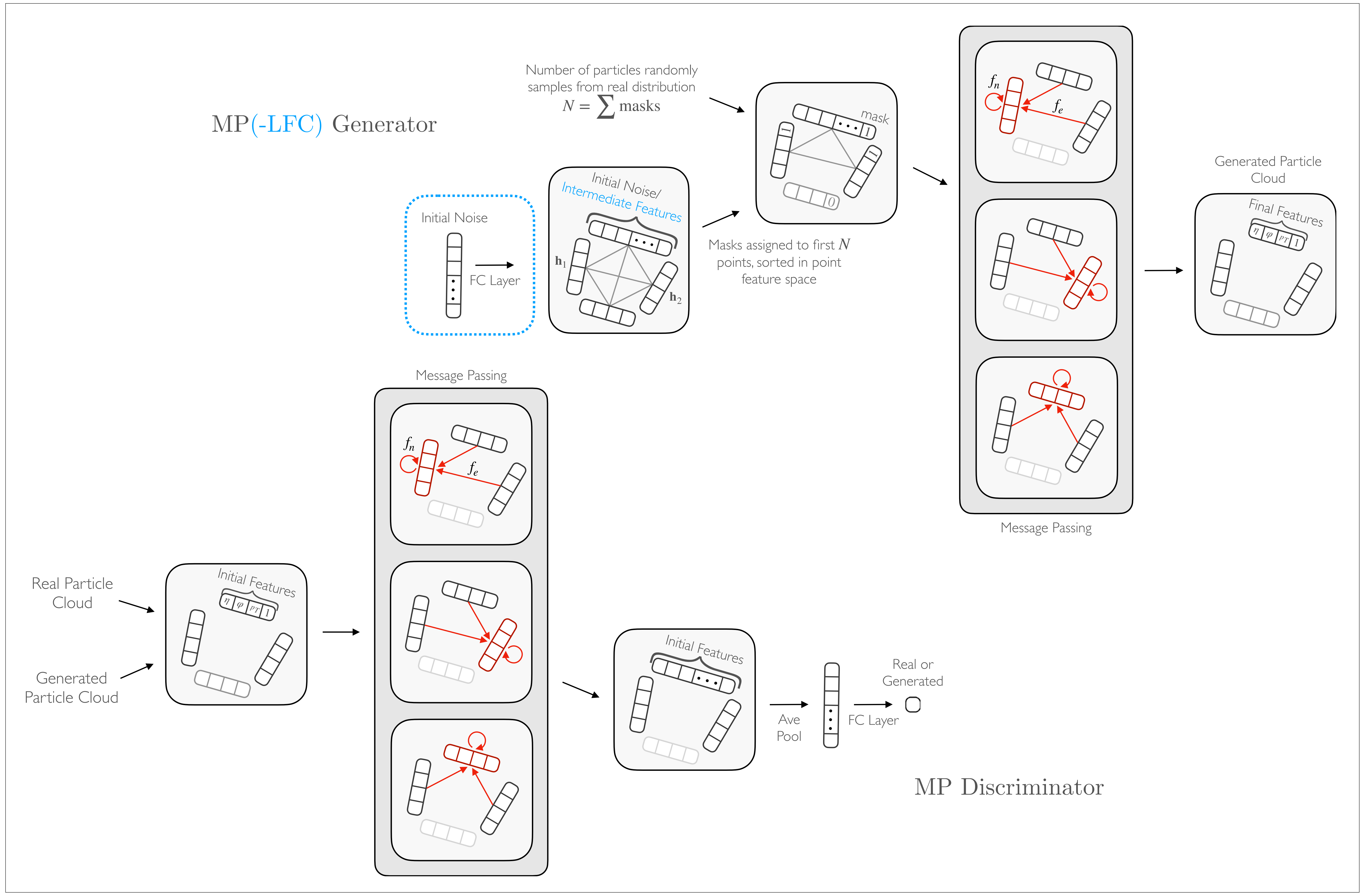
https://towardsdatascience.com/graph-convolutional-networks-deep-99d7fee5706f

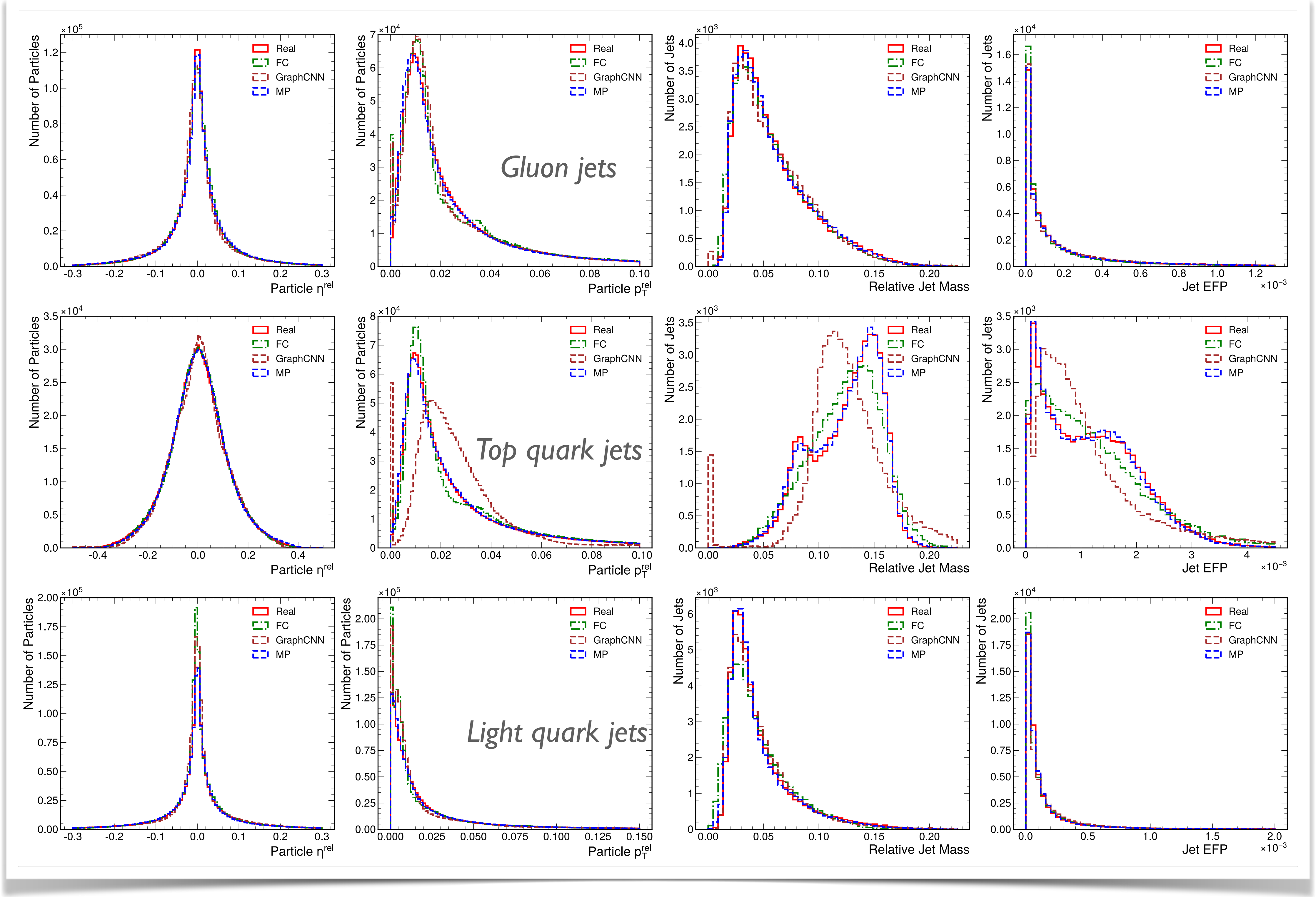
Particle Cloud Generation



- Exploit GNNs for "particle cloud" generation
  - enables fast detector simulation

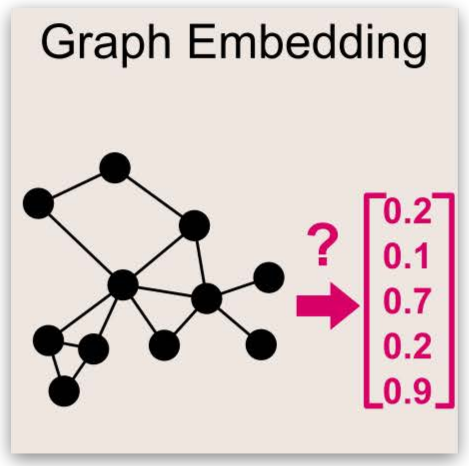
R. Kansal, J. Duarte, H. Su, B. Orzari, T. Tomei, M. Pierini, M. Touranakou, J. R. Vlimant and D. Gunopulos
[NeurIPS 2021]





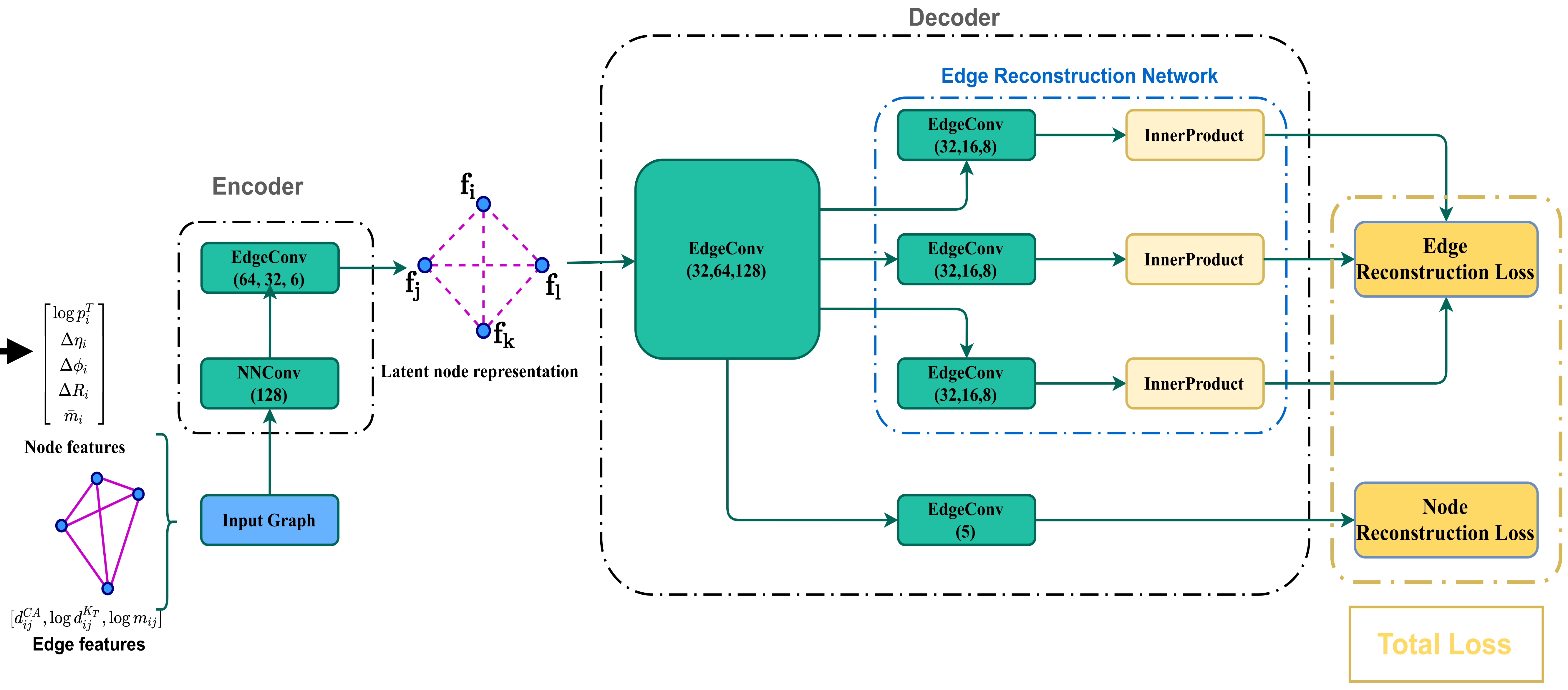
Anomaly Detection

O. Atkinson, A. Bhardwaj, C. Englert, V. S. Ngairangbam and M. Spannowsky [JHEP 08 (2021) 080]



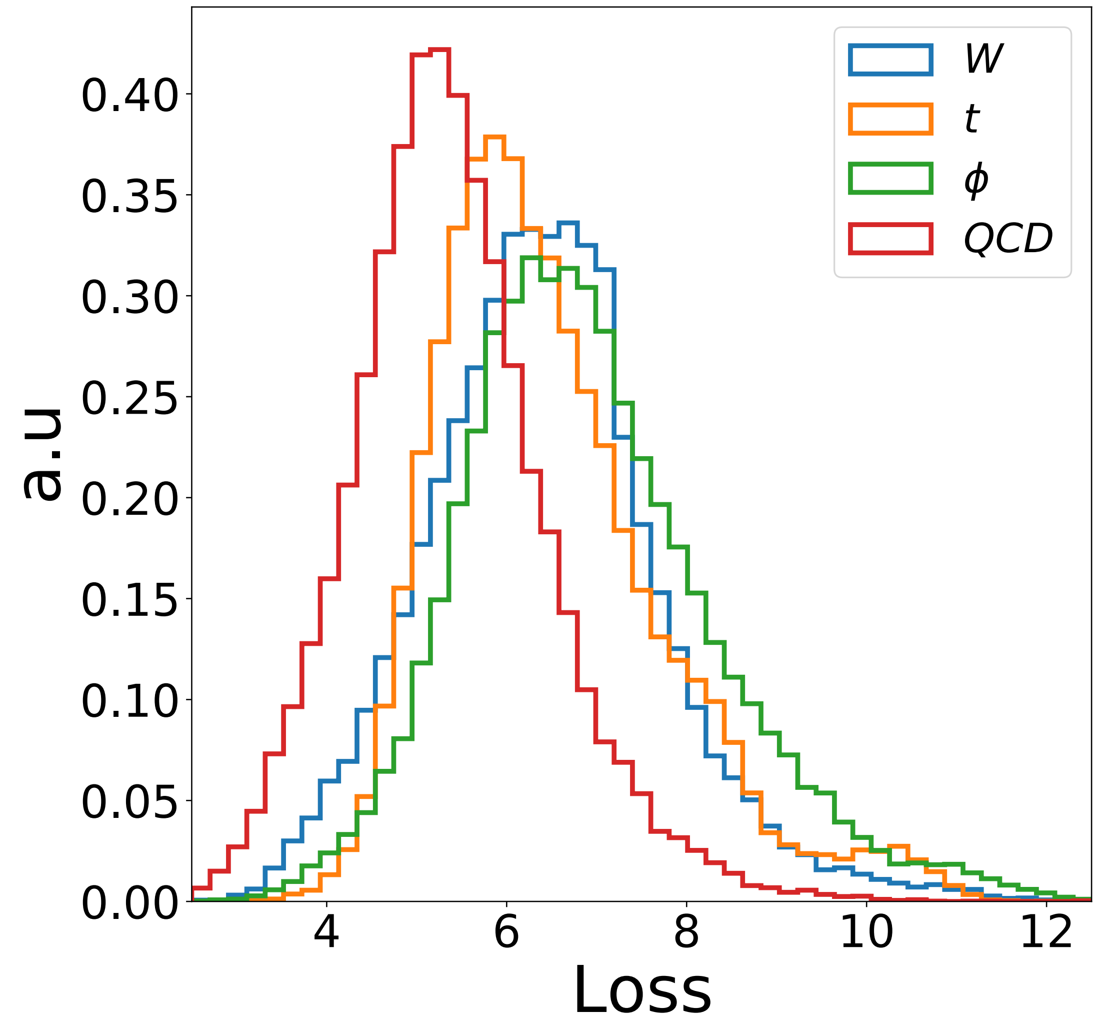
- GNN based autoencoders for anomaly detection
  - enables automated and model-agnostic new physics search

Recluster R=1.5 jets to R=0.1 microjets and use as inputs



$$L_{node} = \sqrt{\sum_{ia} \frac{(\hat{x}_i^a - x_i^a)^2}{N \times 5}}, \qquad L_{edge} = \sum_a \sqrt{\sum_{ij} \frac{(\hat{A}_{ij}^a - A_{ij}^a)^2}{N \times N}}$$

$$L_{auto} = \lambda_{node}\, L_{node} + \lambda_{edge}\, L_{edge}$$



The Road Ahead

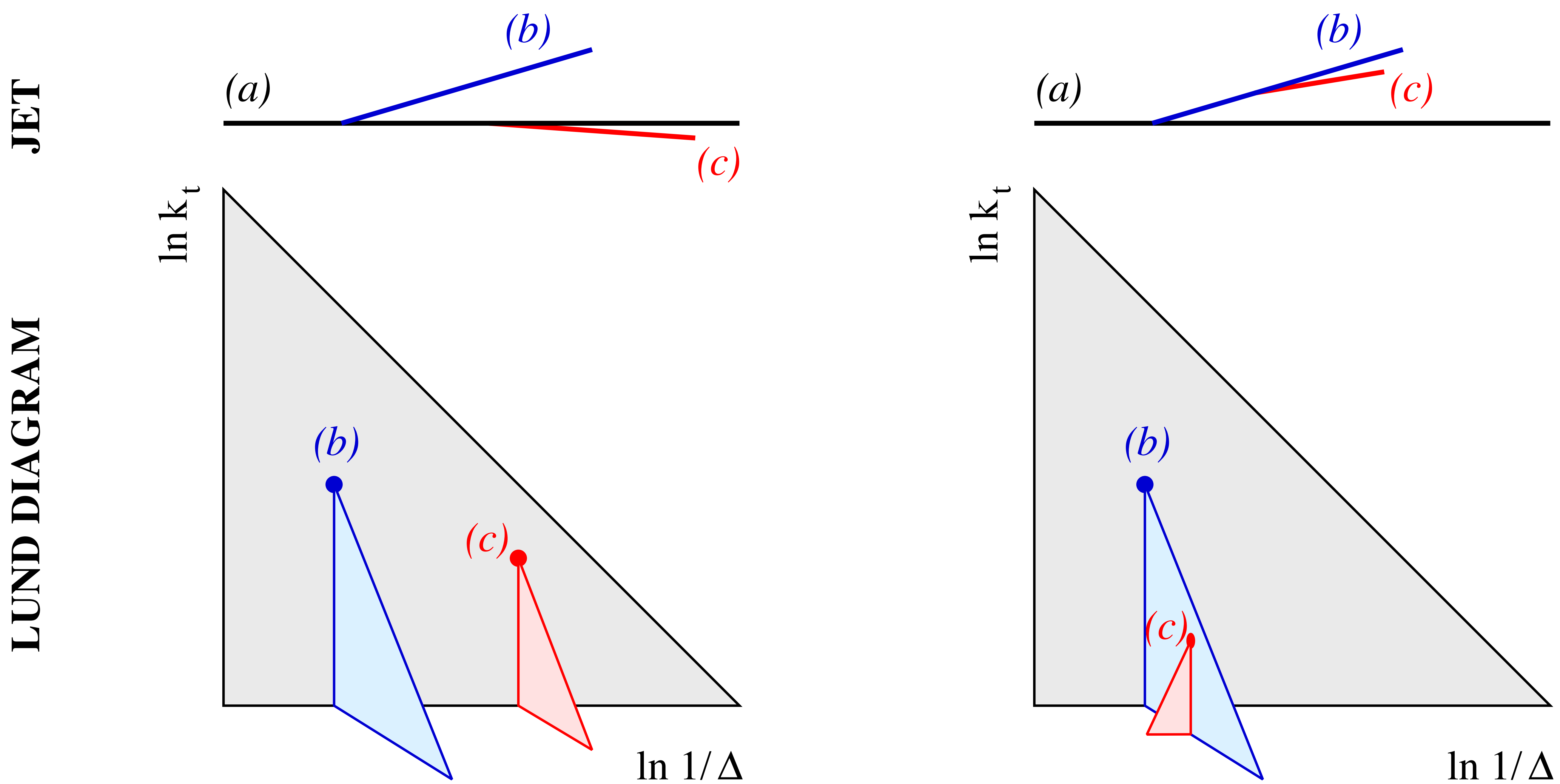
The Road Ahead

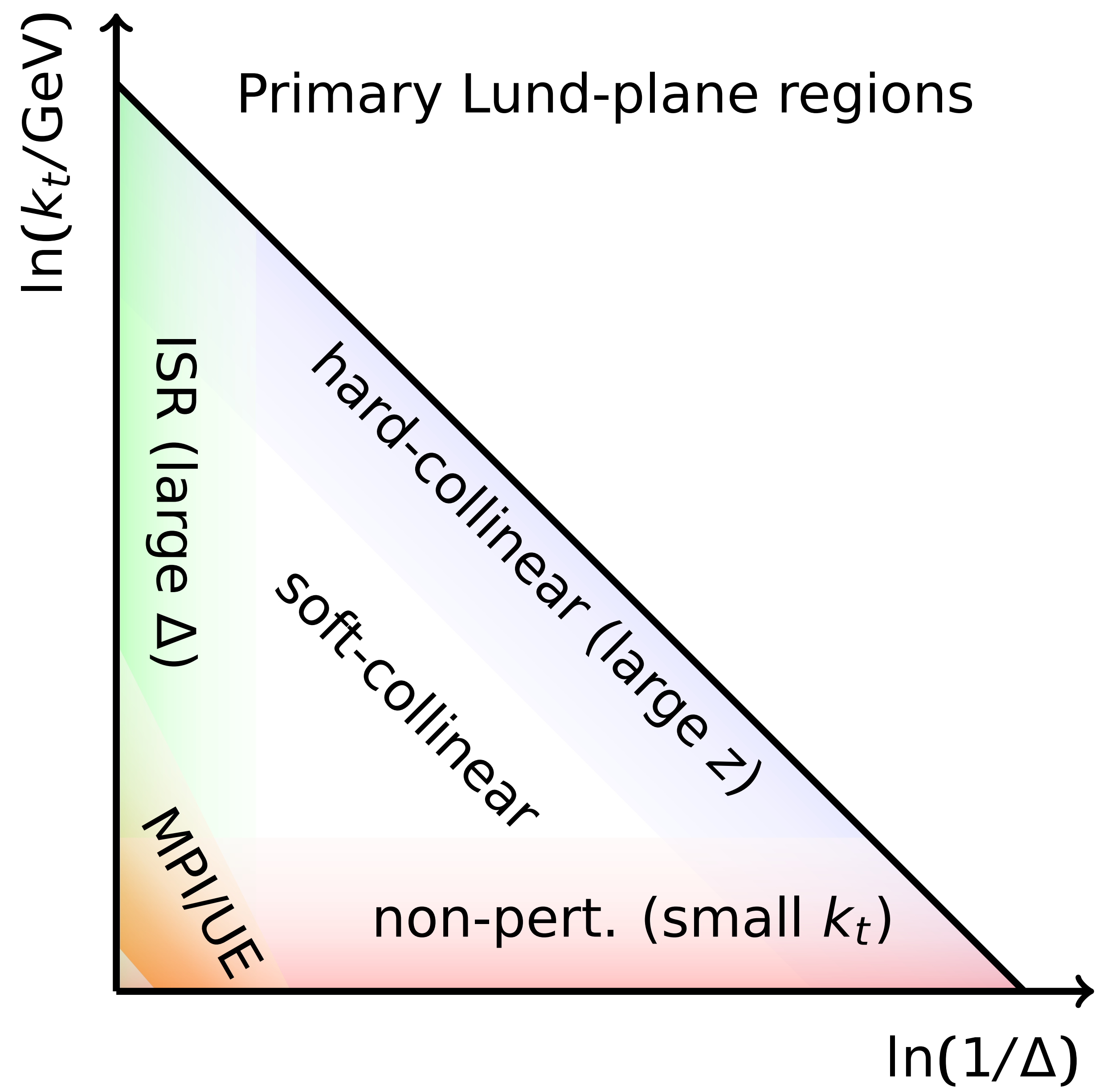
- Can we better incorporate physics knowledge into the network design?
    - physics aware data representation, symmetry group equivariant architecture, ...

