Graph Neural Networks: Development and Deployment in Particle Physics

Lazar Novakovic and Loukas Gouskos

Brown University, Department of Physics

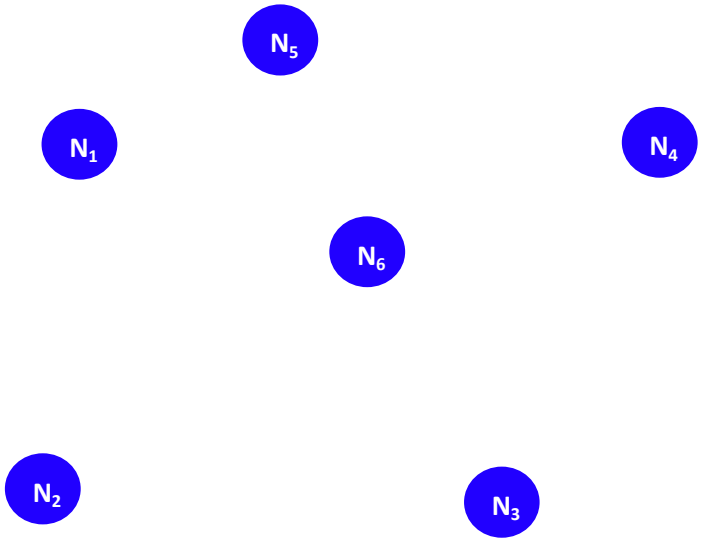
AI Winter AI School (2025)

What is a Graph?

- A mathematical structure used to model pairwise relations between points (objects/particles/measurements)

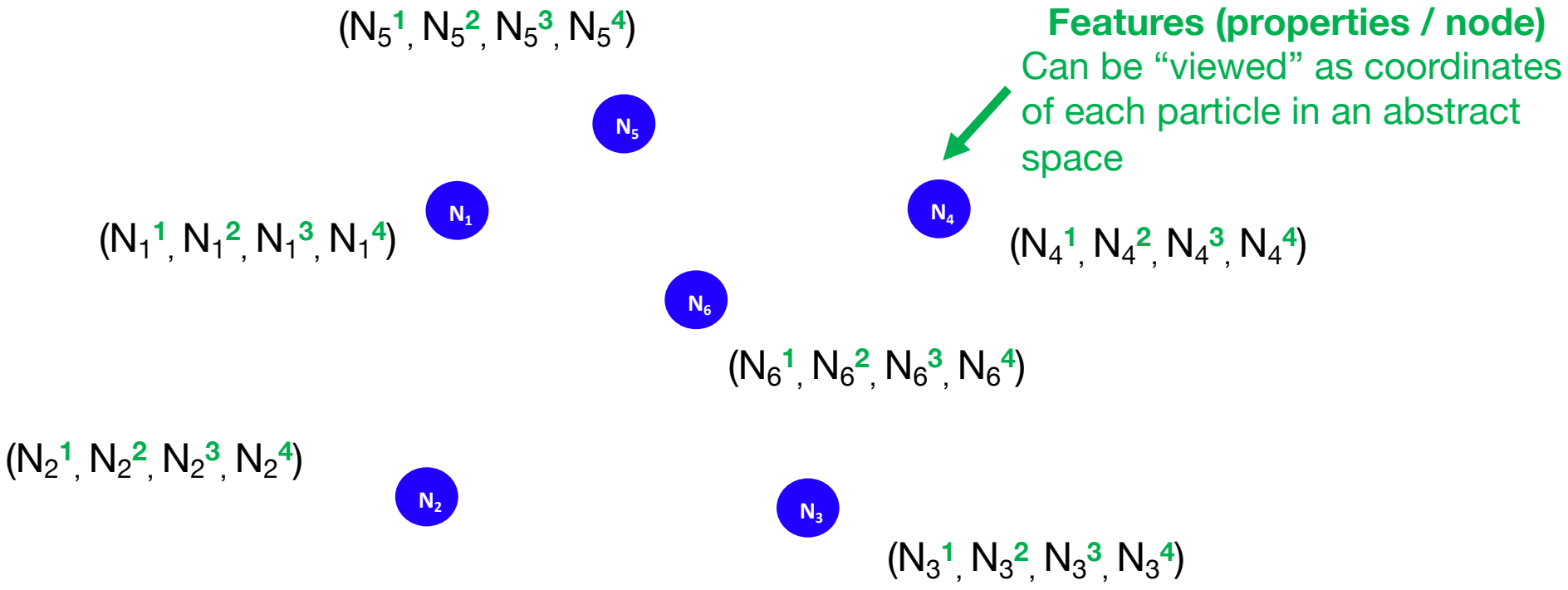
What is a Graph?