JETS IN THE LUND PLANE

F. Dreyer, G. Salam and G. Soyez, JHEP 12 (2018) 064

- The Lund jet plane provides an efficient description of the radiation patterns within a jet



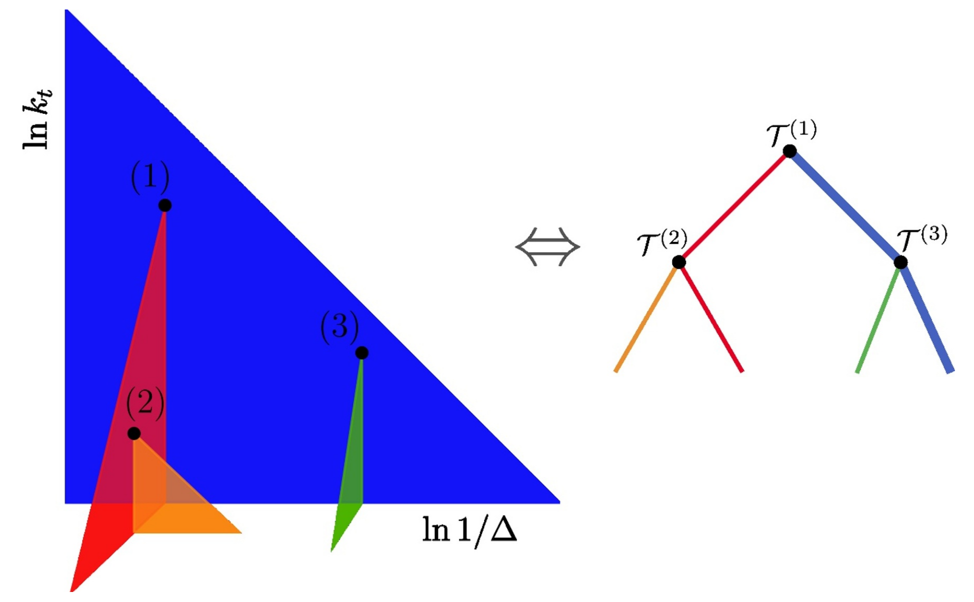


- each emission (splitting) is mapped to a point in the 2D (angle, transverse momentum) plane
  - further emissions (of the secondary particles) are represented in additional leaf planes
- different kinematic regimes are clearly separated in the Lund plane
- a natural input for ML algorithms on jets since it essentially encodes the full radiation patterns of a jet

LundNet

F. Dreyer and H. Qu, JHEP 03 (2021) 052

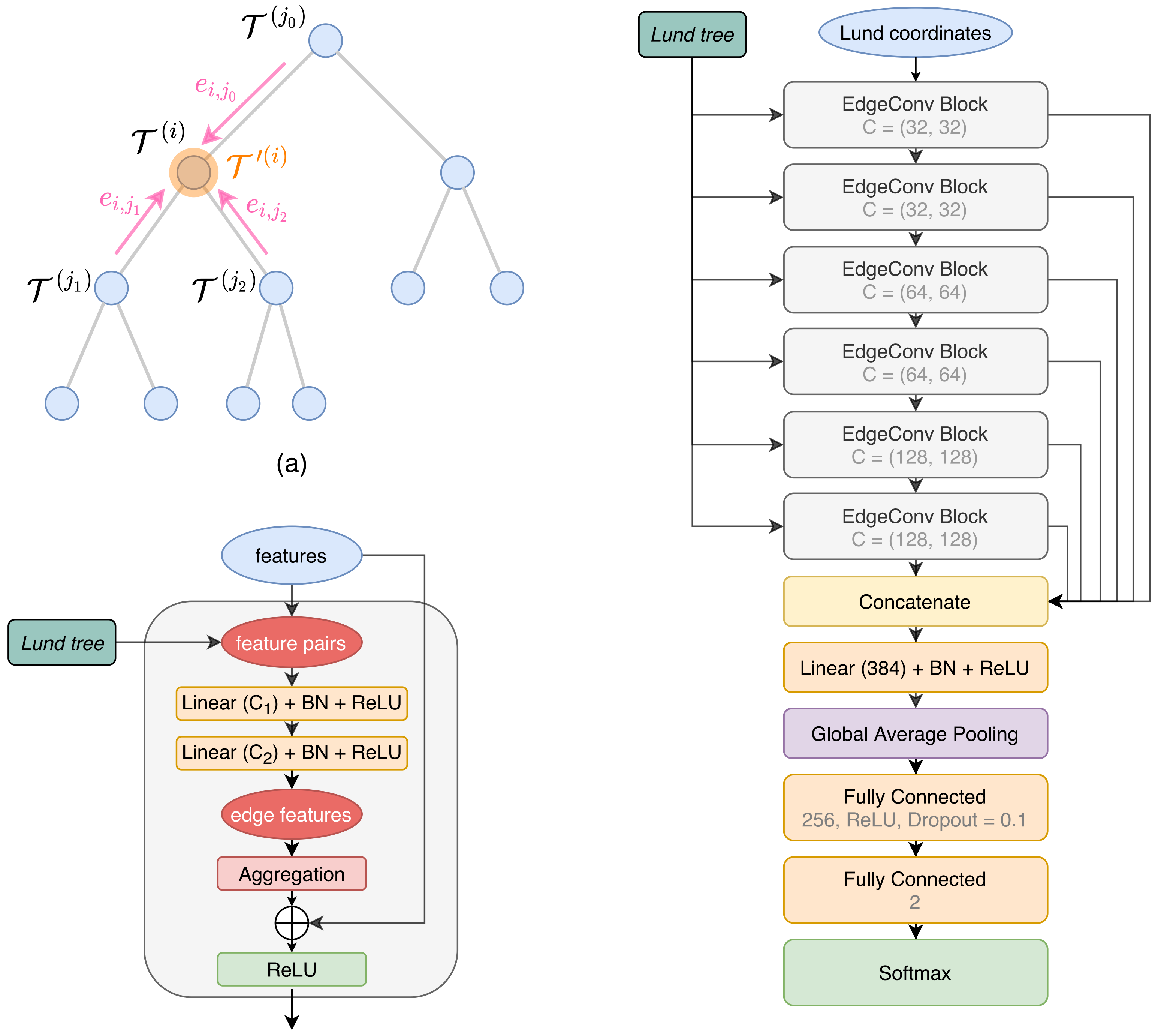
- LundNet: a graph neural network based on the Lund jet plane
  - technically, the input is a binary tree (from Cambridge/Aachen clustering)
    - equivalent to the **full** Lund plane
  - each node corresponds to an emission
    - a set of variables are be defined for the current splitting



$$\Delta^2 = (y_a - y_b)^2 + (\phi_a - \phi_b)^2, \quad k_t \equiv p_{tb}\Delta_{ab}, \quad m^2 \equiv (p_a + p_b)^2,$$

$$z \equiv \frac{p_{tb}}{p_{ta}+p_{tb}}, \quad \kappa \equiv z\Delta\,, \quad \psi \equiv \tan^{-1}\frac{y_b - y_a}{\phi_b - \phi_a},$$

- Similar network architecture as ParticleNet
  - but the graph structure is fixed by the Lund tree
    - instead of the (dynamic) k-nearest neighbors
- Two variants of LundNet studied
  - LundNet-5: using all five Lund variables, $(\ln k_t, \ln\Delta, \ln z, \ln m, \psi)$
  - LundNet-3: using only three Lund variables, $(\ln k_t, \ln\Delta, \ln z)$

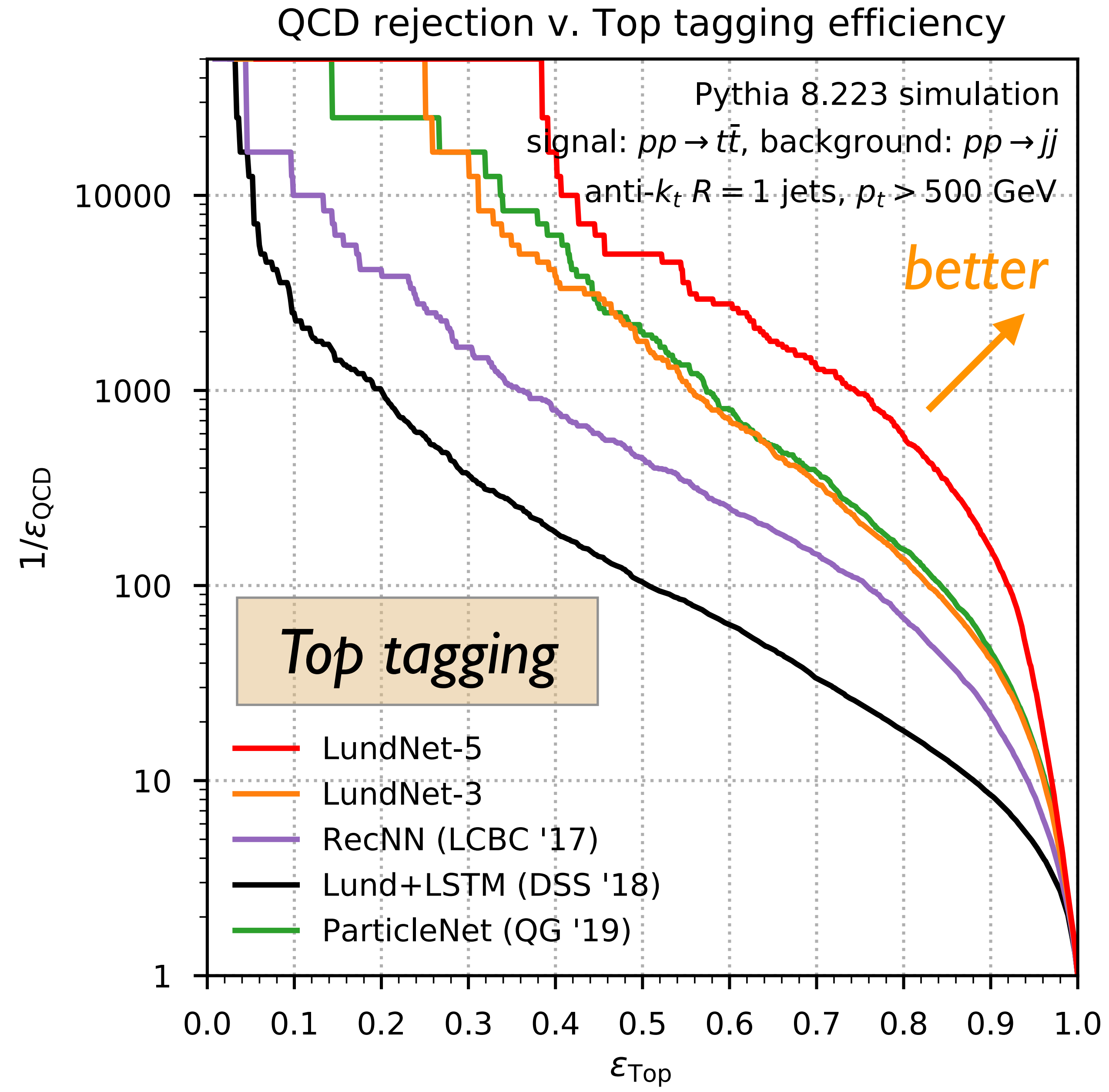


(a) (b) (c)

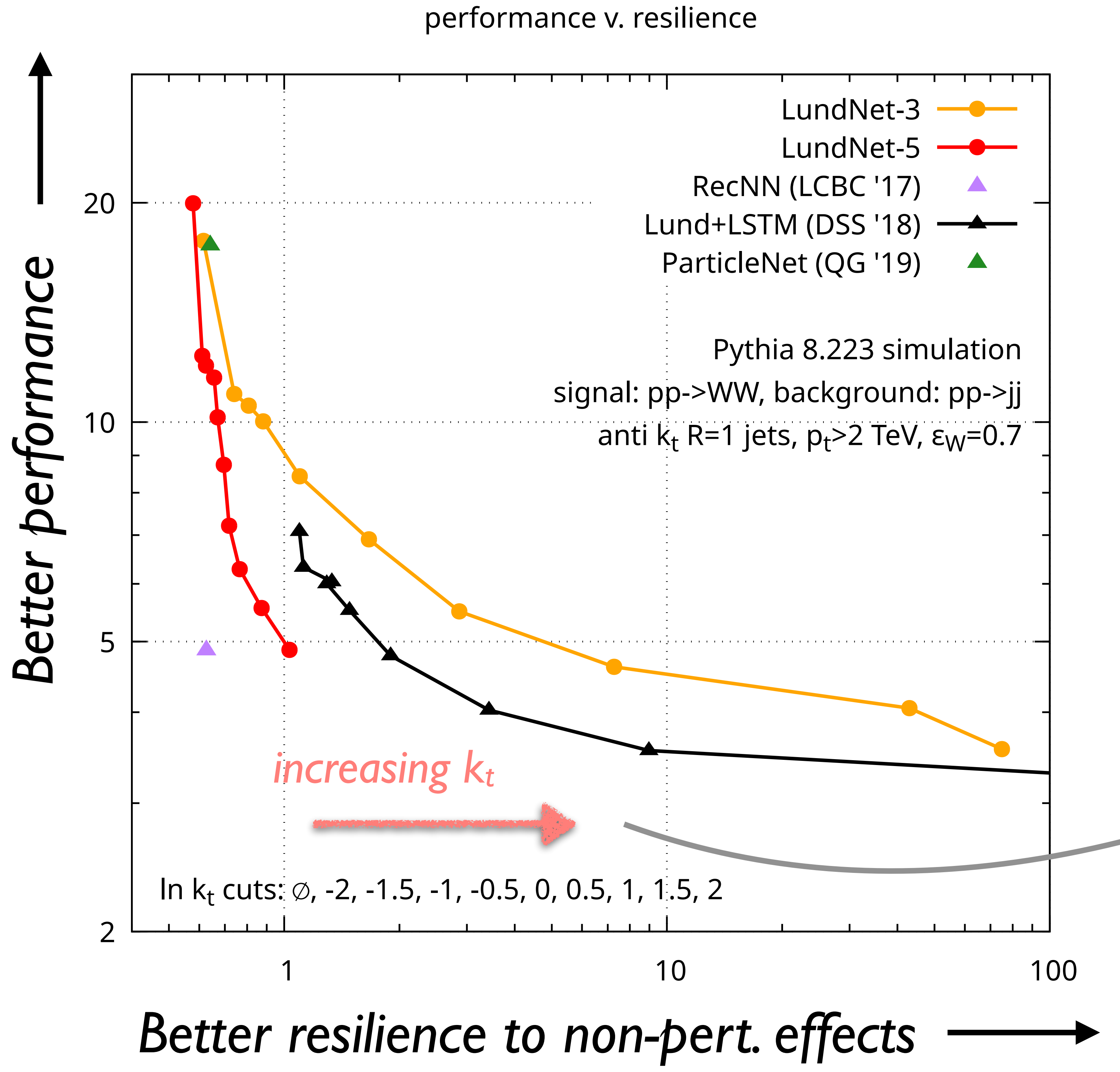
LundNet: Performance

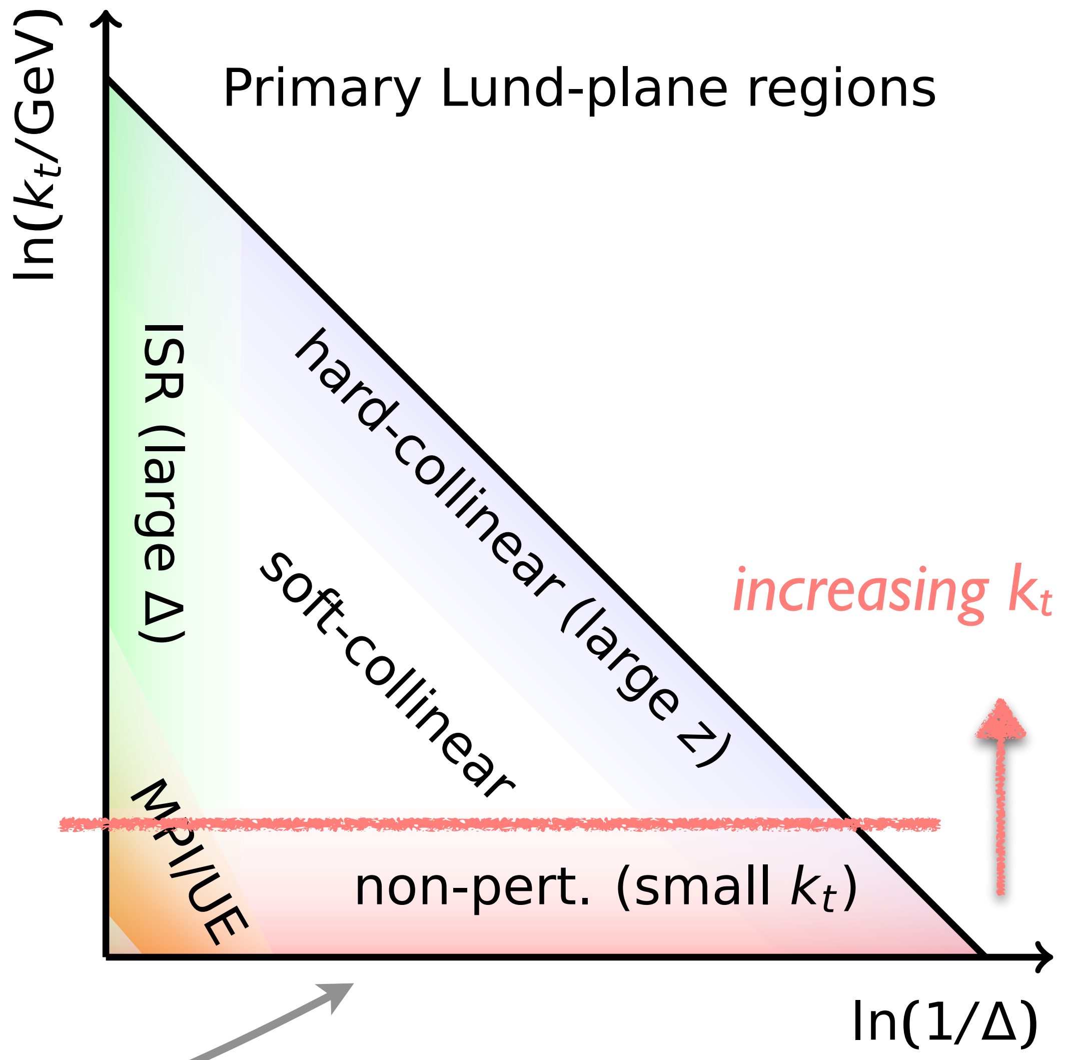
F. Dreyer and H. Qu, JHEP 03 (2021) 052

- LundNet achieves very high performance at significant lower computational cost than ParticleNet
  - due to fewer number of neighbors in a binary tree & static graph structure
- Moreover, LundNet provides a systematic way to control the robustness of the tagger
  - the non-perturbative region can be effectively rejected by applying a k_t cut on the Lund plane



	Number of parameters	Training time [ms/sample/epoch]	Inference time [ms/sample]
LundNet	395k	0.472	0.117
ParticleNet	369k	3.488	1.036
Lund+LSTM	67k	0.424	0.131





** Resilience assessed by applying the model trained on hadron-level samples to parton-level samples and compare the difference*

LorentzNet

- Incorporating Lorentz symmetry into graph neural network architecture

S. Gong, Q. Meng, J. Zhang, HQ, C. Li, S. Qian, W. Du, Z. M. Ma and T.Y. Liu, JHEP 07 (2022) 030

Lorentz 4-vector
Lorentz scalar

Coordinate input: x^0

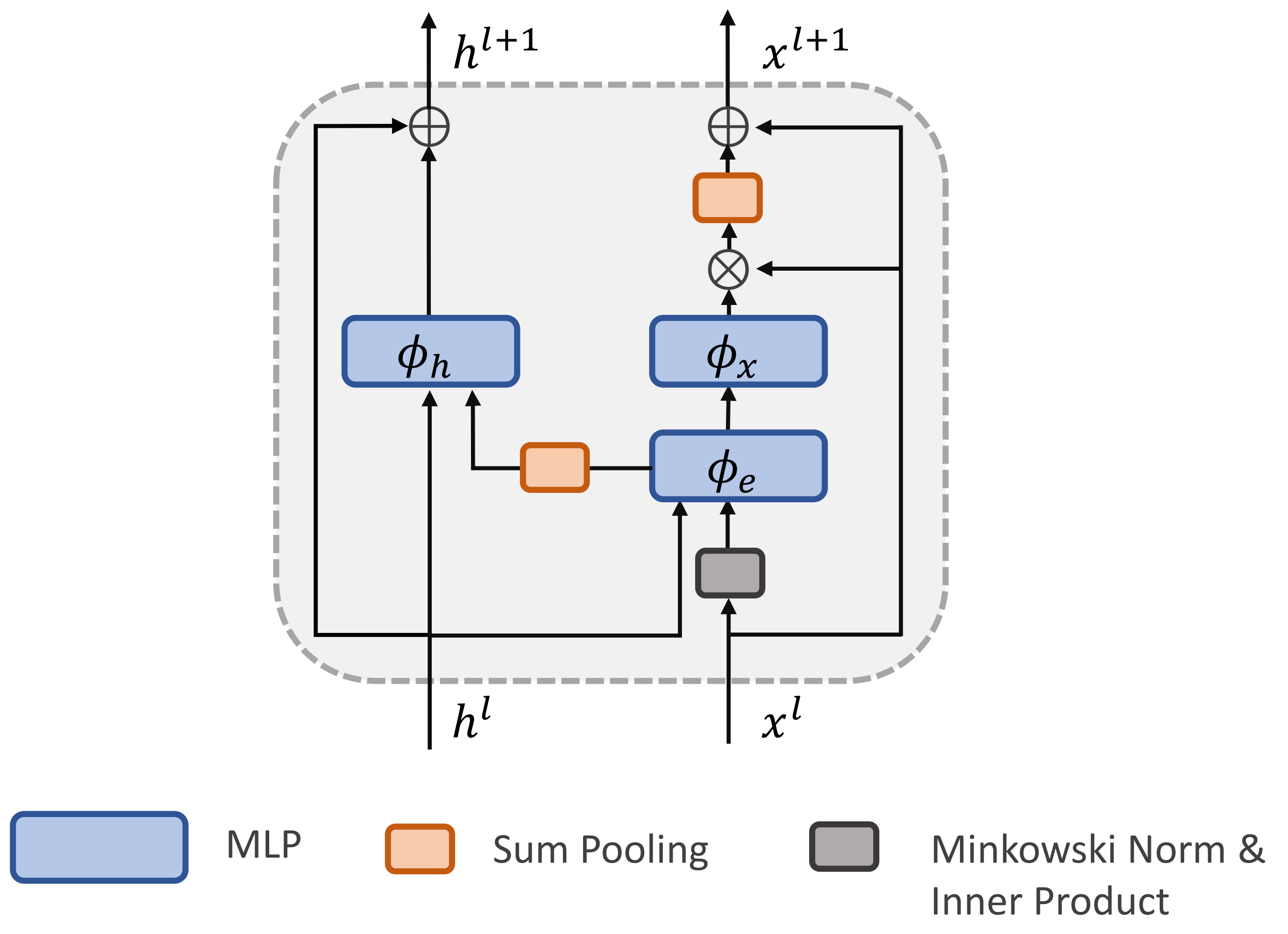
Feature input: h_i^0

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

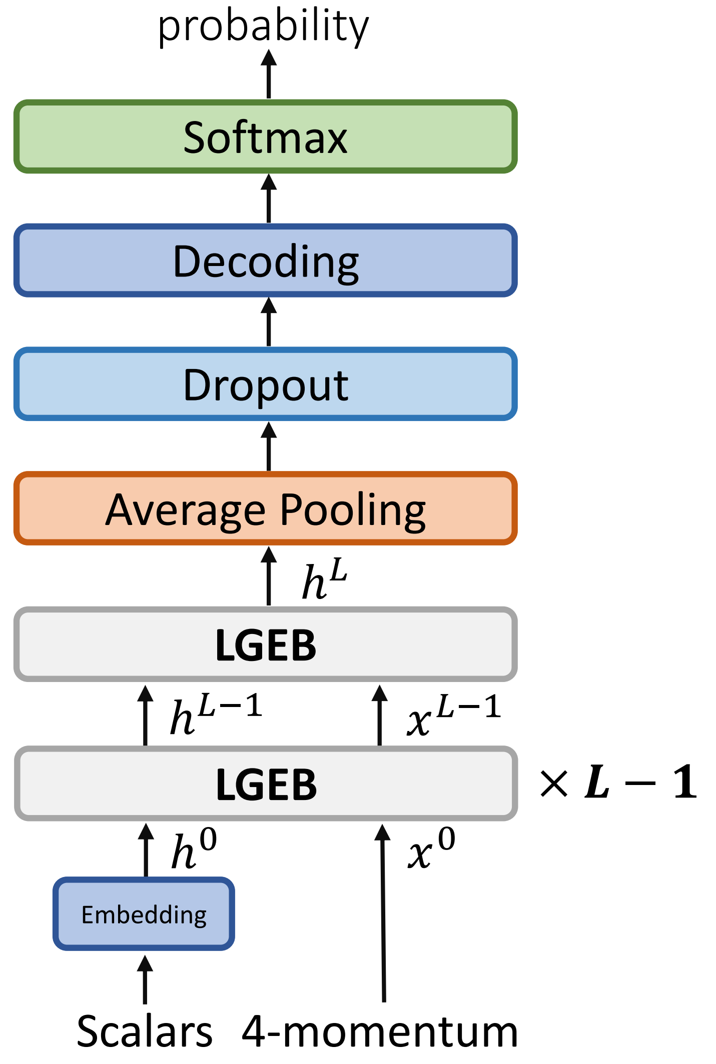
Scalars — Pairwise Lorentz invariants

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)



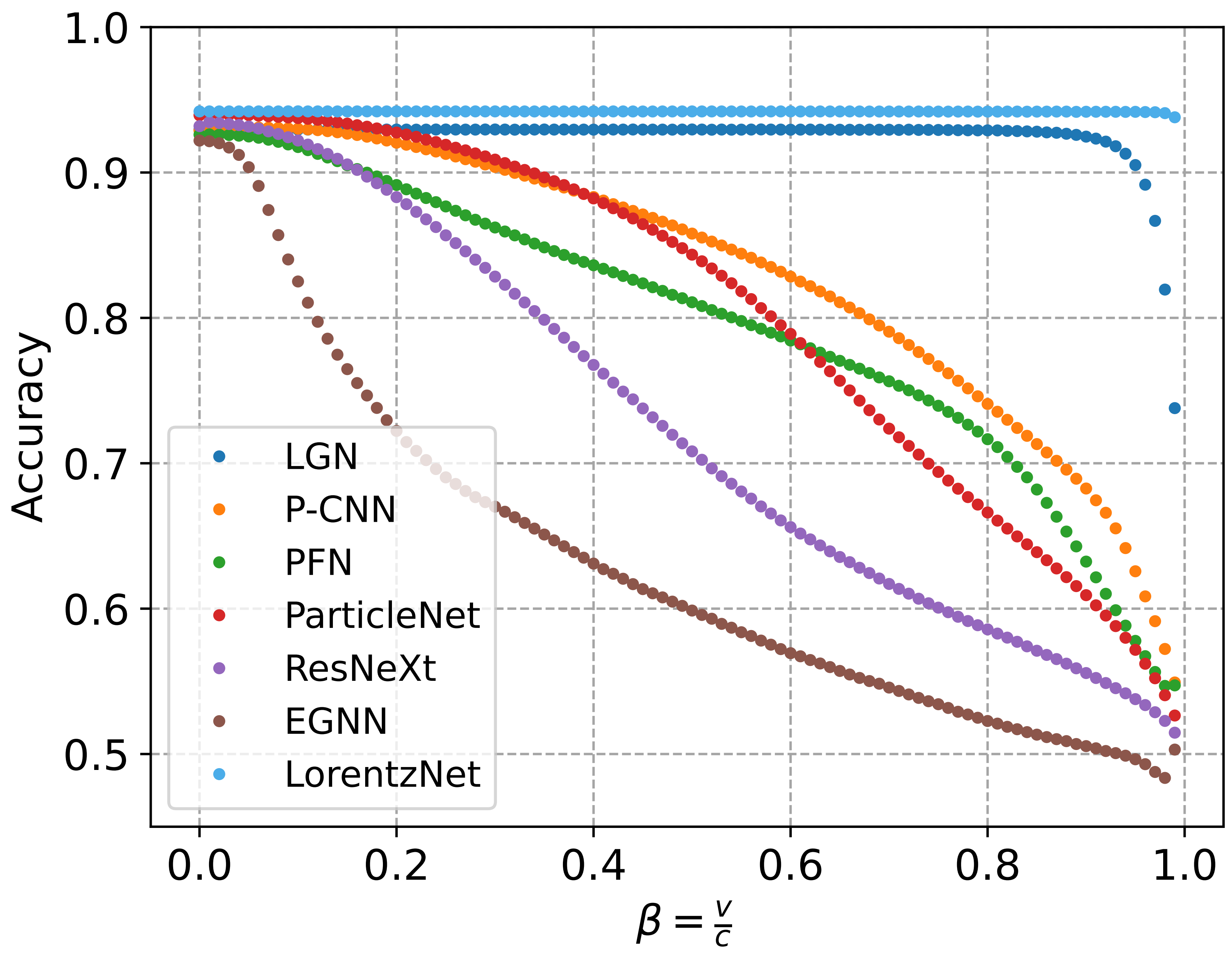
LorentzNet

cf. A. Bogatskiy, B. Anderson, J. Offermann, M. Roussi, D. Miller and R. Kondor, arXiv: 2006.04780 ["LGN"]; A. Bogatskiy, T. Hoffman, D. Miller and J. Offermann, arXiv: 2211.00454 ["PELICAN"]; I. Batatia, M. Geiger, J. Munoz, T. Smidt, L. Silberman and C. Ortner, arXiv: 2306.00091 ["lie-nn"];

LorentzNet: Benefits from Symmetry

- Benefits from the symmetry preservation
  - model response invariant under Lorentz transformation
  - sample efficiency: incorporation of Lorentz symmetry allows to train with very few samples

Model stability under Lorentz boost



Performance when trained on a fraction of the top-tagging dataset

Training Fraction	Model	Accuracy	AUC	$1/\varepsilon_B$ ($\varepsilon_S = 0.5$)	$1/\varepsilon_B$ ($\varepsilon_S = 0.3$)
0.5% (~6k jets)	ParticleNet	0.913	0.9687	77 ± 4	199 ± 14
	LorentzNet	**0.929**	**0.9793**	$\mathbf{176 \pm 14}$	$\mathbf{562 \pm 72}$
1%	ParticleNet	0.919	0.9734	103 ± 5	287 ± 19
	LorentzNet	**0.932**	**0.9812**	$\mathbf{209 \pm 5}$	$\mathbf{697 \pm 58}$
5%	ParticleNet	0.931	0.9807	195 ± 4	609 ± 35
	LorentzNet	**0.937**	**0.9839**	$\mathbf{293 \pm 12}$	$\mathbf{1108 \pm 84}$

THE ROAD AHEAD

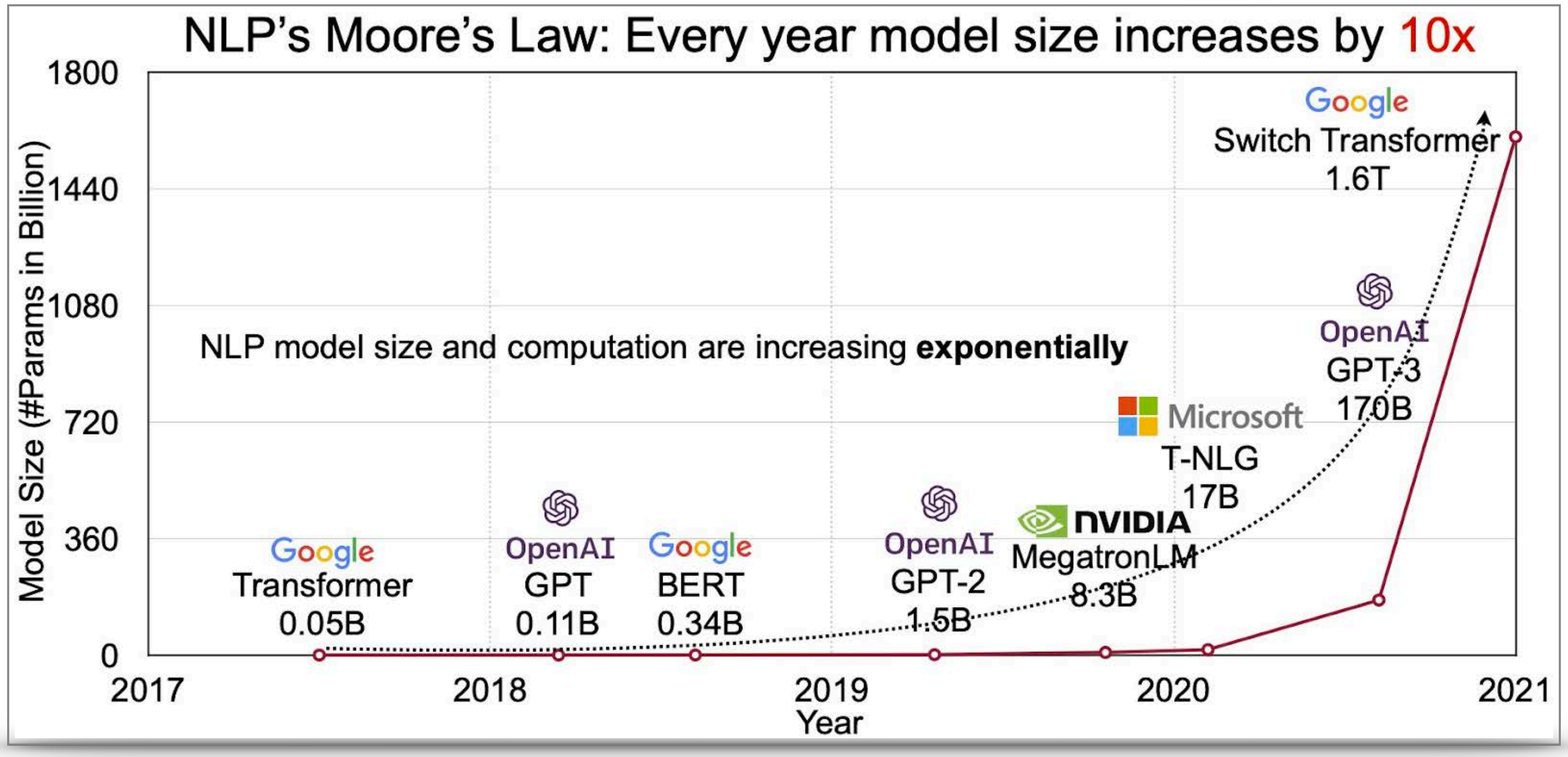
- Can we better incorporate physics knowledge into the network design?
  - physics aware data representation, symmetry group equivariant architecture, ...

- Can we scale up to a large model for HEP?
  - large datasets, pre-training, multi-modal learning, ...

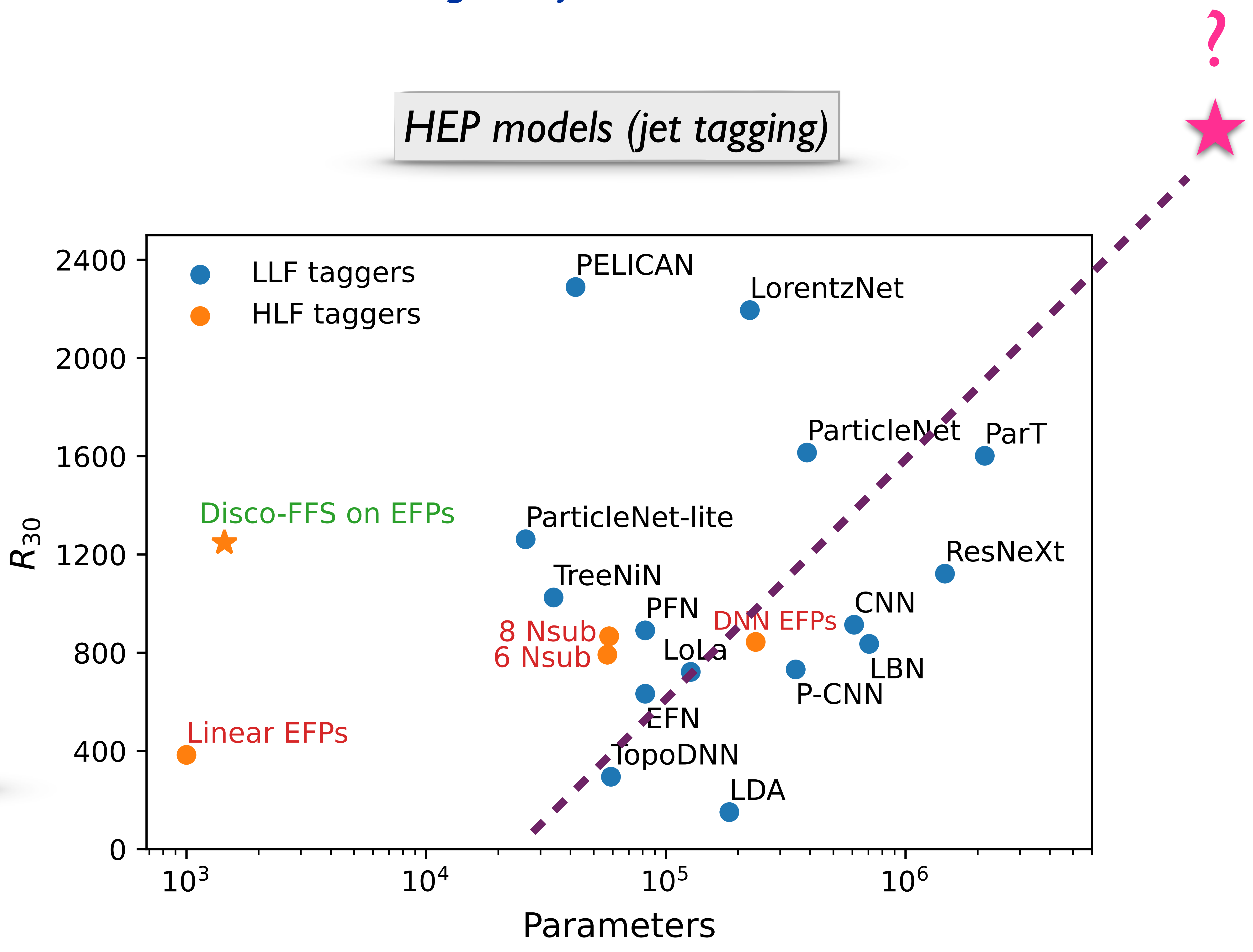
Large Physics Model?

- Large Language Models (like GPT) has transformed NLP. How about a Large Physics Model?

Natural language models



HEP models (jet tagging)



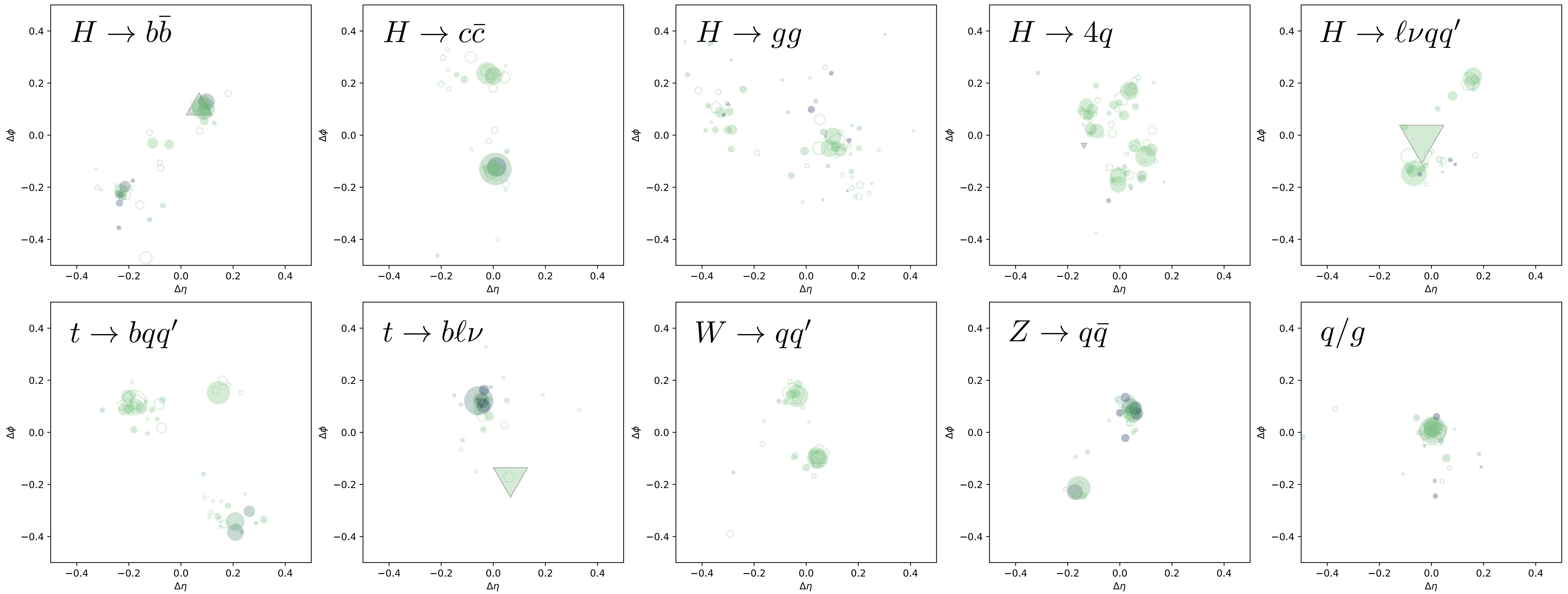
R. Das, G. Kasieczka and D. Shih, arXiv: 2212.00046

GNNs for Particle Physics - July 2023 - Huilin Qu (CERN)

A First Step: New Dataset

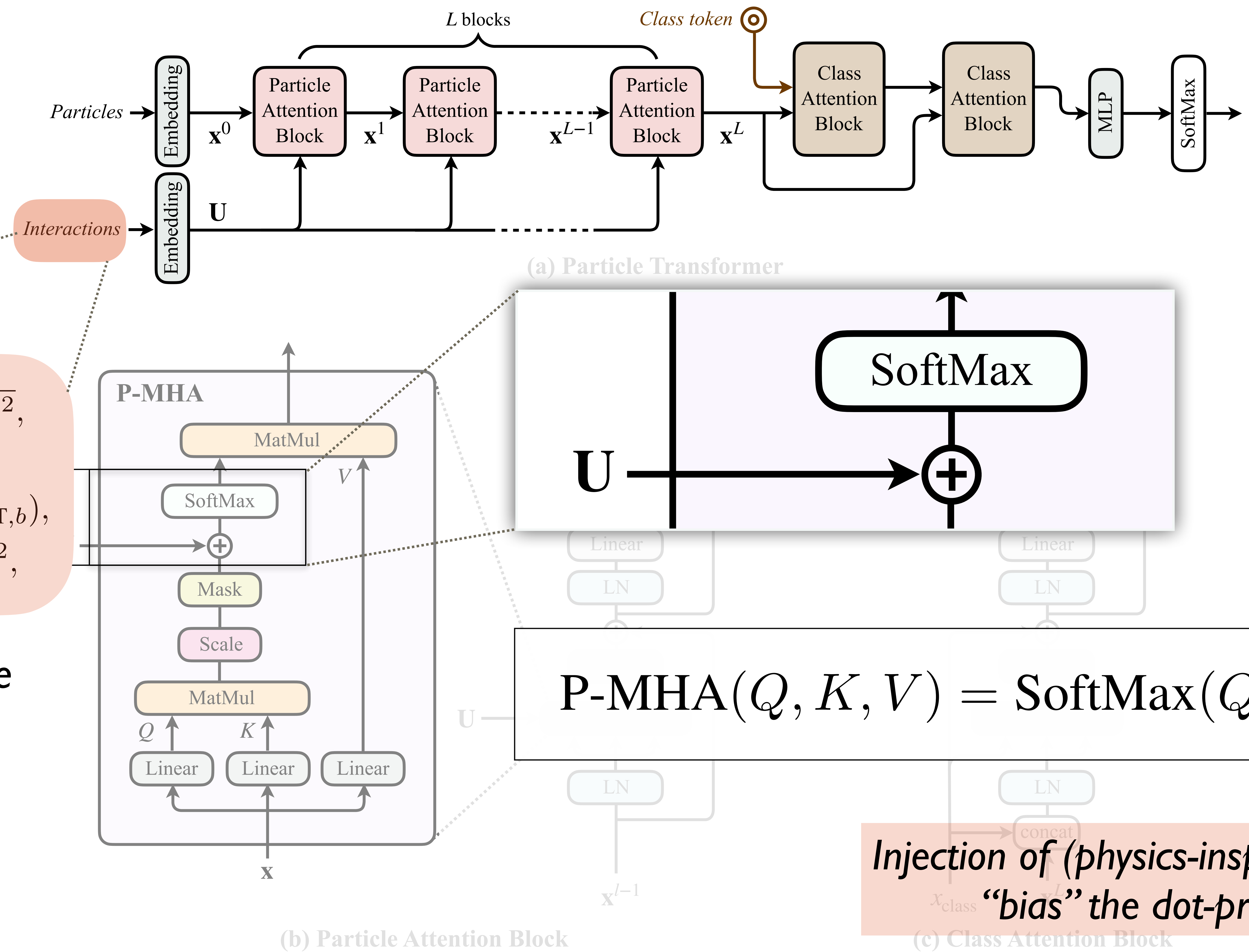
- **JetClass**: a new large and comprehensive jet simulation dataset
  - 100M jets in 10 classes: ~two orders of magnitude larger than existing public datasets

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



Particle Transformer

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



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

and many other possible pairwise features…

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

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

Particle Transformer: Performance

	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$_{50\%}$	$H \to \ell\nu qq'$ Rej$_{99\%}$	$t \to bqq'$ Rej$_{50\%}$	$t \to b\ell\nu$ Rej$_{99.5\%}$	$W \to qq'$ Rej$_{50\%}$	$Z \to q\bar{q}$ Rej$_{50\%}$
PFN	0.772	0.9714	2924	841	75	198	265	797	721	189	159
P-CNN	0.809	0.9789	4890	1276	88	474	947	2907	2304	241	204
ParticleNet	0.844	0.9849	7634	2475	104	954	3339	10526	11173	347	283
ParT	**0.861**	**0.9877**	**10638**	**4149**	**123**	**1864**	**5479**	**32787**	**15873**	**543**	**402**
ParT (plain)	0.849	0.9859	9569	2911	112	1185	3868	17699	12987	384	311

JetClass dataset (100M jets)

- Particle Transformer (ParT): significant performance improvement!
  - compared to the existing state-of-the-art, ParticleNet
    - 1.7% increase in accuracy
    - **up to 3x increase in background rejection** (Rej$_{X\%}$)

$$\text{Rej}_{X\%} \equiv 1/\text{FPR at TPR} = X\%,$$

Particle Transformer: Performance

	All classes		$H \to b\bar{b}$	$H \to c\bar{c}$	$H \to gg$	$H \to 4q$	$H \to \ell\nu qq'$	$t \to bqq'$	$t \to b\ell\nu$	$W \to qq'$	$Z \to q\bar{q}$
	Accuracy	AUC	$\text{Rej}_{50\%}$	$\text{Rej}_{50\%}$	$\text{Rej}_{50\%}$	$\text{Rej}_{50\%}$	$\text{Rej}_{99\%}$	$\text{Rej}_{50\%}$	$\text{Rej}_{99.5\%}$	$\text{Rej}_{50\%}$	$\text{Rej}_{50\%}$
PFN	0.772	0.9714	2924	841	75	198	265	797	721	189	159
P-CNN	0.809	0.9789	4890	1276	88	474	947	2907	2304	241	204
ParticleNet	0.844	0.9849	7634	2475	104	954	3339	10526	11173	347	283
ParT	**0.861**	**0.9877**	**10638**	**4149**	**123**	**1864**	**5479**	**32787**	**15873**	**543**	**402**
ParT (plain)	0.849	0.9859	9569	2911	112	1185	3868	17699	12987	384	311