- A mathematical structure used to model pairwise relations between points (objects/particles/measurements)
  - Set of **nodes/vertices (N_i)**



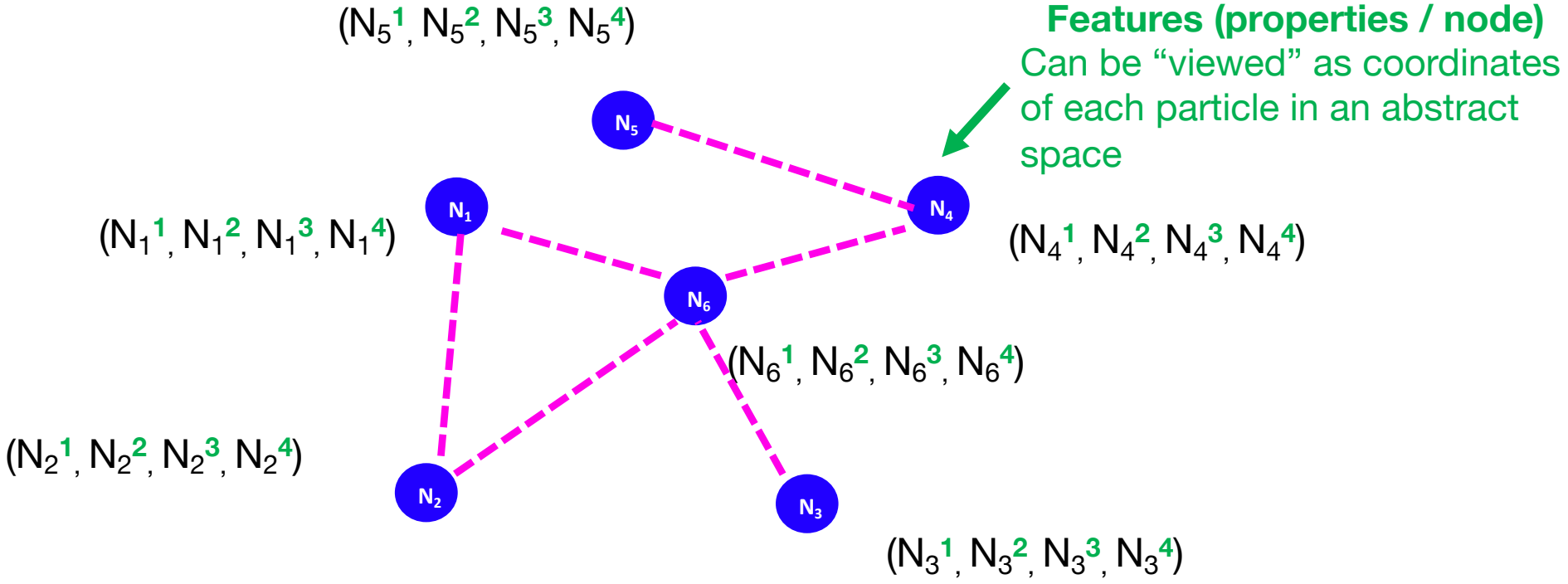
What is a Graph?

- A mathematical structure used to model pairwise relations between points (objects/particles/measurements)
  - Set of **nodes/vertices (N_i)**



What is a Graph?

- A mathematical structure used to model pairwise relations between points (objects/particles/measurements)
  - Set of **nodes/vertices (N_i)**
  - Set of **edges/connections** between **nodes**



What is a Graph?

- A mathematical structure used to model pairwise relations between points (objects/particles/measurements)
  - Set of **nodes/vertices (N_i)**
  - Set of **edges/connections** between **nodes**

$$G = (V, E)$$

$$V = \{v_1, v_2, \ldots, v_n\}$$

$$E = \{e_1, e_2, \ldots, e_m\}$$

What is a Graph?