JetClass dataset (100M jets)

- Particle Transformer (ParT): significant performance improvement!
  - compared to the existing state-of-the-art, ParticleNet
    - 1.7% increase in accuracy
    - **up to 3x increase in background rejection** ($\text{Rej}_{X\%}$)
- ParT (plain): plain Transformer w/o interaction features
  - 1.2% drop in accuracy compared to full ParT
  - **Physics-driven modification of self-attention plays a key role!**

Model complexity

	Accuracy	# params	FLOPs
PFN	0.772	86.1 k	4.62 M
P-CNN	0.809	354 k	15.5 M
ParticleNet	0.844	370 k	540 M
ParT	**0.861**	2.14 M	340 M
ParT (plain)	0.849	2.13 M	260 M

Particle Transformer: Pre-Training + Fine-Tuning

- The large Transformer-based model enables new training paradigm
  - (supervised) pre-training on a large dataset (e.g., JetClass) & fine-tuning to downstream tasks
  - significantly outperforms existing models

Top quark tagging benchmark (~2M jets) [SciPost Phys. 7 (2019) 014]

	Accuracy	AUC	$\mathrm{Rej}_{50\%}$	$\mathrm{Rej}_{30\%}$
P-CNN	0.930	0.9803	201 ± 4	759 ± 24
PFN	—	0.9819	247 ± 3	888 ± 17
ParticleNet	0.940	0.9858	397 ± 7	1615 ± 93
JEDI-net (w/ $\sum O$)	0.930	0.9807	—	774.6
PCT	0.940	0.9855	392 ± 7	1533 ± 101
LGN	0.929	0.964	—	435 ± 95
rPCN	—	0.9845	364 ± 9	1642 ± 93
LorentzNet	0.942	0.9868	498 ± 18	2195 ± 173
ParT	0.940	0.9858	413 ± 16	1602 ± 81
ParticleNet-f.t.	0.942	0.9866	487 ± 9	1771 ± 80
ParT-f.t.	**0.944**	**0.9877**	$\mathbf{691 \pm 15}$	$\mathbf{2766 \pm 130}$

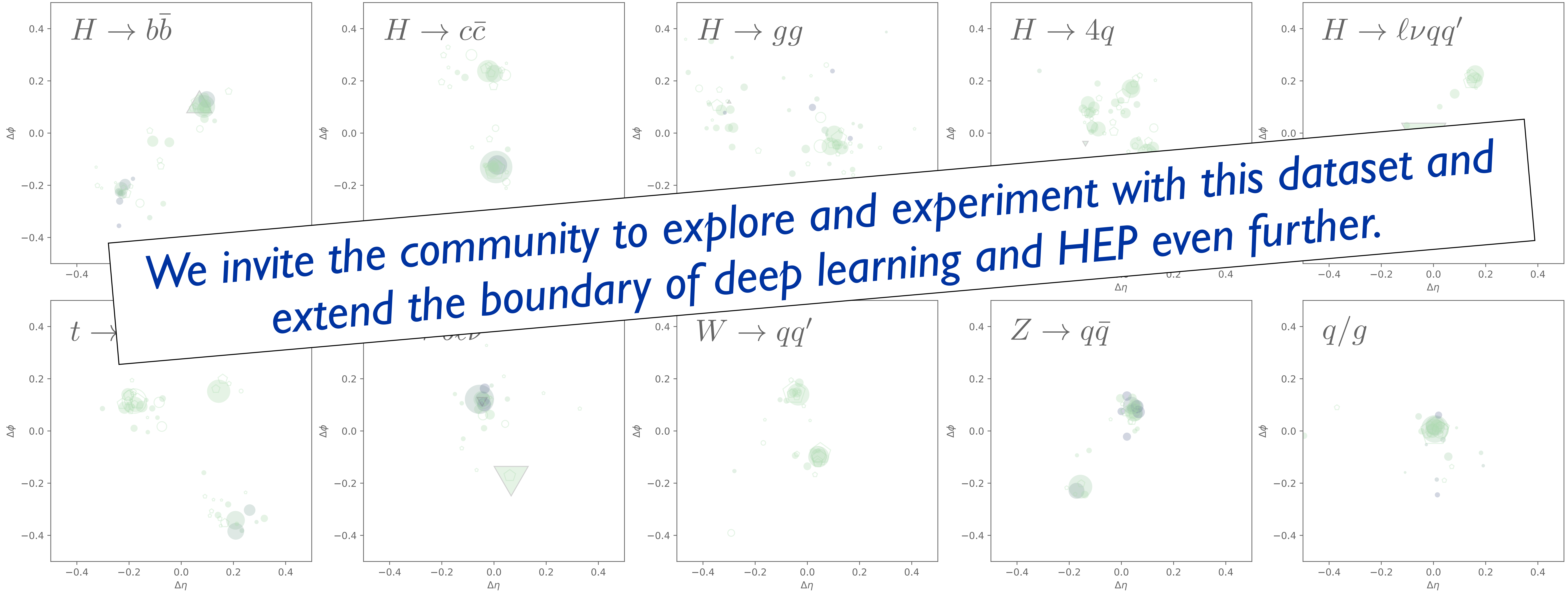
Quark-gluon tagging benchmark (~2M jets) [JHEP 01 (2019) 121]

	Accuracy	AUC	$\mathrm{Rej}_{50\%}$	$\mathrm{Rej}_{30\%}$
$\text{P-CNN}_{\text{exp}}$	0.827	0.9002	34.7	91.0
PFN_{exp}	—	0.9005	34.7 ± 0.4	—
$\text{ParticleNet}_{\text{exp}}$	0.840	0.9116	39.8 ± 0.2	98.6 ± 1.3
rPCN_{exp}	—	0.9081	38.6 ± 0.5	—
ParT_{exp}	0.840	0.9121	41.3 ± 0.3	101.2 ± 1.1
$\text{ParticleNet-f.t.}_{\text{exp}}$	0.839	0.9115	40.1 ± 0.2	100.3 ± 1.0
$\mathbf{ParT\text{-}f.t._{exp}}$	**0.843**	**0.9151**	$\mathbf{42.4 \pm 0.2}$	$\mathbf{107.9 \pm 0.5}$
PFN_{full}	—	0.9052	37.4 ± 0.7	—
$\text{ABCNet}_{\text{full}}$	0.840	0.9126	42.6 ± 0.4	118.4 ± 1.5
PCT_{full}	0.841	0.9140	43.2 ± 0.7	118.0 ± 2.2
$\text{LorentzNet}_{\text{full}}$	0.844	0.9156	42.4 ± 0.4	110.2 ± 1.3
$\text{ParT}_{\text{full}}$	0.849	0.9203	47.9 ± 0.5	129.5 ± 0.9
$\mathbf{ParT\text{-}f.t._{full}}$	**0.852**	**0.9230**	$\mathbf{50.6 \pm 0.2}$	$\mathbf{138.7 \pm 1.3}$

GOING BEYOND?

- **JETCLASS**: a new large and comprehensive jet simulation dataset
  - 100M jets in 10 classes: ~two orders of magnitude larger than existing public datasets

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



We invite the community to explore and experiment with this dataset and extend the boundary of deep learning and HEP even further.

The Road Ahead

- Can we better incorporate physics knowledge into the network design?
  - physics aware data representation, symmetry group equivariant architecture, ...
- Can we scale up to a large model for HEP?
  - large datasets, pre-training, multi-modal learning, ...
- Can we improve the computational efficiency of GNNs?
  - emerging specialized libraries for GNN training and inference (PyG, DGL, TF-GNN, ...)
  - accelerated inference on specialized ASICs / FPGAs (e.g., for triggering), software hardware co-design, ...
- Can we improve the robustness of GNNs (e.g., data/simulation difference)?
  - domain adaption? calibration? uncertainty aware training? ...
- Can we improve the interpretability and explainability of GNNs?

The Road Ahead

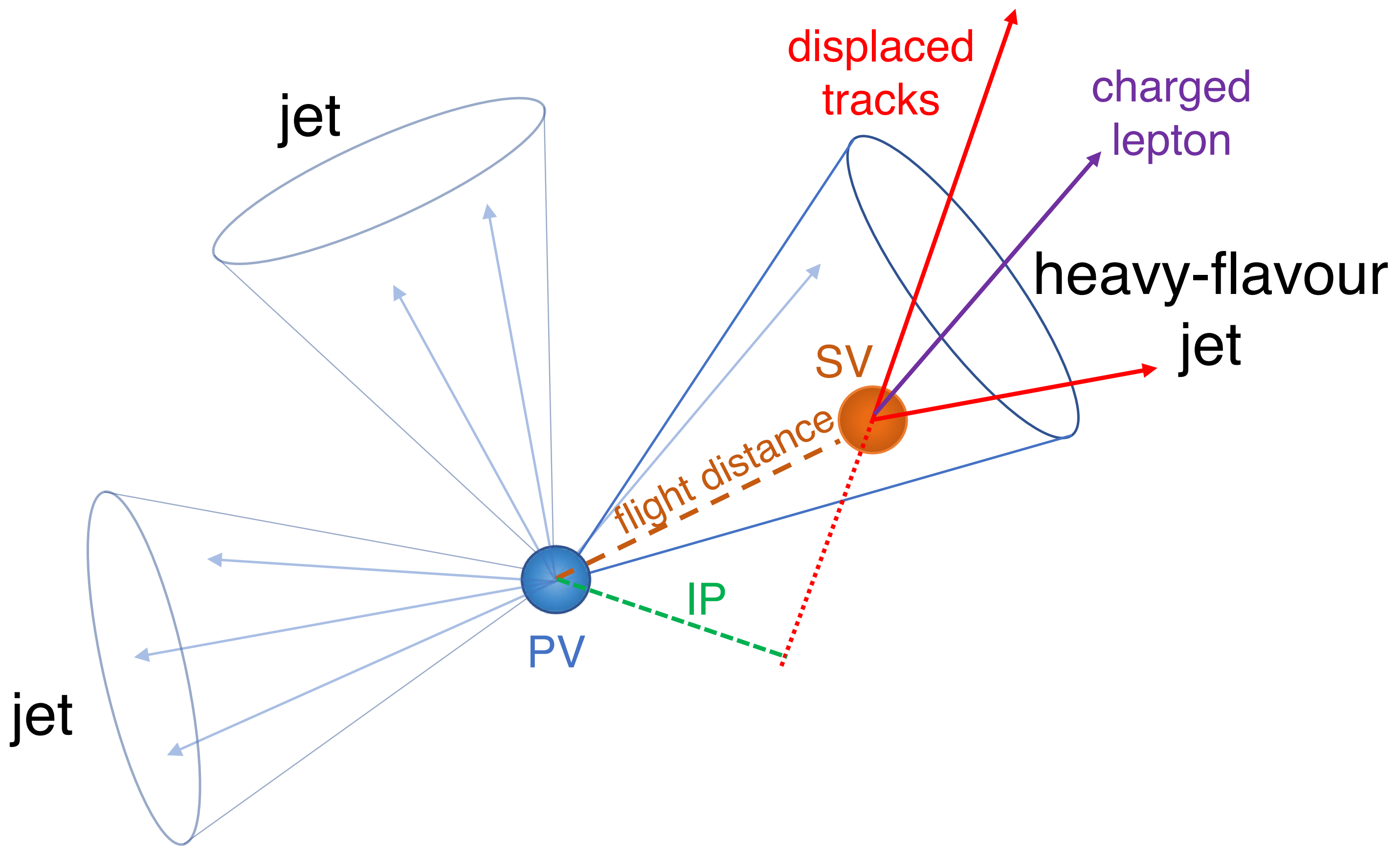
- Can we better incorporate physics knowledge into the network design?
  - physics aware data representation, symmetry group equivariant architecture, …
- Can we scale up to a large model for HEP?
  - large datasets, pre-training, multi-modal learning, …
- Can we impro
  - emerging sp … for GNN training and inference (PyG, DGL, TF-GNN, …)
  - accelerated inference on specialized ASICs / FPGAs (e.g., for triggering), software hardware co-design, …
- Can we improve the robustness of GNNs (e.g., data/simulation difference)?
  - domain adaption? calibration? uncertainty aware training? …
- Can we improve the interpretability and explainability of GNNs?

Your innovation and creativity can make a big difference!

EXTRA: PARTICAL JET TAGGING (IN CMS)

Boosted Jet Tagging

- Hadronic decays of highly Lorentz-boosted heavy particles (Higgs/W/Z/top) lead to large-radius jets with distinctive characteristics:
  - different radiation patterns ("**substructure**")
    - 3-prong (top), 2-prong (W/Z/H) vs 1-prong (gluon/light quark jet)
  - different **flavor** content: existence of one or more b-/c-quarks
- Boosted jet tagging:
  - simultaneously exploiting both **substructure** and **flavor** to maximize the performance
  - significant performance leap thanks to deep learning techniques



DeepAK8

- Advanced deep learning-based algorithm for boosted jet tagging, using AK8 (anti-k_T R=0.8) jets
  - **multi-class classifier** for top quark and W, Z, Higgs boson tagging
  - **directly uses jet constituents** (particle-flow candidates / secondary vertices)
  - **1D convolutional neural network (CNN)**, based on the ResNet [arXiv: 1512.03385] architecture

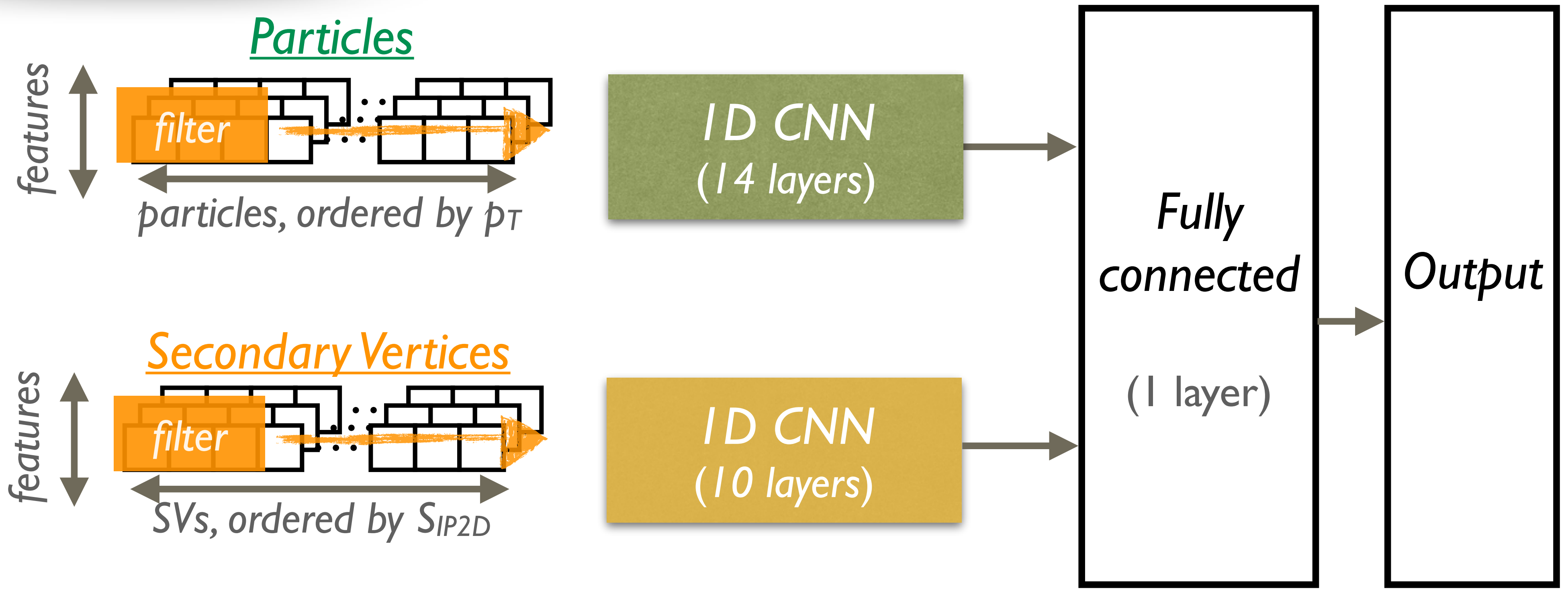
Inputs

Substructure — *Particles*
- Up to 100 PF candidates(*)
- Sorted in descending p_T order
- Uses basic kinematic variables, Puppi weights, and track properties (quality, covariance, displacement, etc.)

Flavour — *Secondary vertices*
- Up to 7 SVs(*) (inside jet cone)
- Sorted in descending S_{IP2D} order
- Uses SV kinematics and properties (quality, displacement, etc.)

(*) Number chosen to include all candidates for ≥ 90% of the events

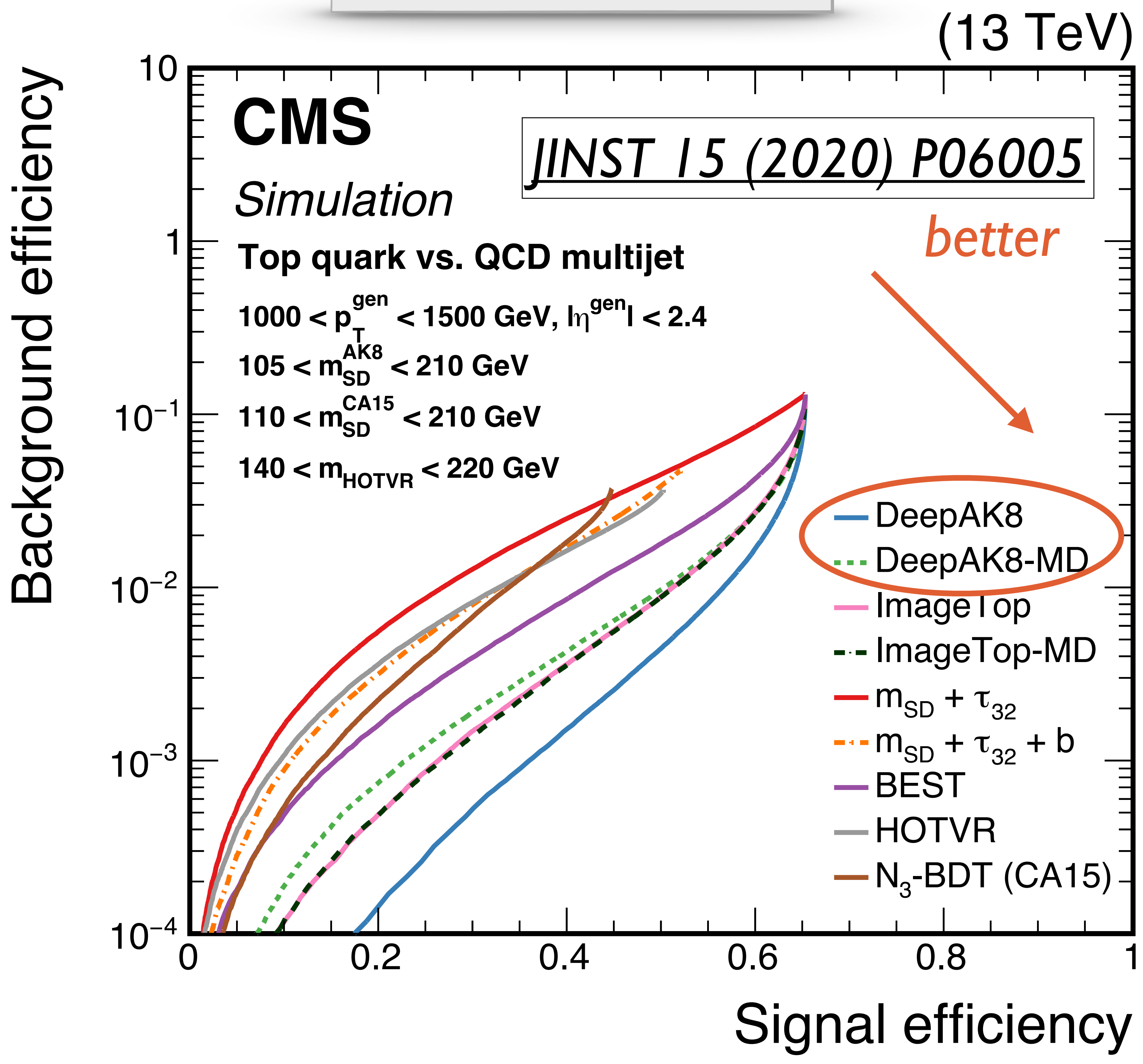
Architecture



Output

Category	Label
Higgs	H (bb)
	H (cc)
	H (VV*→qqqq)
Top	top (bcq)
	top (bqq)
	top (bc)
	top (bq)
W	W (cq)
	W (qq)
Z	Z (bb)
	Z (cc)
	Z (qq)
QCD	QCD (bb)
	QCD (cc)
	QCD (b)
	QCD (c)
	QCD (others)

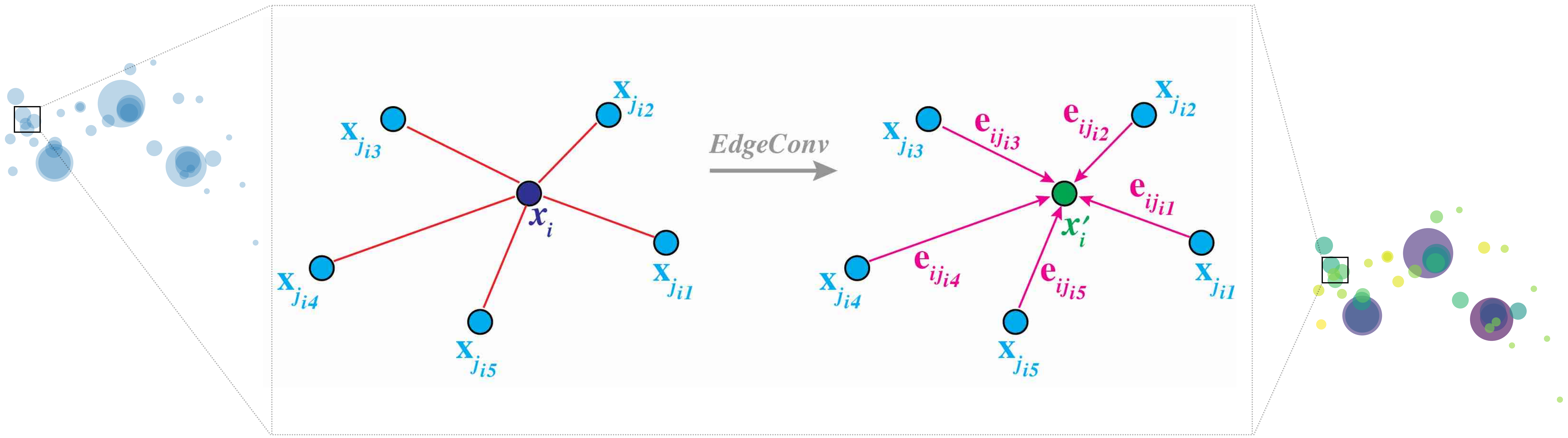
Top quark tagging



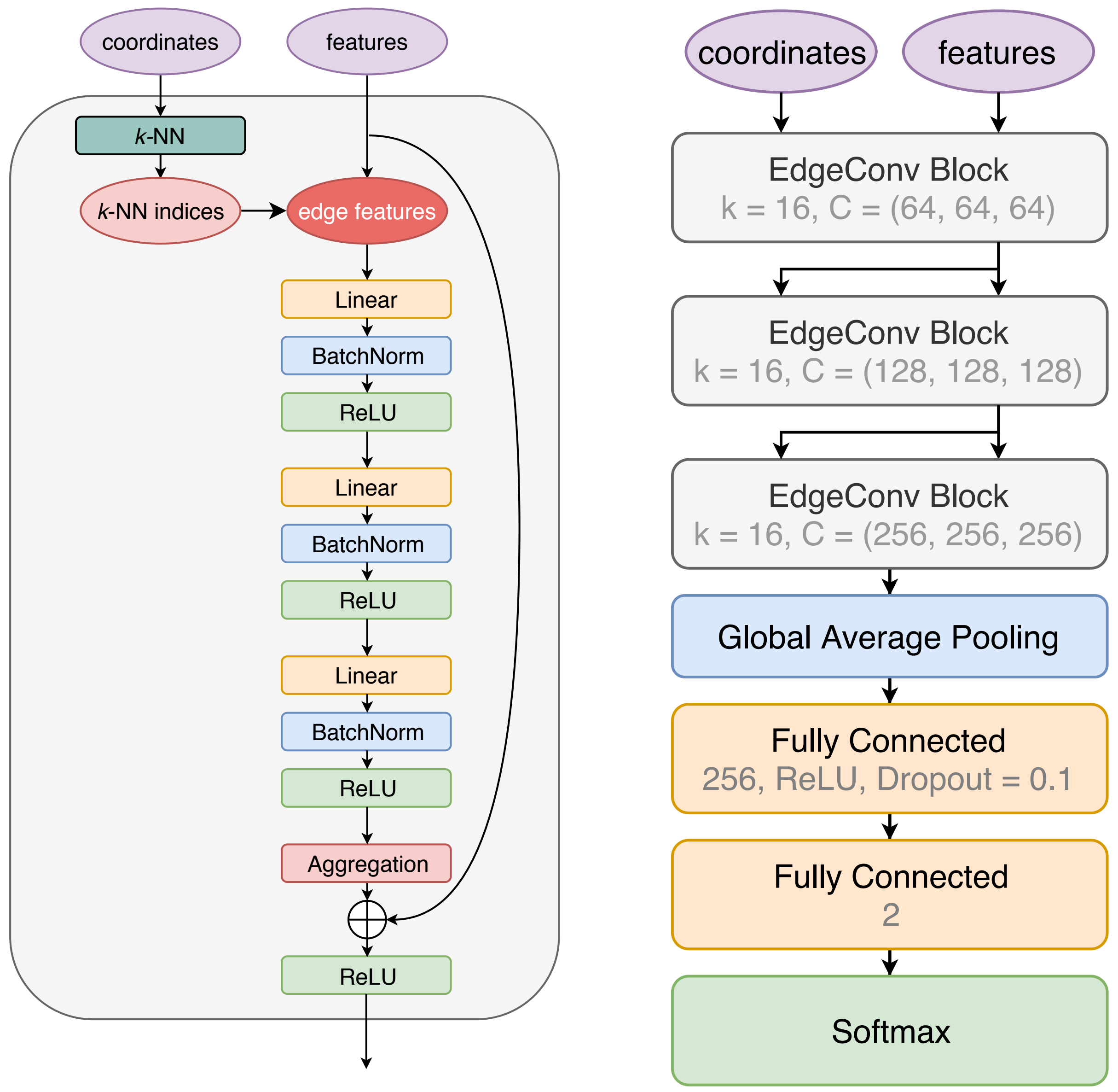
ParticleNet

HQ and L. Gouskos
Phys.Rev.D 101 (2020) 5, 056019

- 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: $e_{ij} = \mathrm{MLP}(x_i, x_j)$
      - aggregate the edge features in a symmetric way: $x_i' = \mathrm{mean}_j\, e_{ij}$