- Mathematical representation via **adjacency matrix** (A)
  - Vertices: $V = \{v_1, v_2, v_3\}$
  - Edges: $E = \{\{v_1, v_2\}, \{v_2, v_3\}\}$
  - If vertices i,j form an edge:

$$a_{ij} = 1$$

    - or equal to the “weight” of the edge
  - else:

$$a_{ij} = 0$$

What is a Graph?

- Mathematical representation via **adjacency matrix** (A)
  - Vertices: $V = \{v_1, v_2, v_3\}$
  - Edges: $E = \{\{v_1, v_2\}, \{v_2, v_3\}\}$
  - If vertices i,j form an edge:

$$a_{ij} = 1$$

    - or equal to the "weight" of the edge
  - else:

$$a_{ij} = 0$$

$$A = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

Graph Neural Networks

- Graphs: Powerful math structures to model relationships
  - Yet, simply representing data as a graph isn't enough

Graph Neural Networks

- Graphs: Powerful math structures to model relationships
  - Yet, simply representing data as a graph isn't enough
- **G**raph **N**eural **N**etworks (**GNN**s) come into play!

Graph Neural Networks

- Graphs: Powerful math structures to model relationships
  - Yet, simply representing data as a graph isn't enough
- **G**raph **N**eural **N**etworks (**GNN**s) come into play!
  - Real-world data: complex relationships
    - Traditional ML: struggle with non-Euclidean data
  - GNNs: can learn to extract features from graph structures
    - ideal where relationships (edges) b/w entities (nodes) matter

Graph Neural Networks

- Graphs: Powerful math structures to model relationships
  - Yet, simply representing data as a graph isn't enough
- **G**raph **N**eural **N**etworks (**GNN**s) come into play!
  - Real-world data: complex relationships
    - Traditional ML: struggle with non-Euclidean data
  - GNNs: can learn to extract features from graph structures
    - ideal where relationships (edges) b/w entities (nodes) matter
- Core Idea:
  - Message Passing
    - Nodes exchange info w/ their neighbors to aggregate local context
  - Feature Transformation
    - Node features updated based on neighboring info (e.g., via NN)
  - Iterative Updates (repeat over multiple layers)
    - Info flows across Graph → Capture long-range dependences

Overview of this lecture

- Executive summary: LHC experiments
- Why Graph Neural Networks (GNNs) in Particle Physics
- Hands-on:
  - GNNs for classification (i.e., discriminate b/w different particles)
  - GNNs for clustering (i.e., group detector signals)