ParticleNet architecture

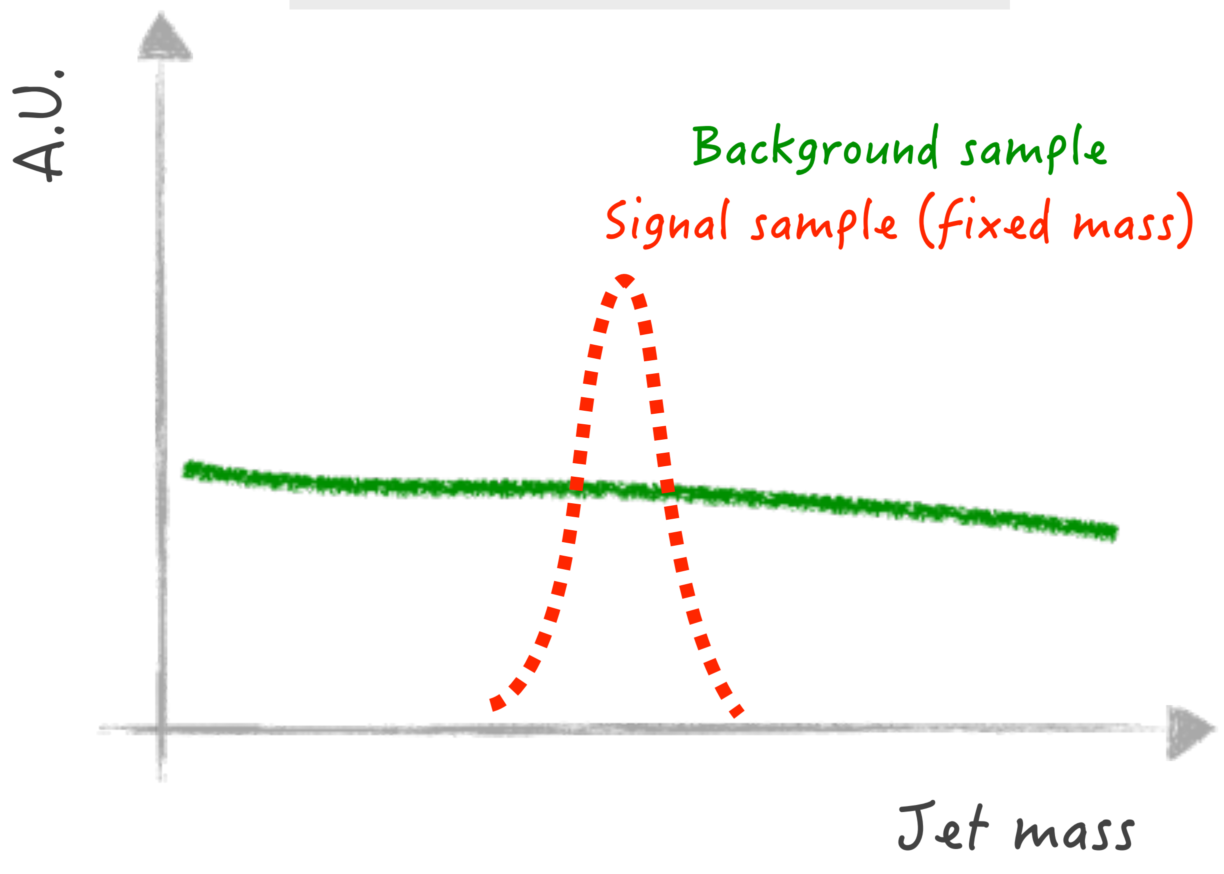


cf. P.T. Komiske, E. M. Metodiev and J. Thaler, JHEP 01 (2019) 121;
V. Mikuni and F. Canelli, Eur. Phys. J. Plus 135, 463 (2020); Mach.Learn.Sci.Tech. 2 (2021) 3, 035027.

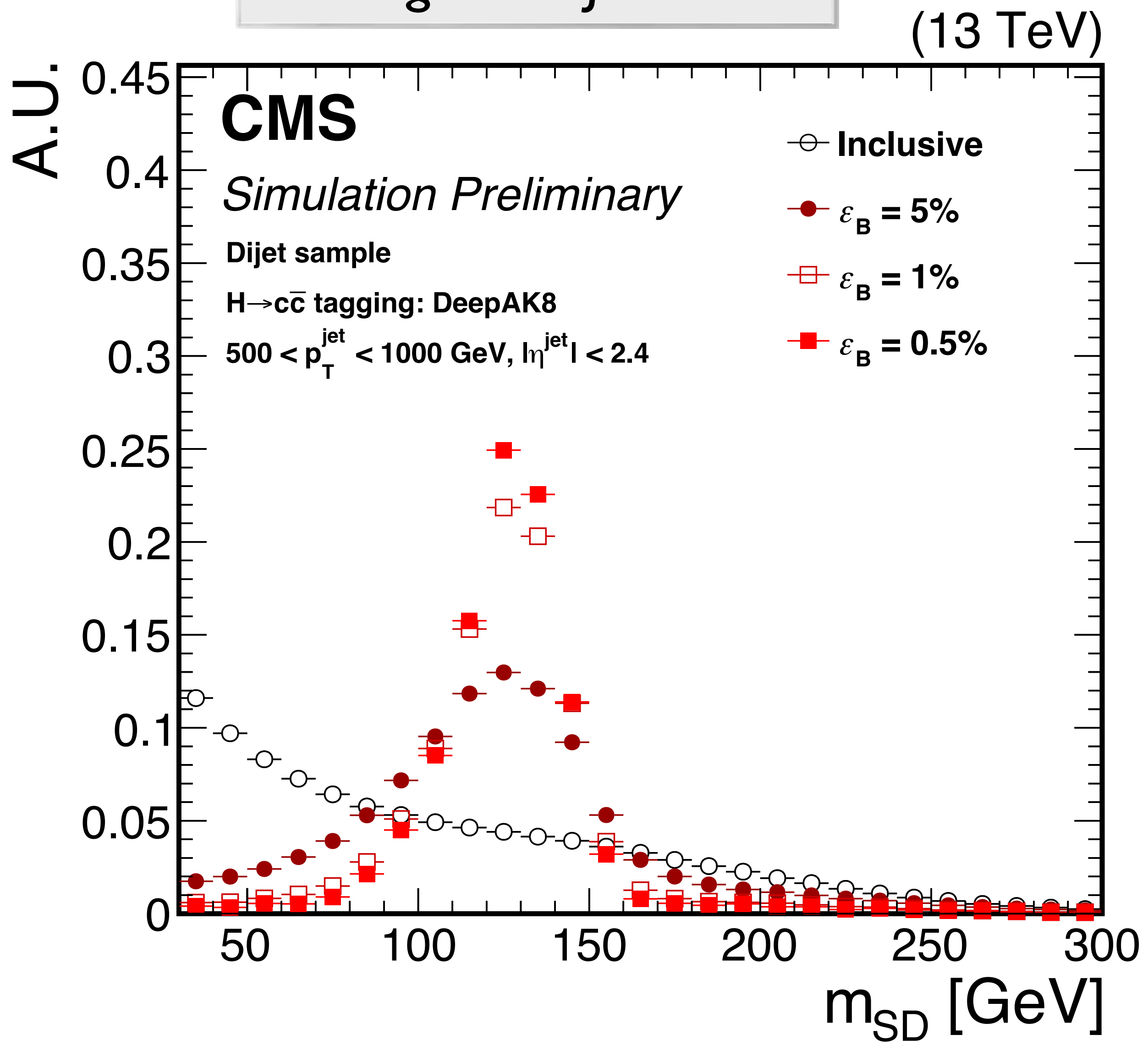
Correlation with the Jet Mass

CMS DP-2020/002

Plain training: no mass decorrelation



Background jet mass

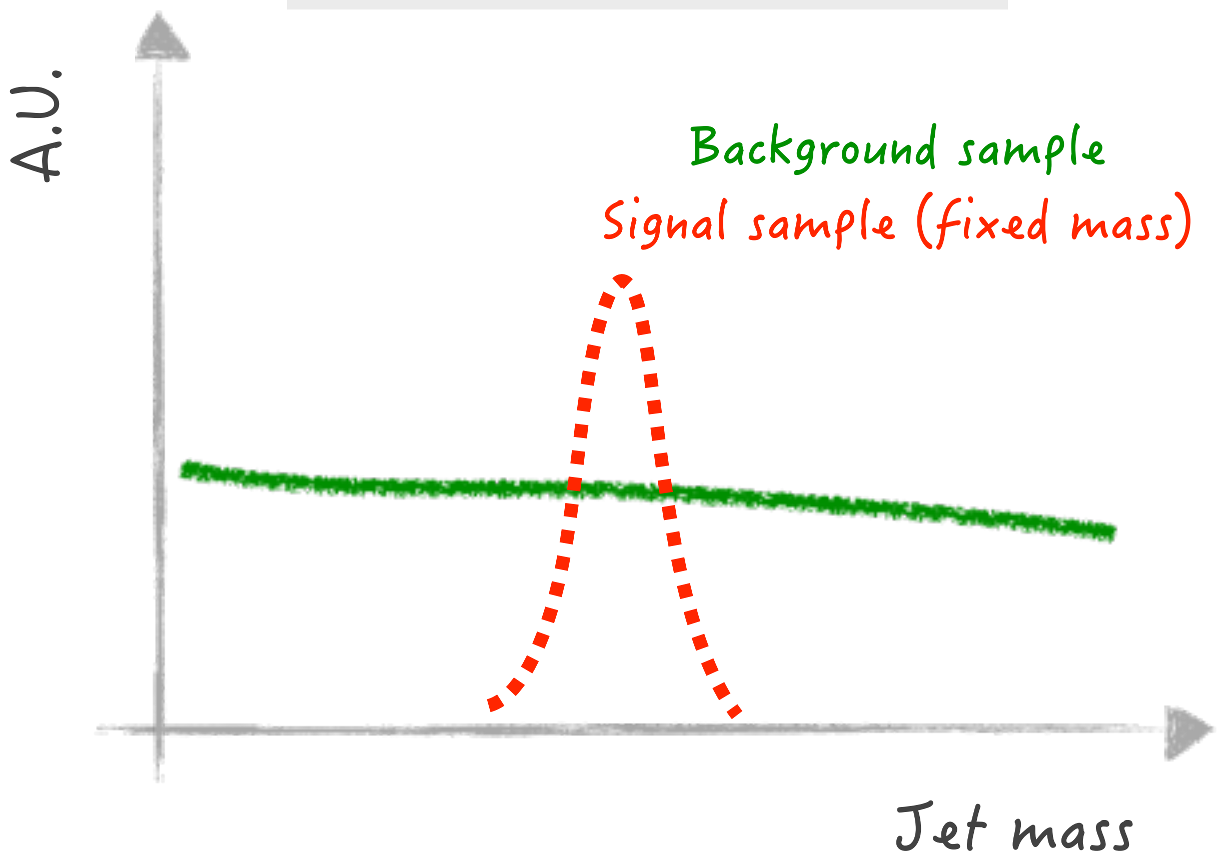


- One feature of these taggers is the correlation with the jet mass
  - jet mass shape of the background becomes similar to that of the signal after selection with the tagger: **"mass sculpting"**
  - not necessarily a problem, but a mass-independent tagger is often more desirable:
    - allows to use the mass variable to further separate signal and background
    - enables tagging signal jets with an unknown mass
    - ...

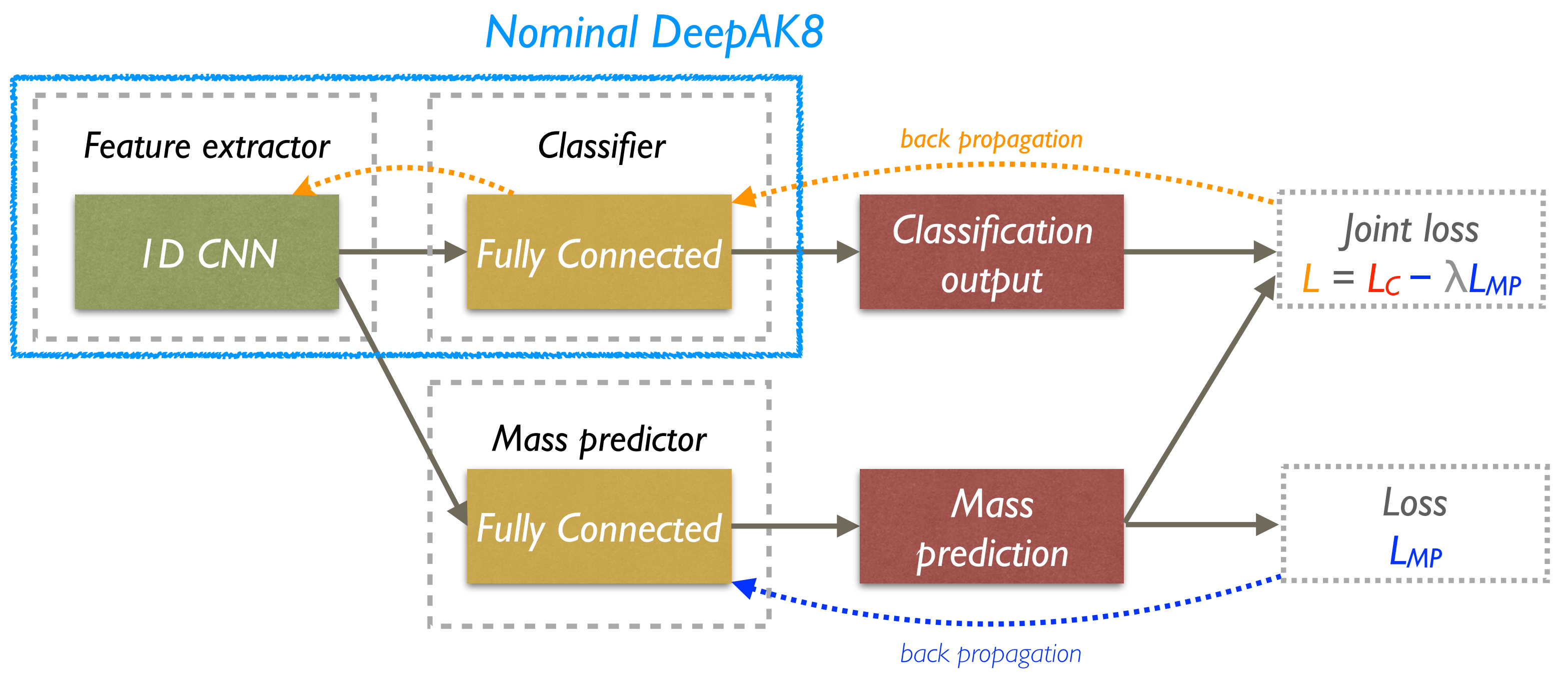
Decorrelation with the Jet Mass

CMS DP-2020/002

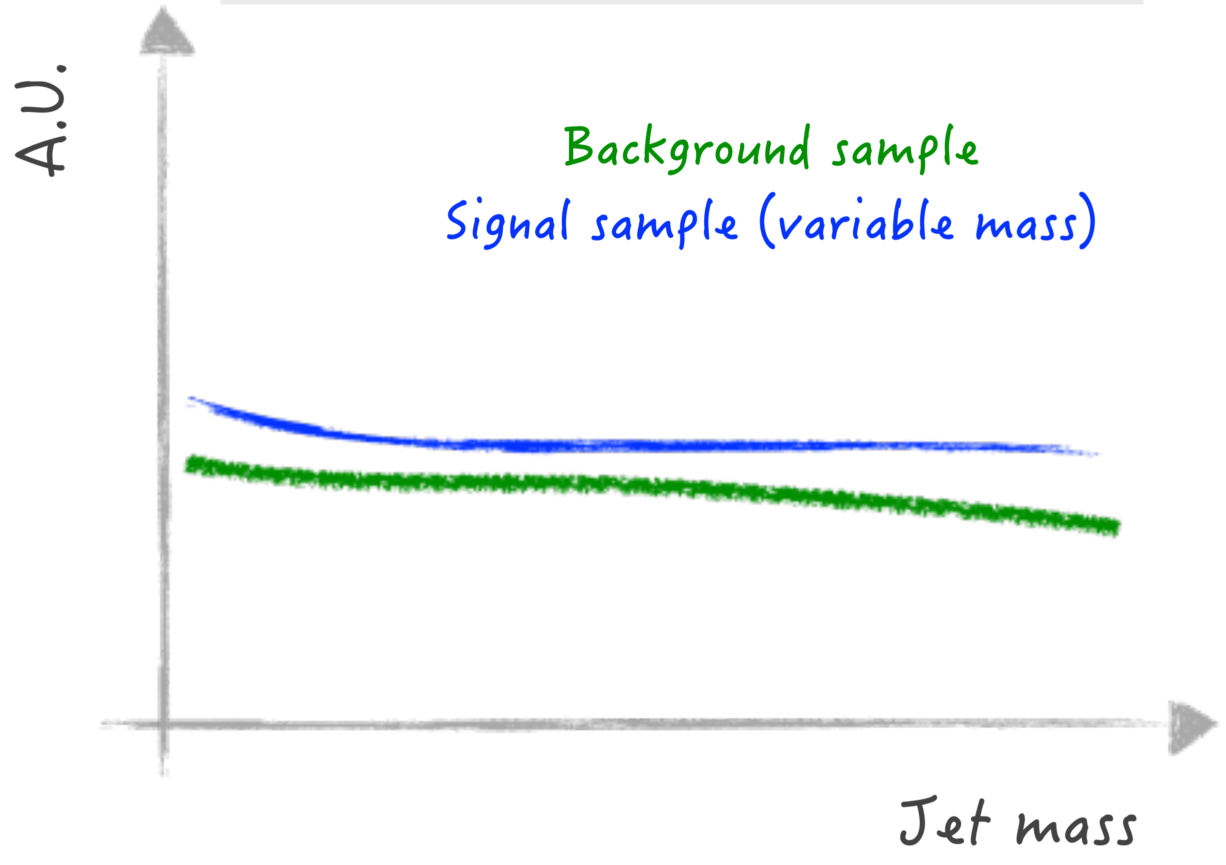
Plain training:
no mass decorrelation



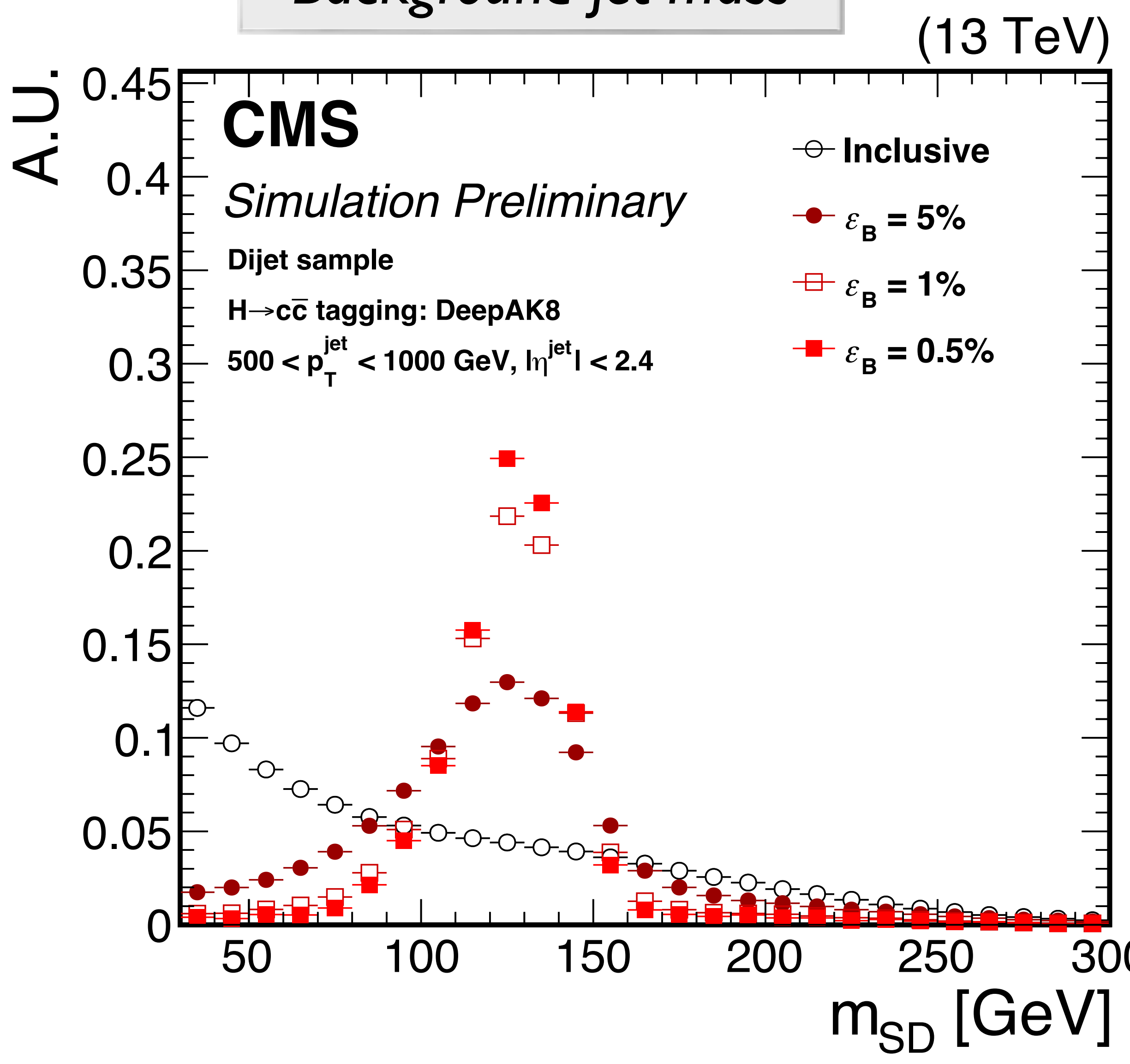
Mass-decorrelated DeepAK8:
"adversarial training"



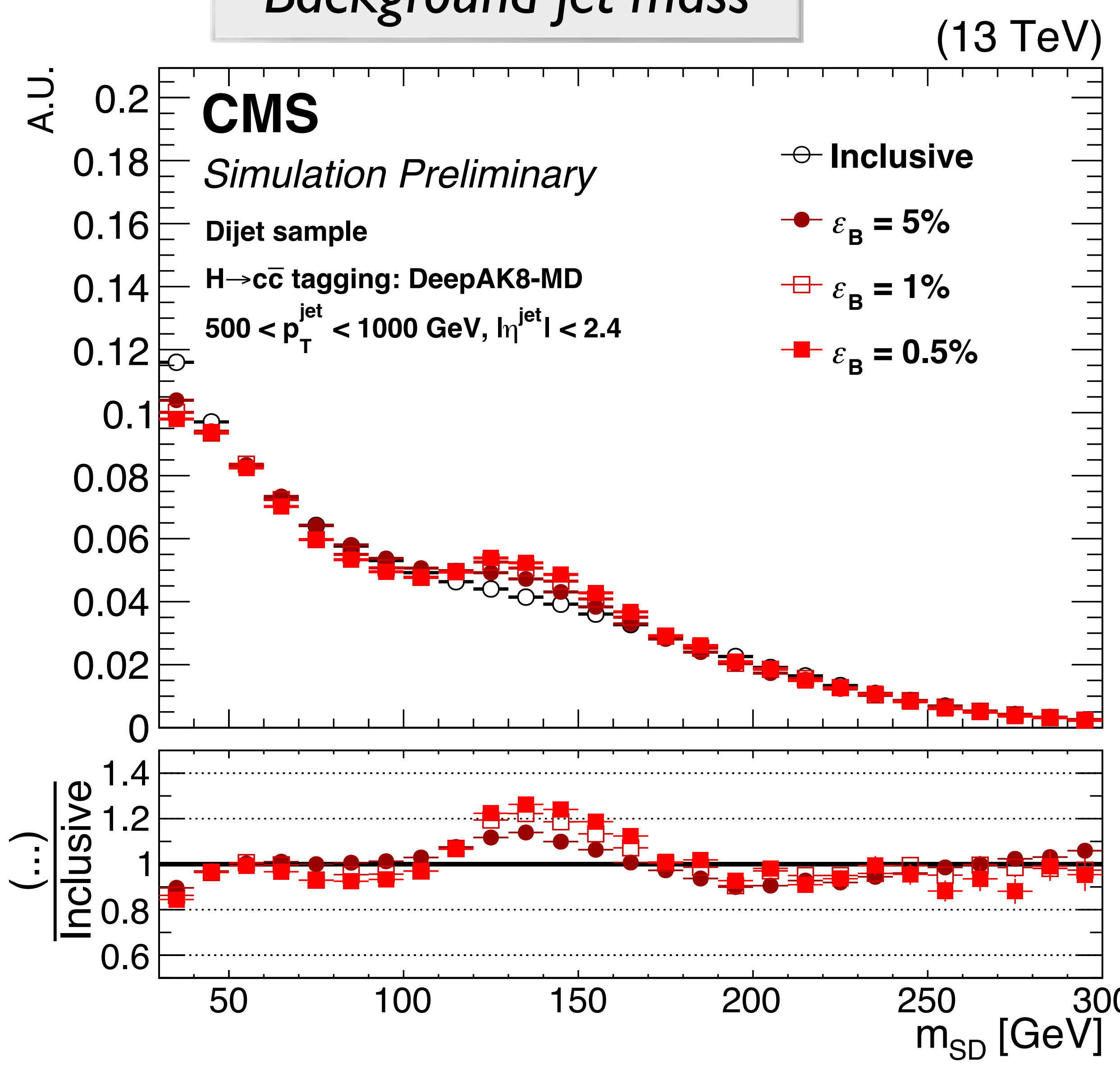
Mass-decorrelated ParticleNet:
training with variable-mass signal



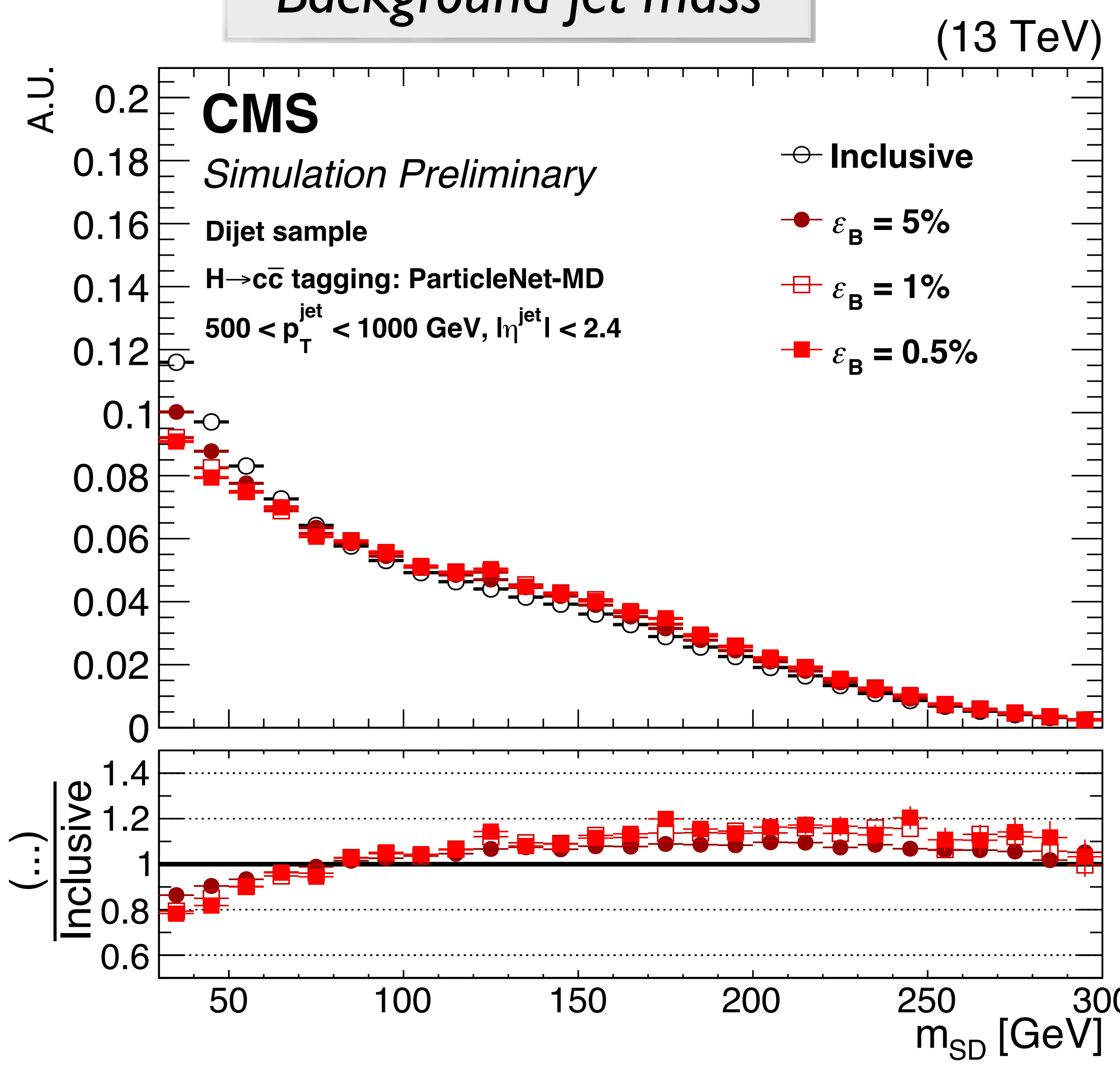
Background jet mass



Background jet mass

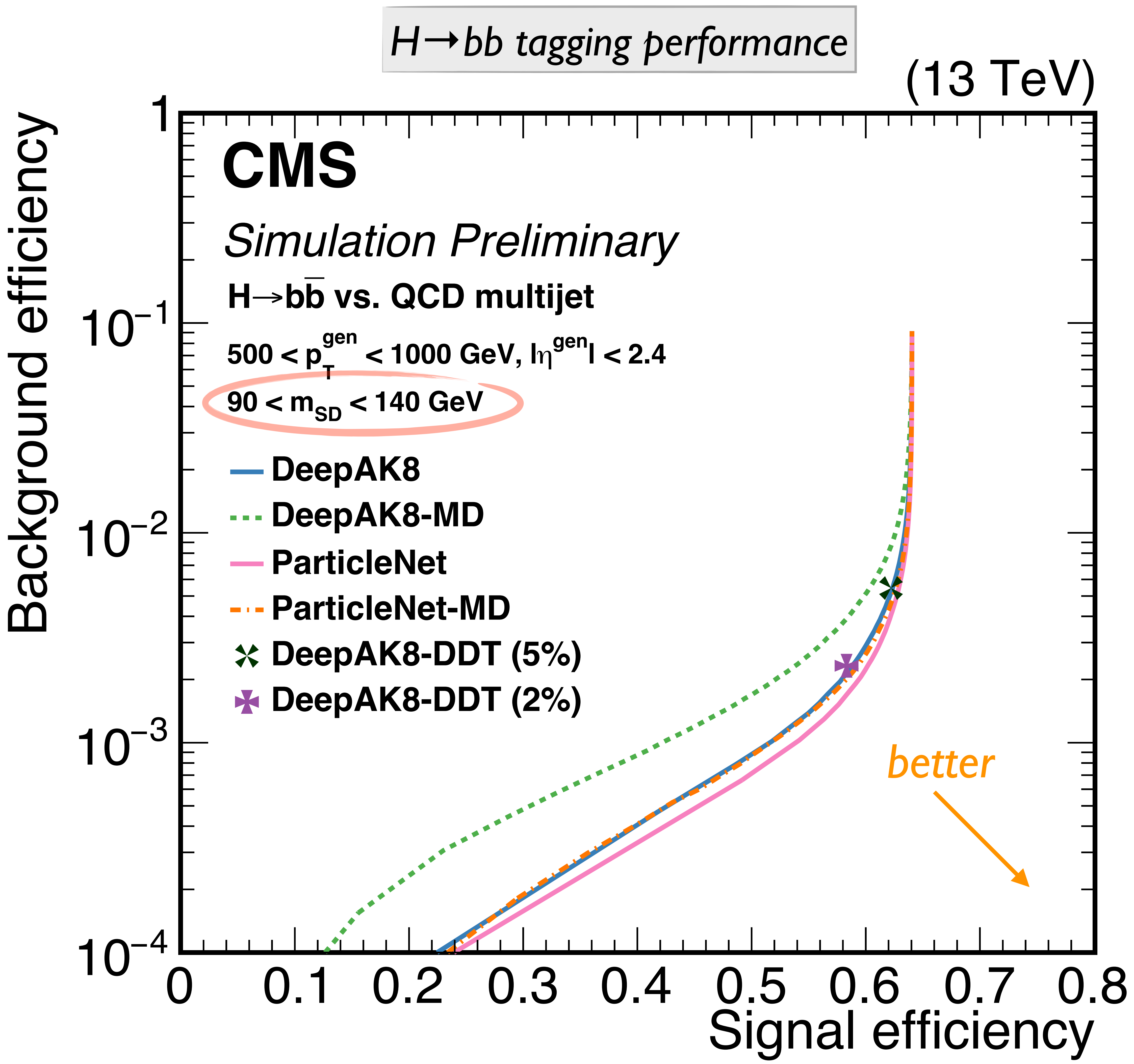


Background jet mass



Performance Comparison

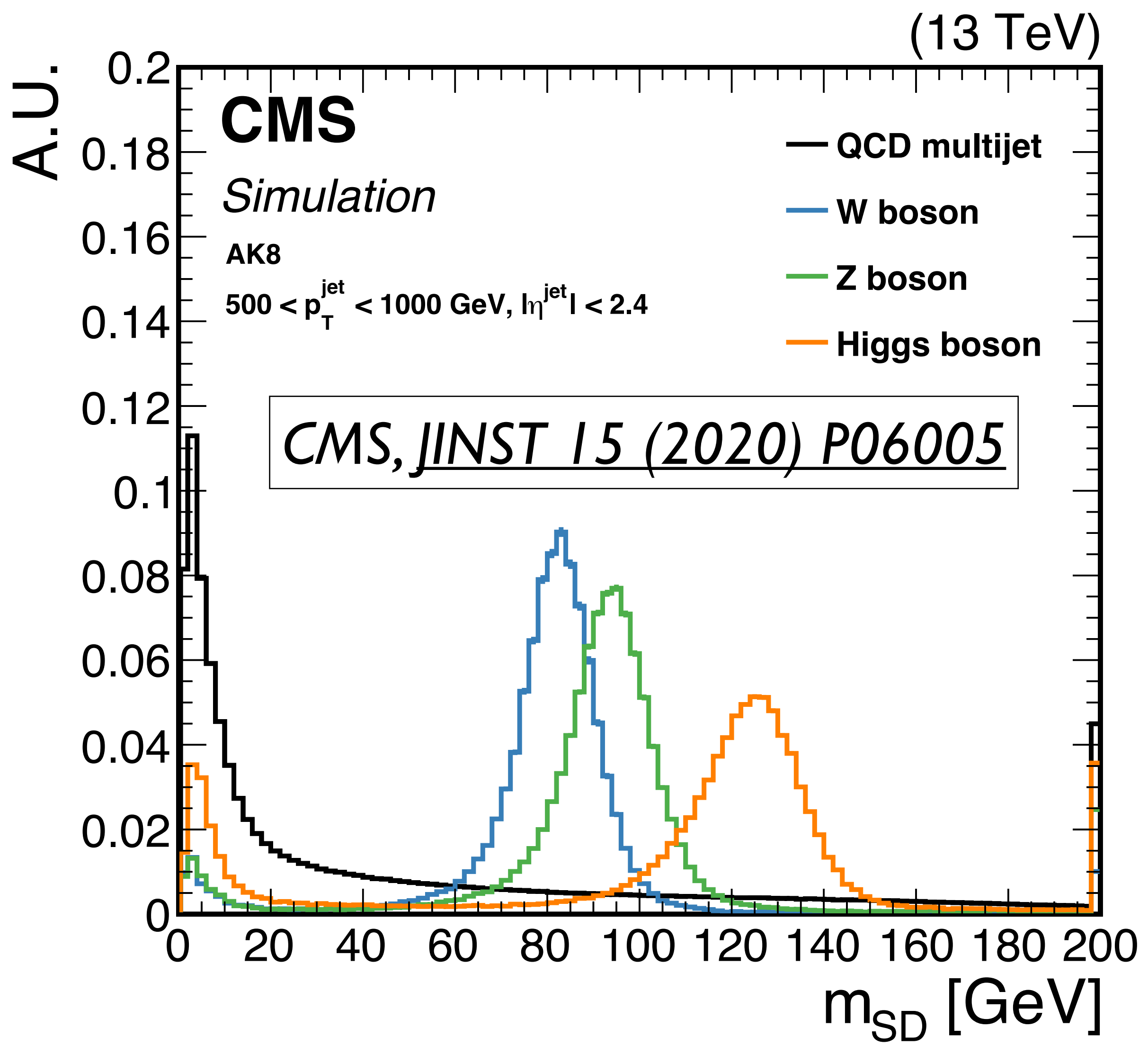
CMS DP-2020/002



- ParticleNet-MD
  - using a special signal sample for training
    - hadronic decays of a spin-0 particle X
      - X → bb, X → cc, X → qq
    - not a fixed mass, but a flat mass spectrum
      - m(X) ∈ [15, 250] GeV
    - allows to easily reweight both signal and background to a ~flat 2D distribution in (p_T, mass) for the training
- ParticleNet-MD shows the best performance
  - ~3-4x better background rejection compared to DeepAK8-MD (based on "adversarial training")
  - only slight performance loss compared to the nominal version w/o mass decorrelation

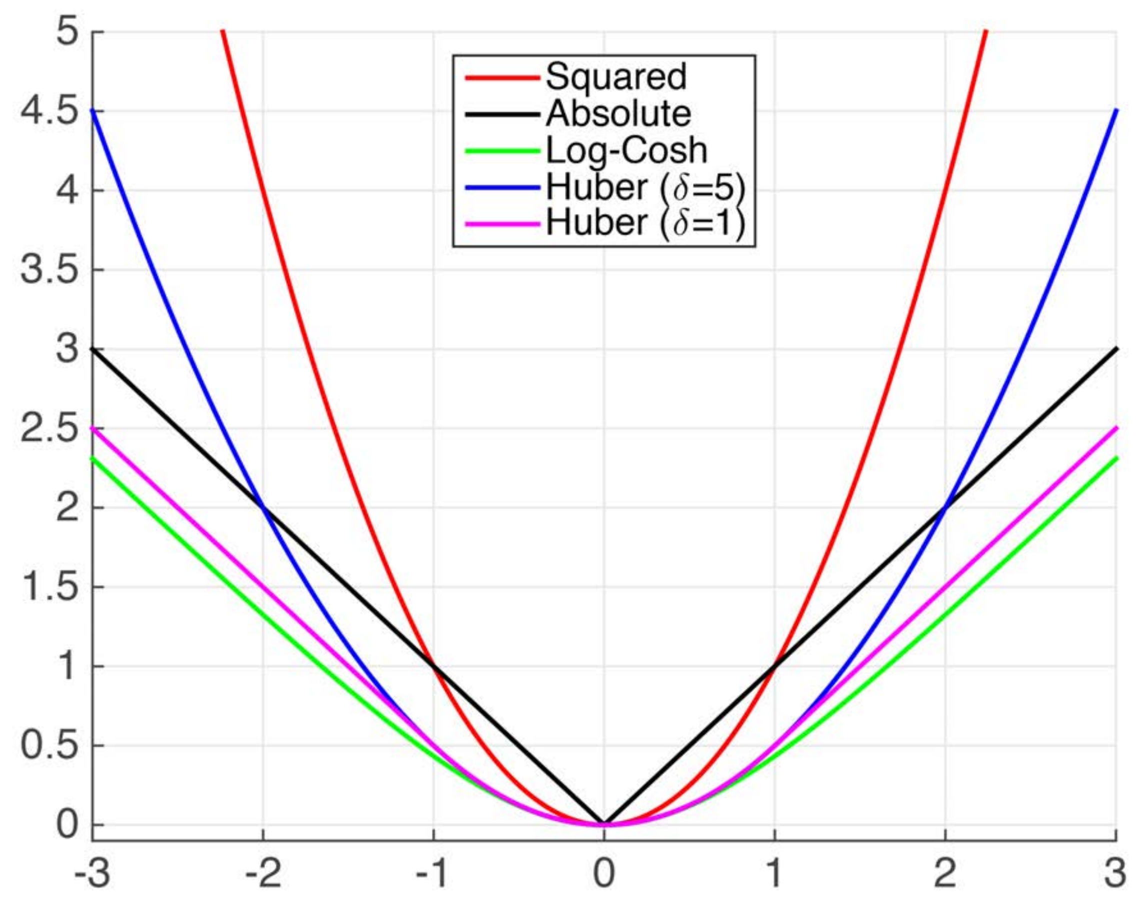
Mass Regression

- Jet mass: one of the most powerful observables for boosted jet tagging
  - characteristic mass peak for top/W/Z/H jets v.s. continuum for QCD jets



- Mass regression:
  - exploit deep learning to reconstruct jet mass with the highest possible resolution
  - training setup similar to the ParticleNet tagger
    - but: predict the jet mass directly from the jet consitituents

- Regression target:
  - signal (X → bb/cc/qq): generated particle mass of X [flat spectrum in 15 – 250 GeV]
  - background (QCD) jets: soft drop mass of the generated particle-level jet

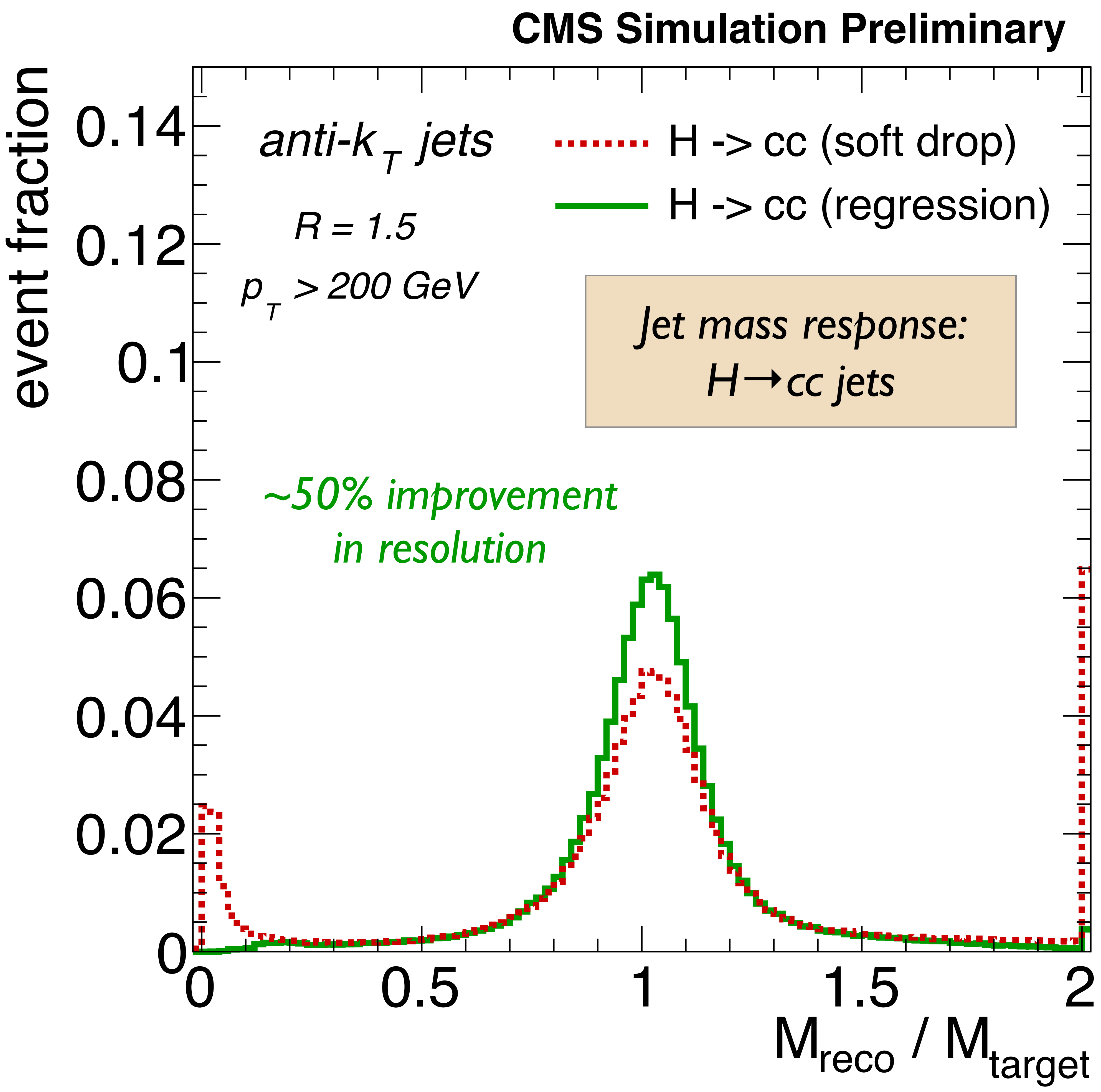


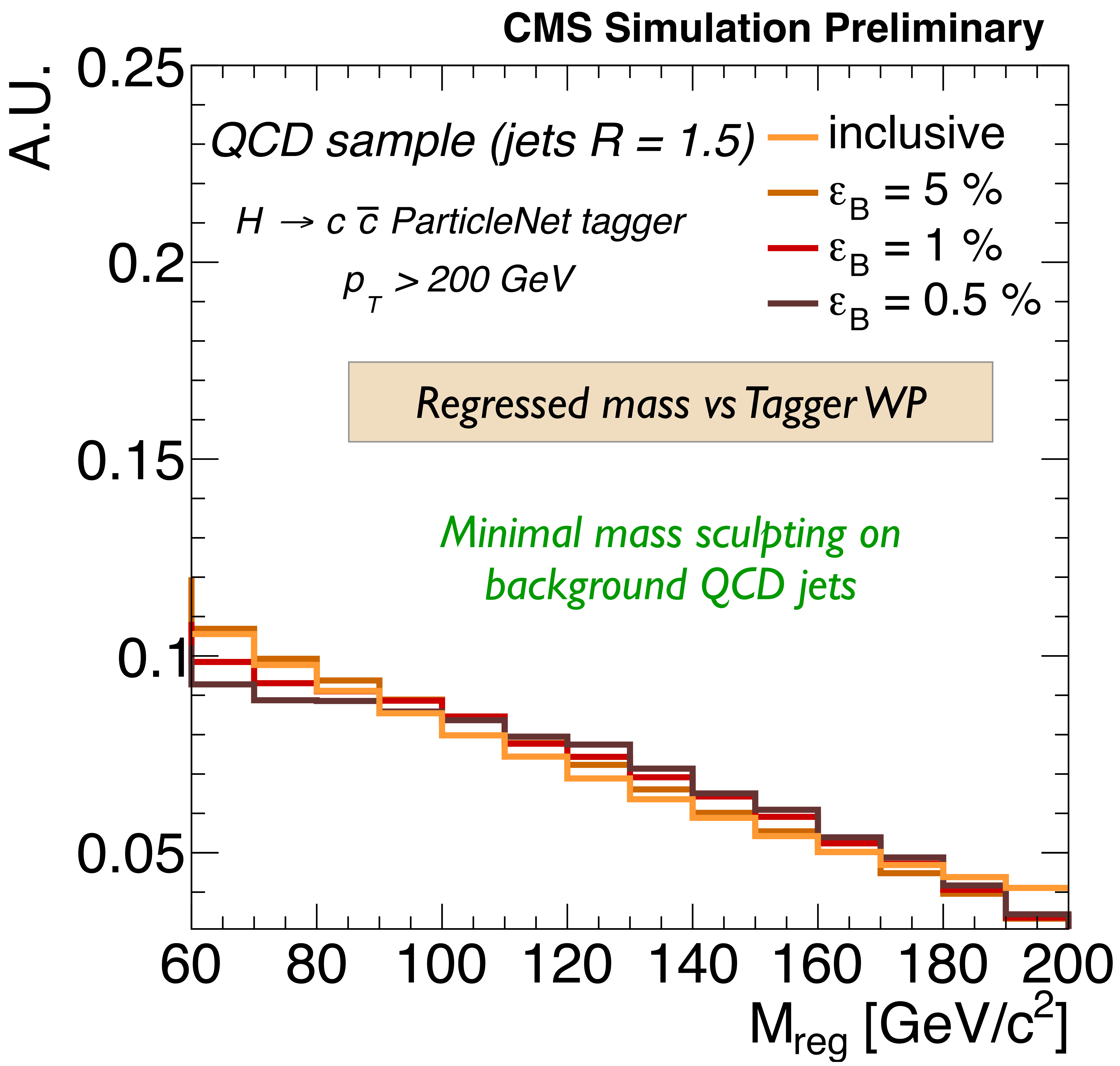
https://www.cs.cornell.edu/courses/cs4780/2015fa/web/lecturenotes/lecturenote10.html