The Large Hadron Collider (LHC) at CERN

CMS

LHCb

ALICE

ATLAS

LHC
Higgs discovery

SPS
W/Z discovery

Geneva
airport

The Large Hadron Collider (LHC) at CERN

CMS

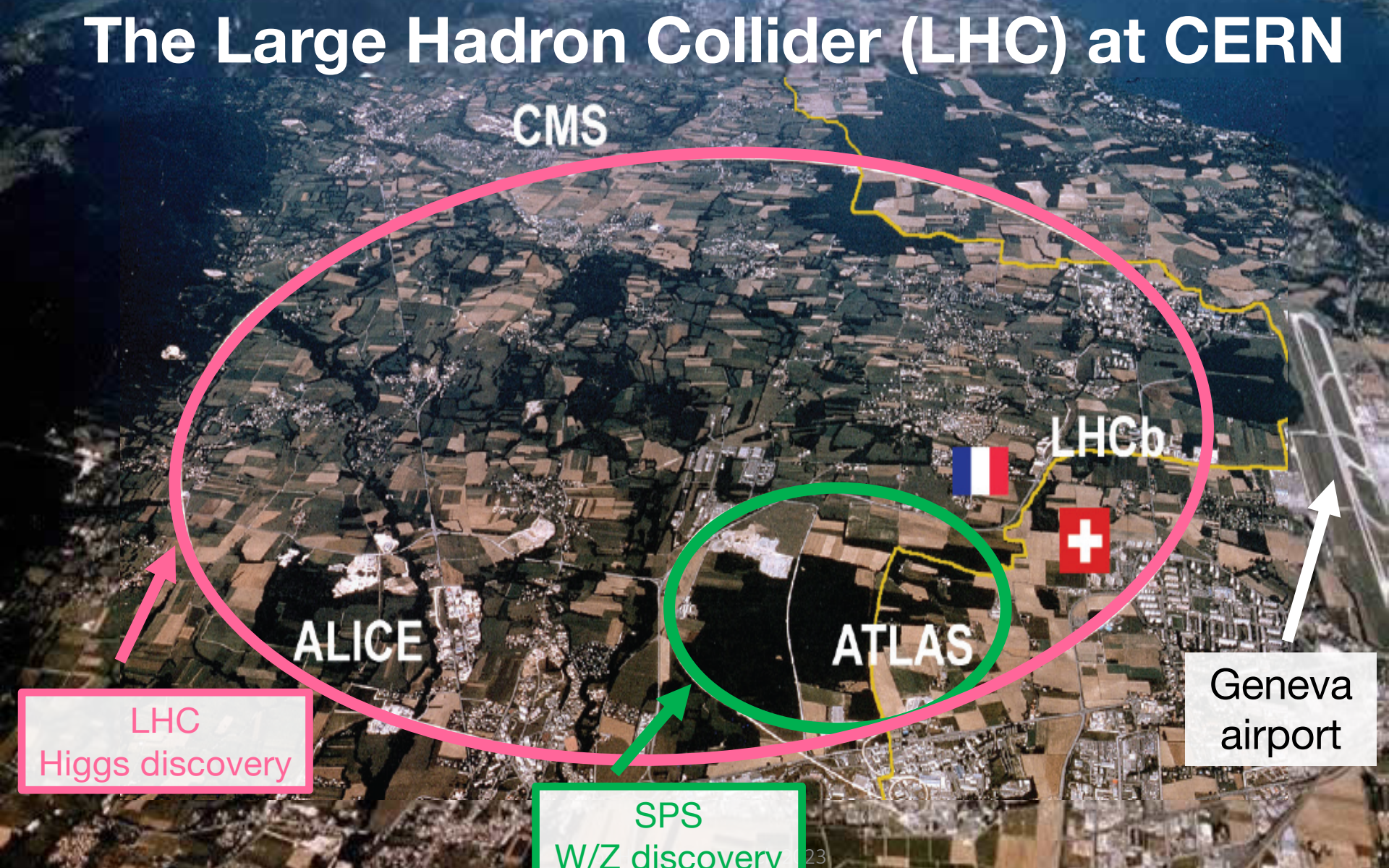
- **LHC:** The largest particle collider in the world
  - **27km** perimeter
  - >1.2K dipole magnets @ 8T
    - cooled @ -271° C
  - Colliding protons at energies: **7 – 14 TeV**
    - currently @13.6 TeV (Run 3)
      [almost 14K times proton mass]
  - 2808 bunches collide every **25nsec**
    O(10^{12}) protons/bunch