- Loss function
  - LogCosh: $$L(y, y^p) = \sum_{i=1}^{n} \log(\cosh(y_i^p - y_i))$$

Mass Regression: Performance

CMS DP-2021/017

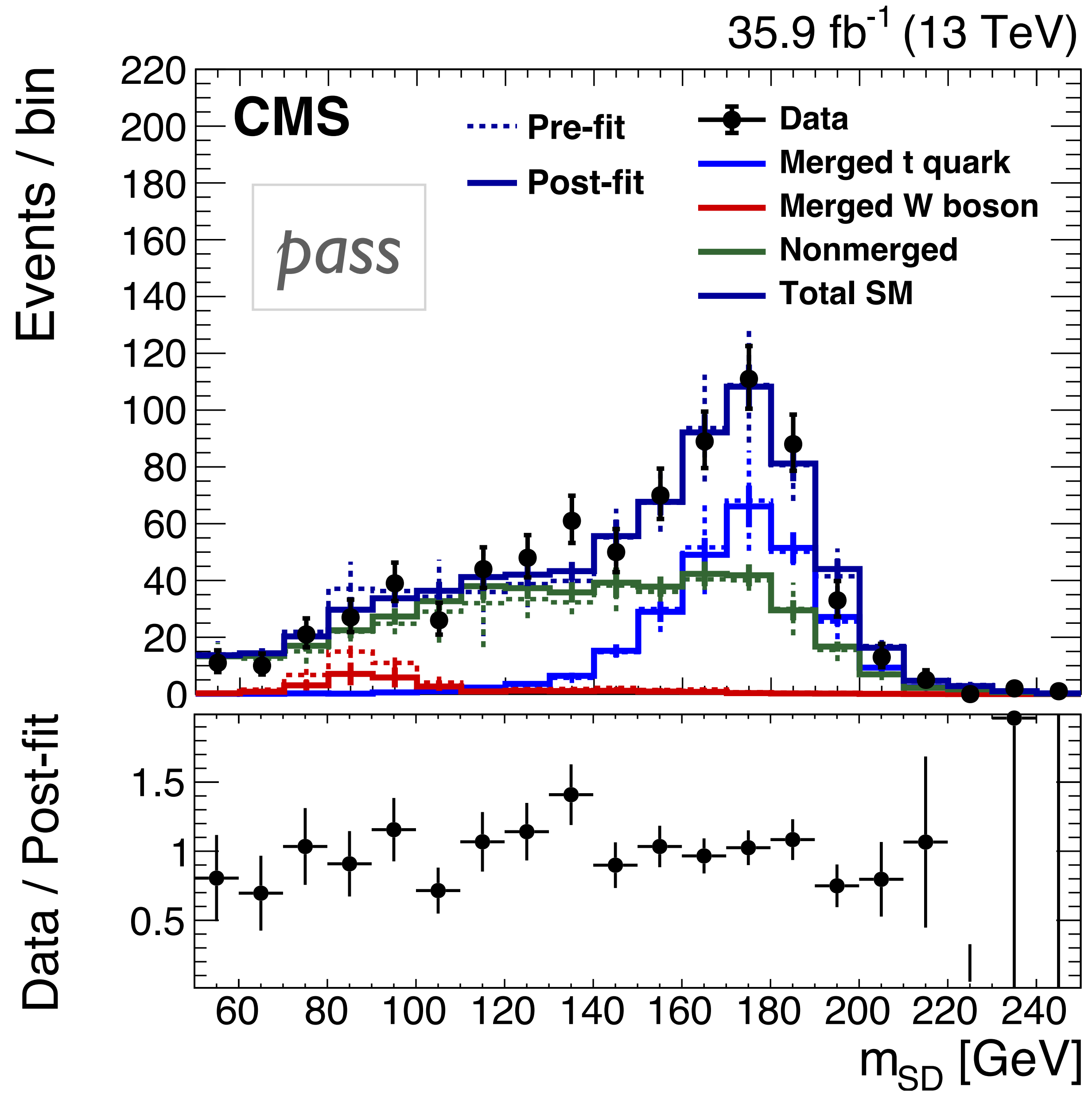


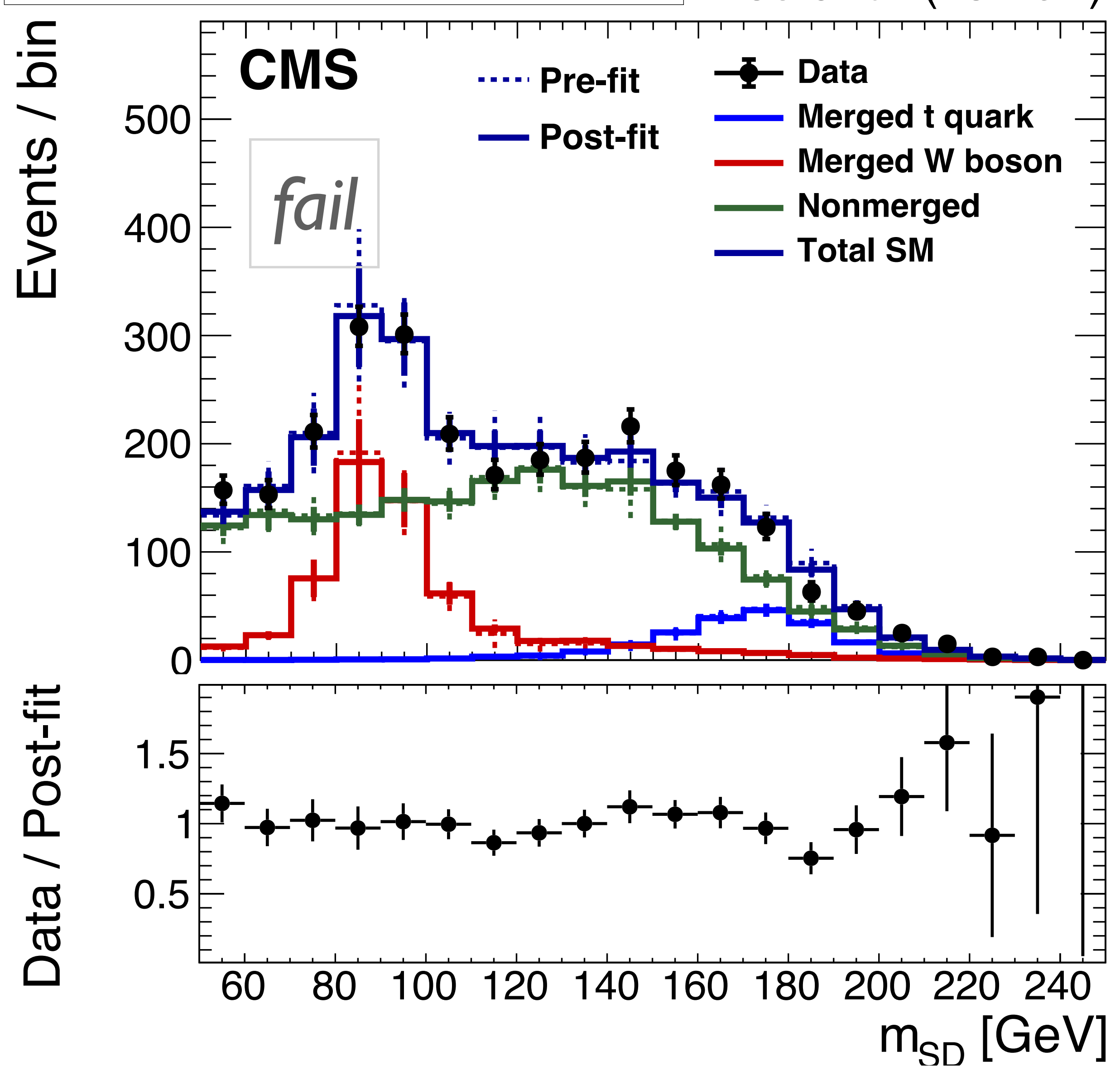


Tagger Calibration in Data

- Crucial to calibrate these taggers in real data for them to be used in analyses
- Top/W tagging efficiency

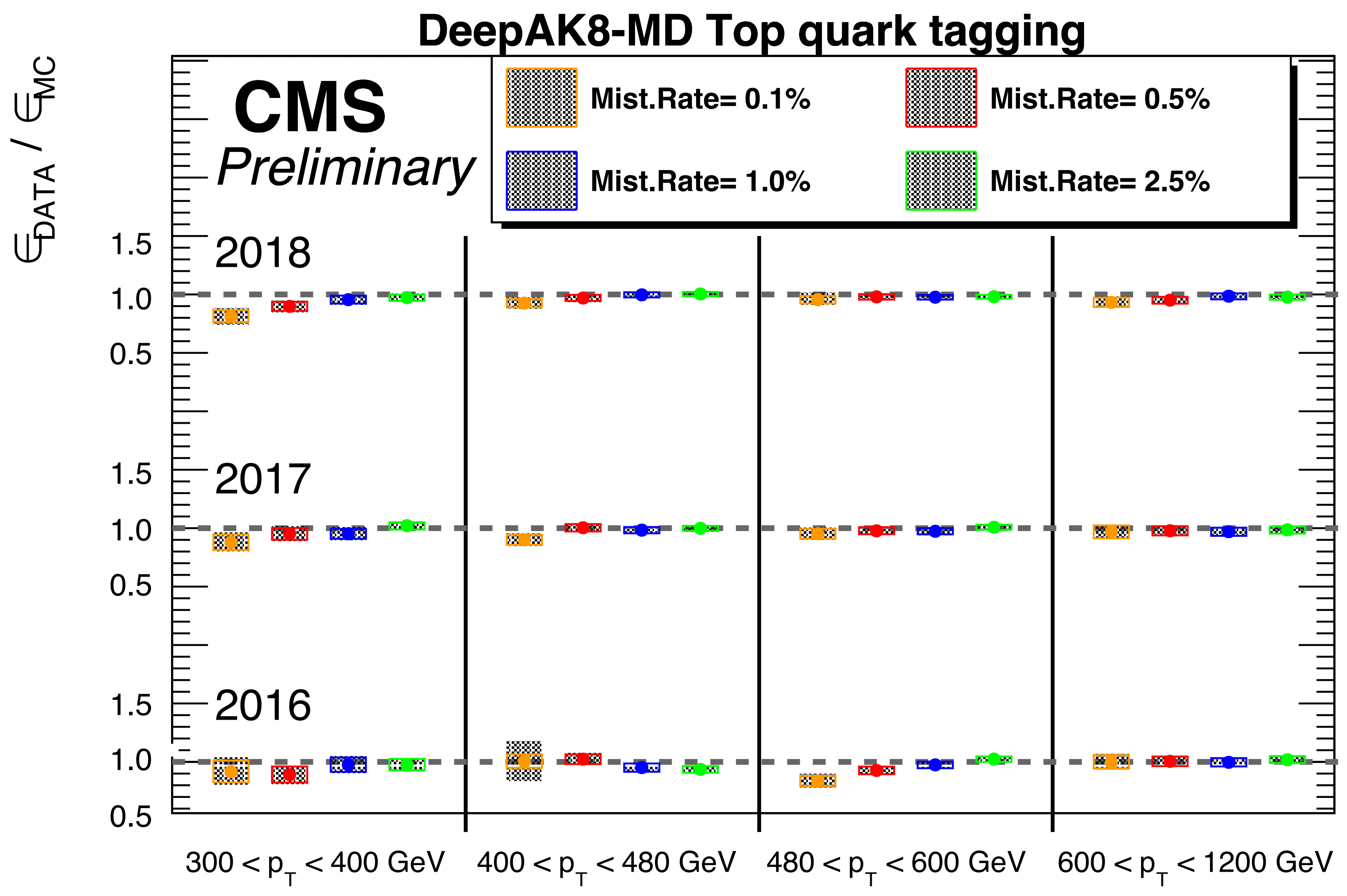
JINST 15 (2020) P06005





Top tagging: DeepAK8-MD

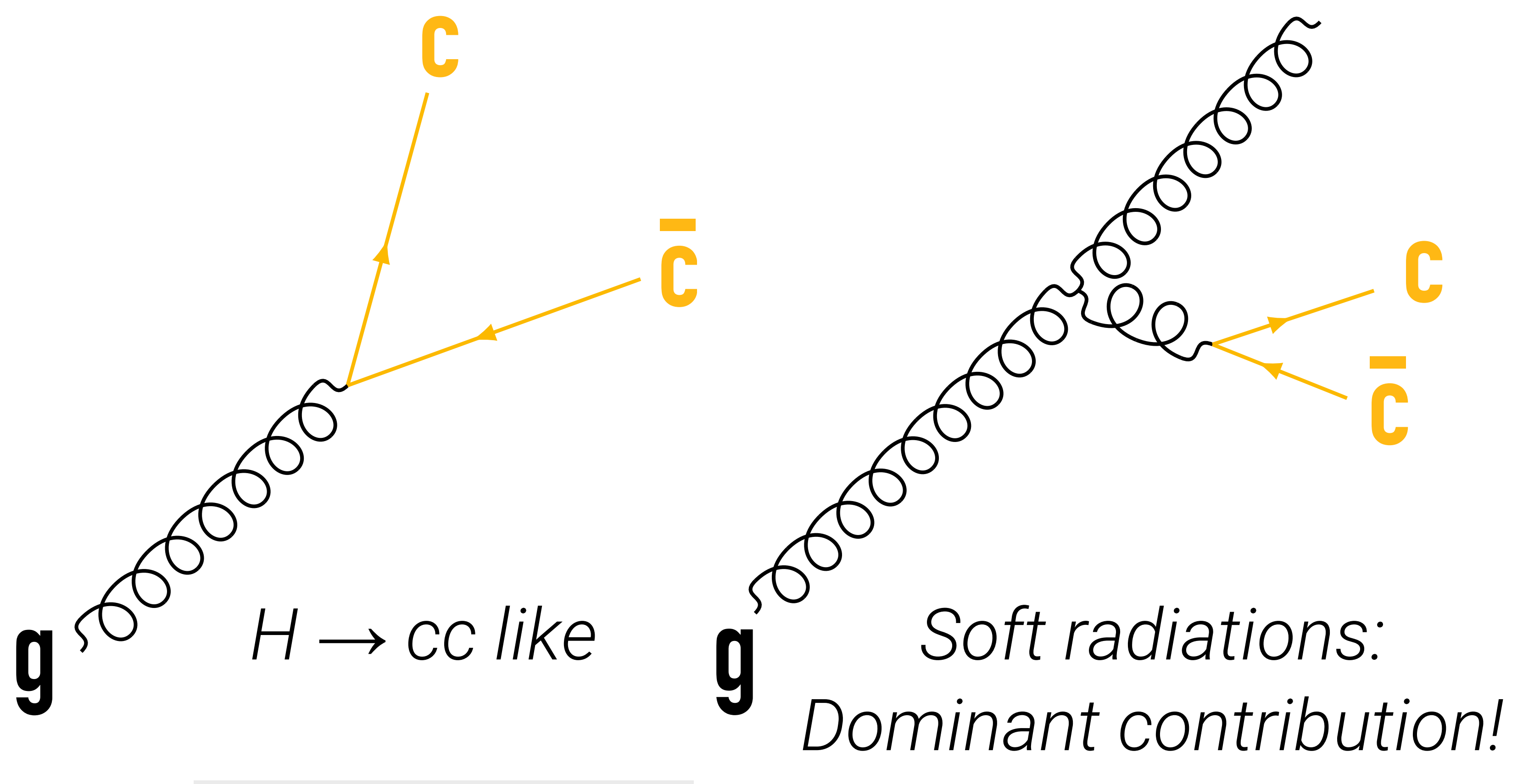
CMS DP-2020/025

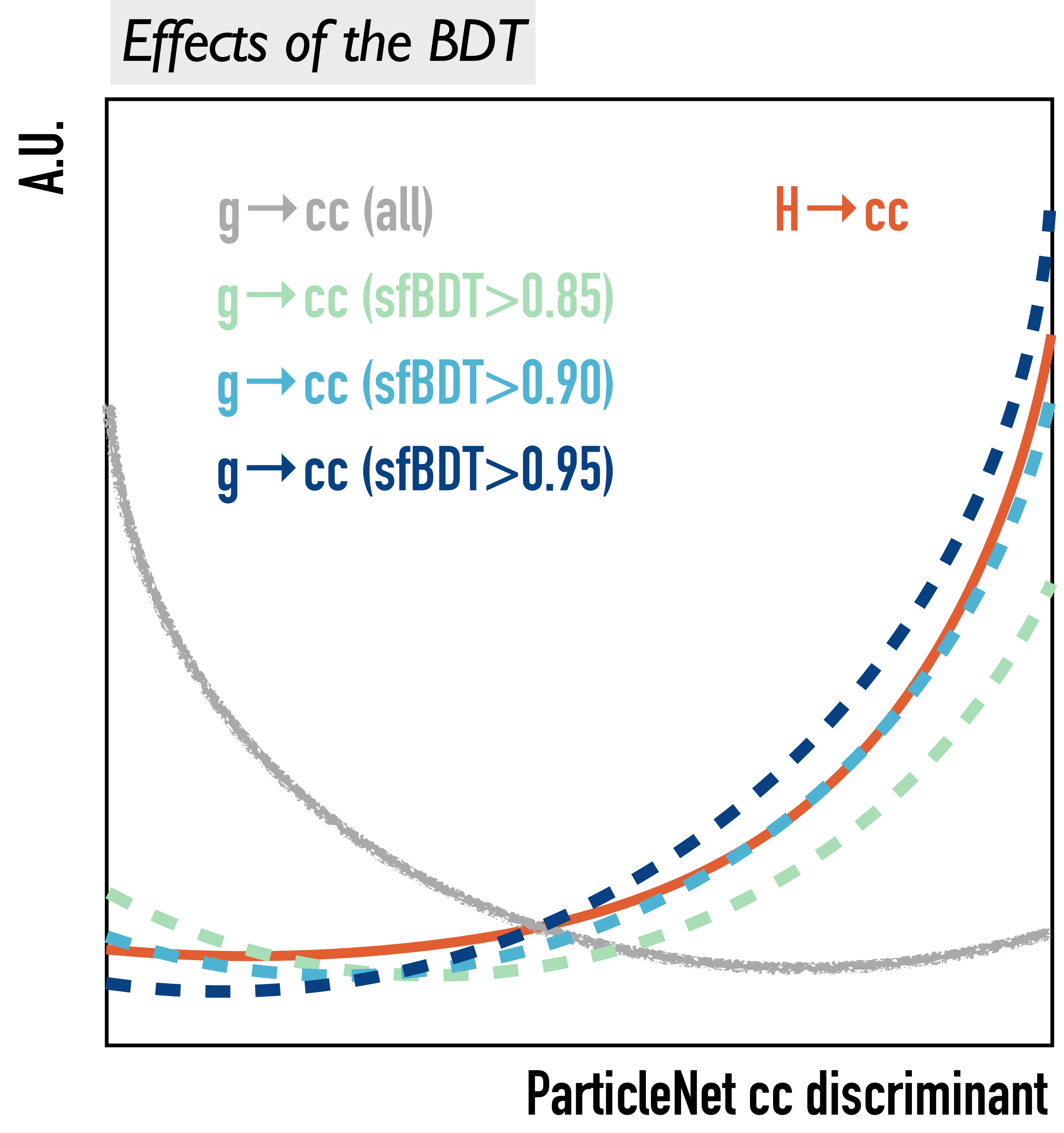


  - measured using the single-μ sample enriched in semi-leptonic ttbar events
  - fit jet mass templates in the “pass” and “fail” categories simultaneously to extract efficiency in data
    - simulation-to-data scale factors SF := eff(data) / eff(MC) derived to correct the simulation
  - jet mass scale and resolution scale factors can also be extracted
- Mistag rates of background jet typically derived directly from analysis-specific control regions

Calibration of the cc-tagger

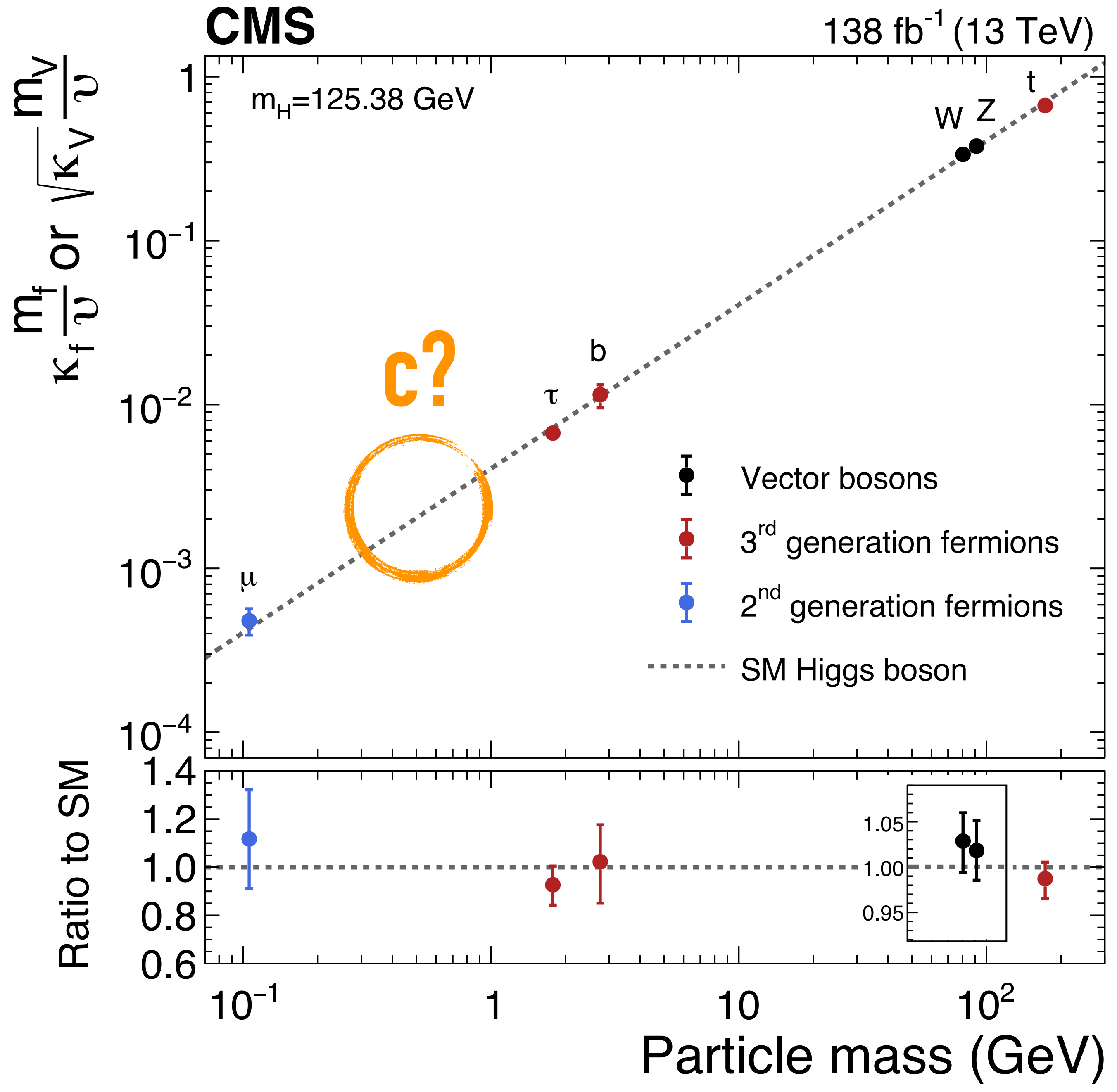
- Need to measure ParticleNet cc-tagging efficiency in data
  - no pure sample of H → cc jets (or even Z → cc) in data
  - using g → cc in QCD multi-jet events as a proxy
- Difficulty: select a phase-space in g → cc that resembles H → cc
  - solution: a **dedicated BDT** developed to distinguish **hard 2-prong splittings** (*i.e., high quark contribution to the jet momentum*) from **soft cc radiations** (*i.e., high gluon contribution to the jet momentum*)
  - also allows to adjust the similarity between proxy and signal jets
    - by varying the sfBDT cut — treated as a systematic uncertainty
- Perform a fit to the secondary vertex mass shapes in the “passing” and “failing” regions simultaneously to extract the scale factors
  - three templates: cc (+ single c), bb (+ single b), light flavor jets
- Derived cc-tagging scale factors typically 0.9—1.3
  - corresponding uncertainties are 20—30%

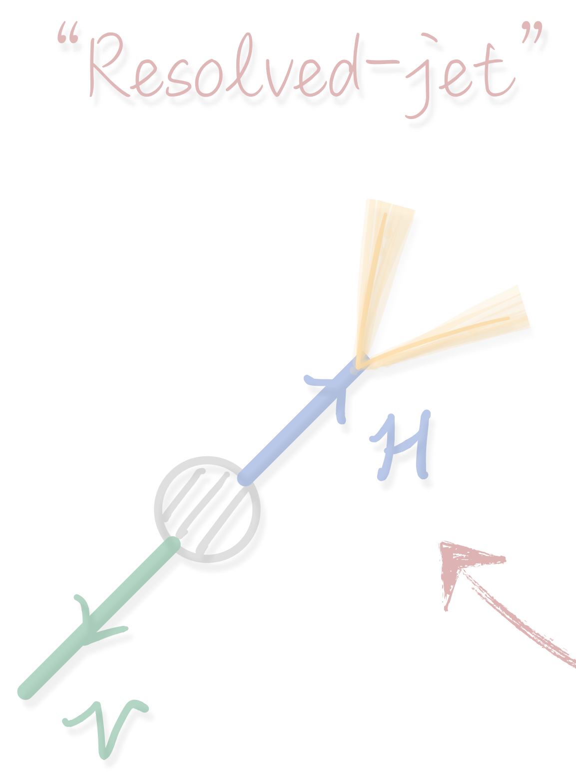


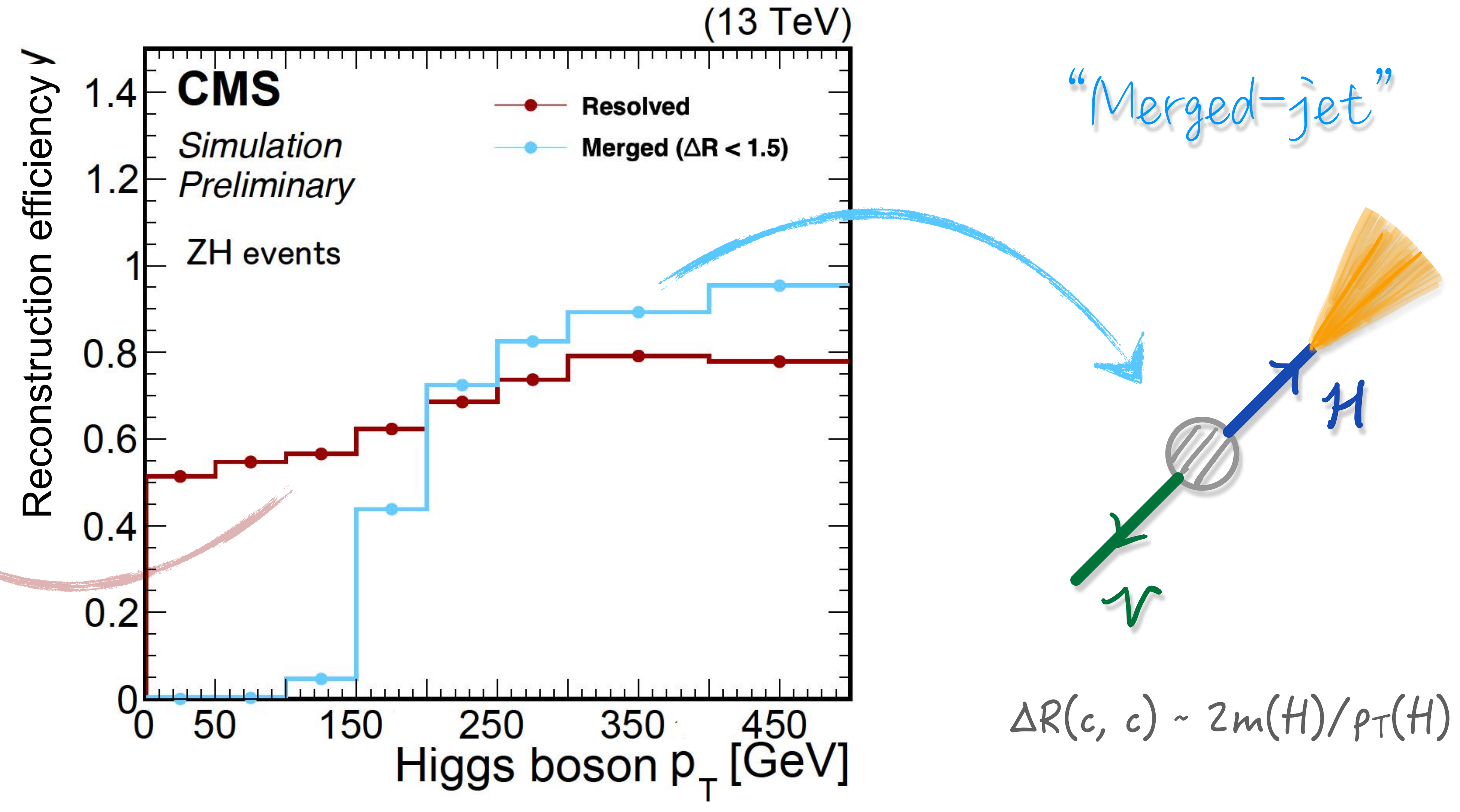


ParticleNet in Action: H→cc Search

- **Higgs-charm coupling: next milestone in Higgs physics**
  - a crucial test of fermion mass generation mechanism in SM
  - H→cc: extremely challenging search at the LHC
    - small branching fraction (~3%) vs enormous backgrounds – **charm tagging** is the key
- **Innovative approach**: search for VH(H→cc) in the **"merged-jet" topology**
  - reconstructs H→cc decay with one large-R jet (R=1.5)
  - then: exploits advanced ML for H→cc identification





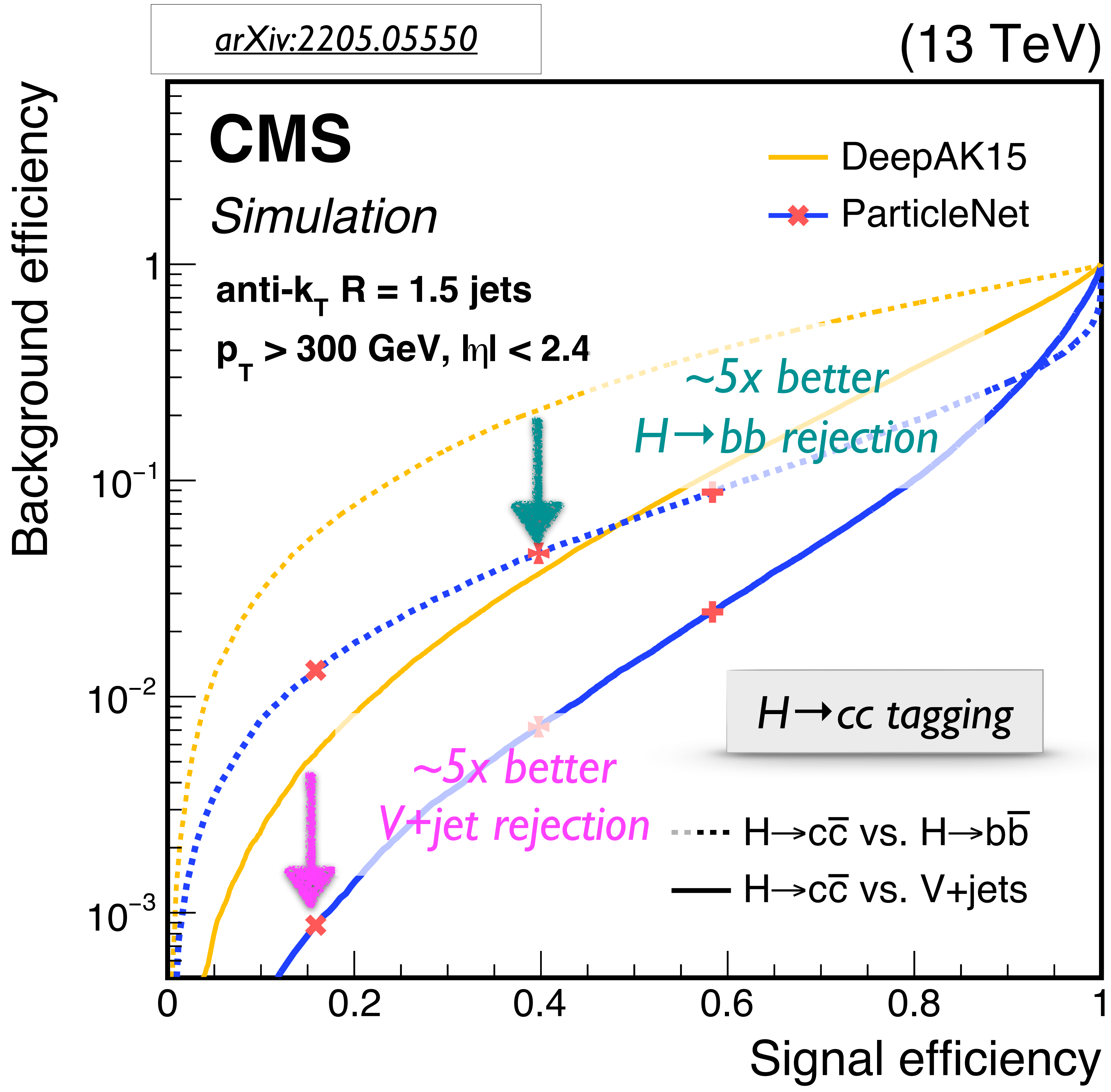


Why merged-jet topology?

- *better signal purity at higher p_T*
- *higher reconstruction efficiency with large-R jets*
- *better exploiting correlations between the two charm quarks — especially with deep learning*

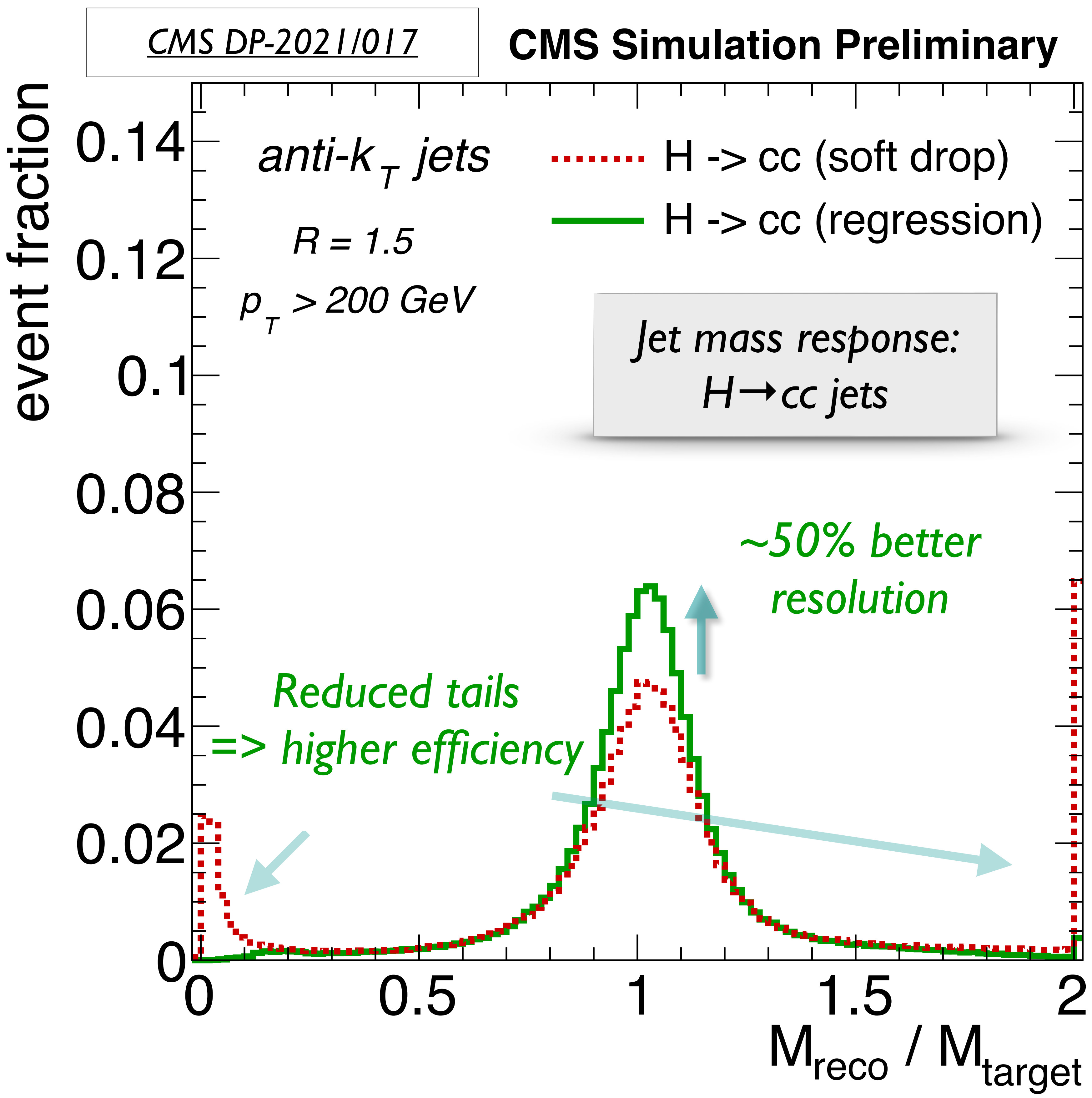
ParticleNet in Action: H→cc Search

- **ParticleNet for H→cc jet tagging and mass reconstruction: substantial improvements**



ParticleNet tagger for H→cc tagging

>2x improvement in final sensitivity



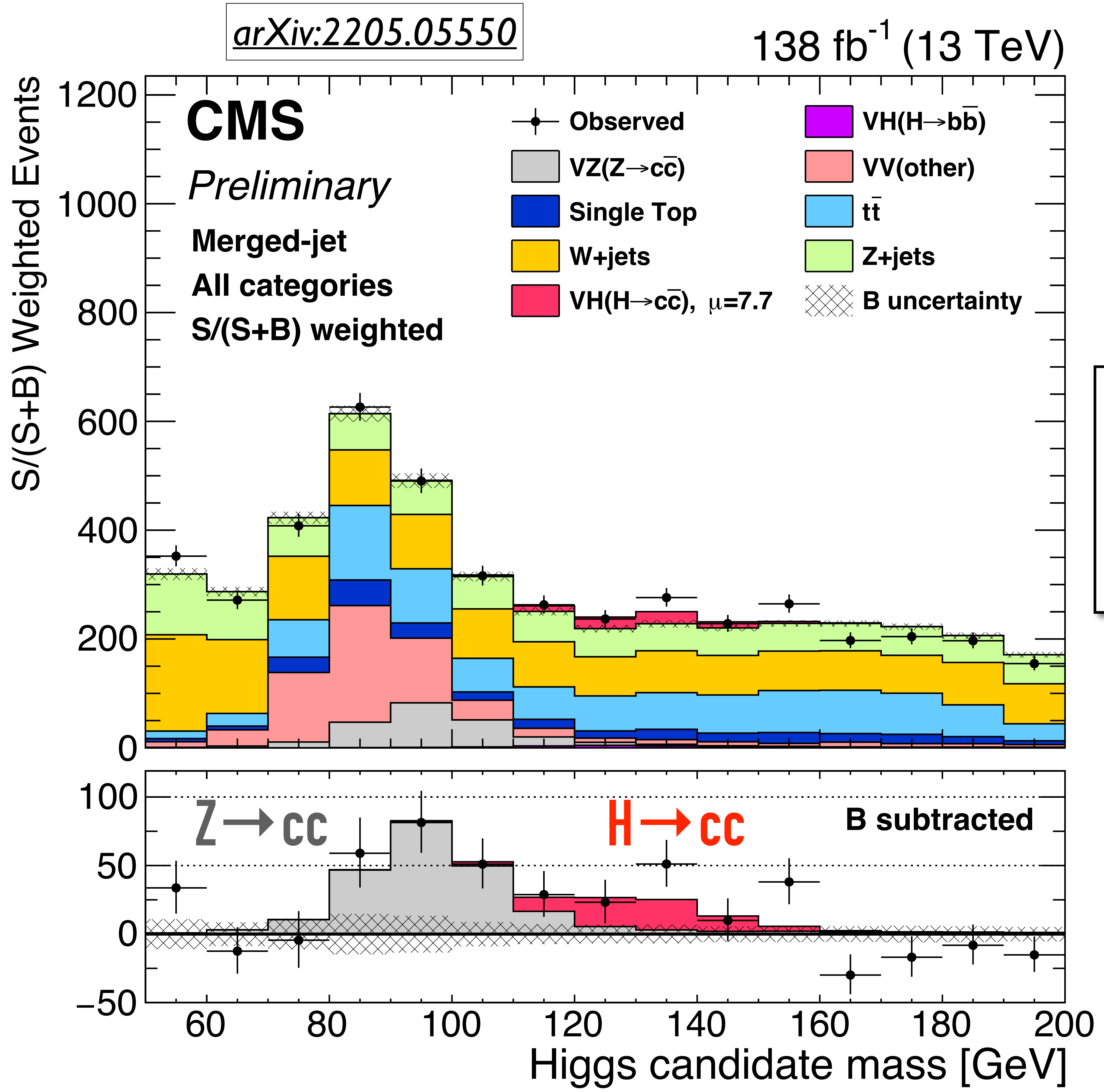
ParticleNet-based jet mass regression

~20-25% improvement in final sensitivity

ParticleNet in Action: H→cc Search

- **ParticleNet for H→cc jet tagging and mass reconstruction: substantial improvements**

arXiv:2205.05550



Most stringent limit on H→cc to date.
- *~4x higher sensitivity than the ATLAS search*
- *Comparable to previous HL-LHC projection, but with only 5% of the data.*

First observation of Z→cc at a hadron collider!

EXTRA: MORE ABOUT PRACTICALITIES

DISCLAIMER

- These are based on my very personal experiences in using ML to solved HEP problems
  - and highly biased to collider experiments / jet tagging
  - so please take them with a large grain of salt

- My take is that ML is 50% science and 50% engineering
  - and probably another 20% alchemy…
  - so things that should work may not necessarily work in reality…

Data Matters

- Always inspect your training data first
  - check the distributions for different classes / in different phase space (p_T, energy scale, vs time, ...)
    - do they make sense?
    - are the trends expected?
    - do you see expected / unexpected separation power between different classes?
  - check for significant outliers / NaN / Inf / etc.

- Think carefully about how to choose your training data, how to define training target (truth labels, etc.)
  - highly case dependent, but this can have significant impact on the performance, generalization power, etc.

Data Matters (II)

- Mindful preprocessing
  - neural networks work best with “Gaussian-like” inputs
    - transform the inputs if needed, e.g., log(…) or tanh(const x …) for long-tail distributions (energy, p_T, mass, d_0/dz, …)
    - shift/scale the inputs, and then truncate (if needed) — extreme outliers can destabilize training and affect performance
    - use normalization layers (BatchNorm, LayerNorm, …)
  - dealing with phase space difference between classes => reweighting (or better, sampling) if needed
  - decorrelation (e.g., mass decorrelation in jet tagging)

- Get more data whenever you can
  - if can not: consider data augmentation (rotation, reflection, smearing, dropout, …)

Baseline First, then Iterate

- A good practice is to always establish a baseline algorithm first before developing more advanced approaches
  - with a baseline ready, then one can easily evaluate if the new algorithm is too good (to be true), or too bad (so probably missing something obvious), or just promising :)
  - if the problem is not new and a baseline already exists – just use/adapt it
  - the baseline can be a cut-based / rule-based algo, or a shallow model (e.g., BDT)
    - consider trying newer BDT libraries – even XGBoost is no longer the state-of-the-art
      - e.g., TensorFlow Decision Forests (TF-DF), LightGBM, CatBoost, …

Some Useful Links

- Tutorial / hands-on textbook on Deep Learning:
  - **Dive into Deep Learning**: https://d2l.ai/

- More on Graph Neural Networks:
  - https://distill.pub/2021/gnn-intro/
  - https://distill.pub/2021/understanding-gnns/
  - and many more interactive ML posts on https://distill.pub

- HEP x ML:
  - **A Living Review of Machine Learning for Particle Physics**: https://github.com/iml-wg/HEPML-LivingReview

- My little framework:
  - **weaver** (data loading and classification/regression training pipelines): https://github.com/hqucms/weaver-core
  - weaver-examples (under construction): https://github.com/hqucms/weaver-examples