LHC
Higgs discovery

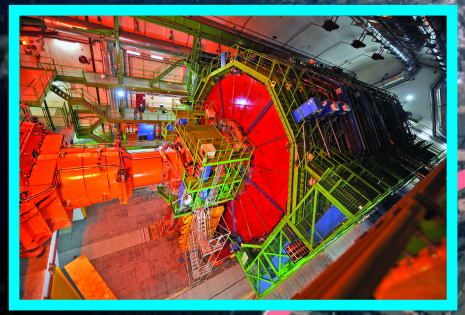
SPS
W/Z discovery

Geneva
airport

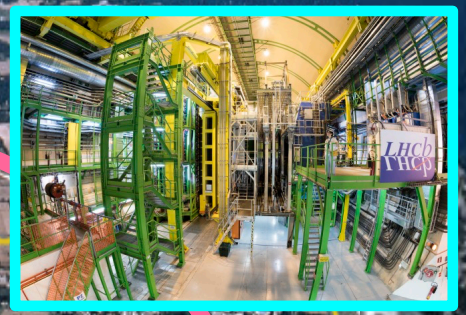
The Large Hadron Collider (LHC) at CERN



The Large Hadron Collider (LHC) at CERN



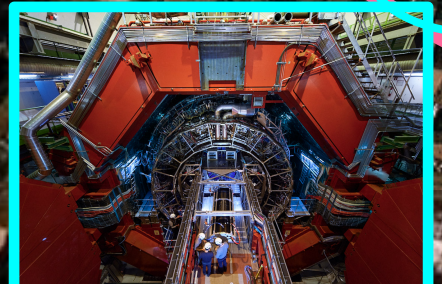
CMS

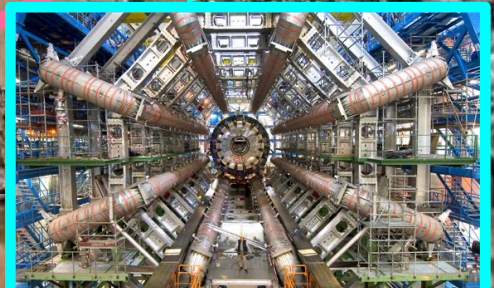


LHCb

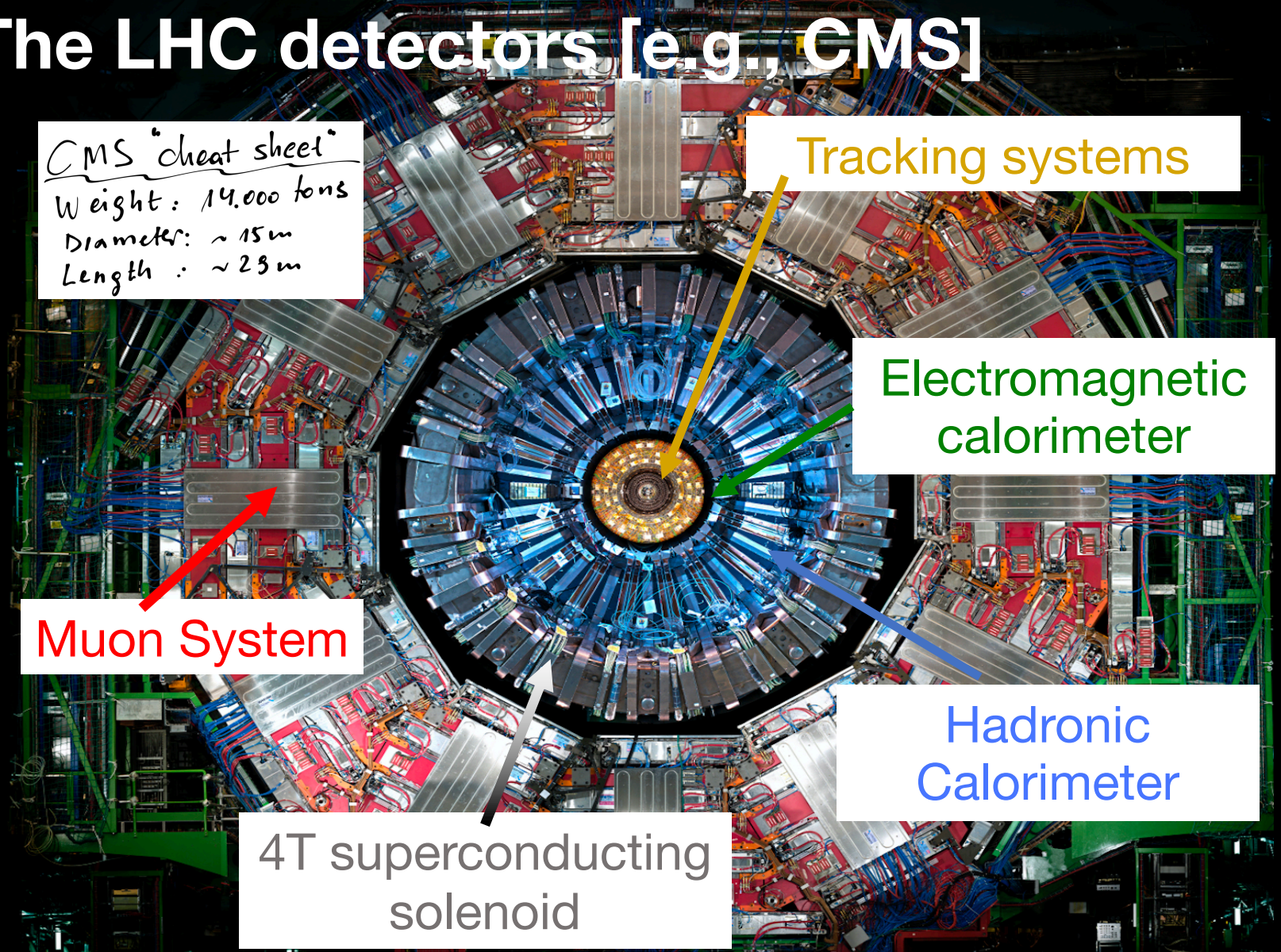
ALICE

ATLAS

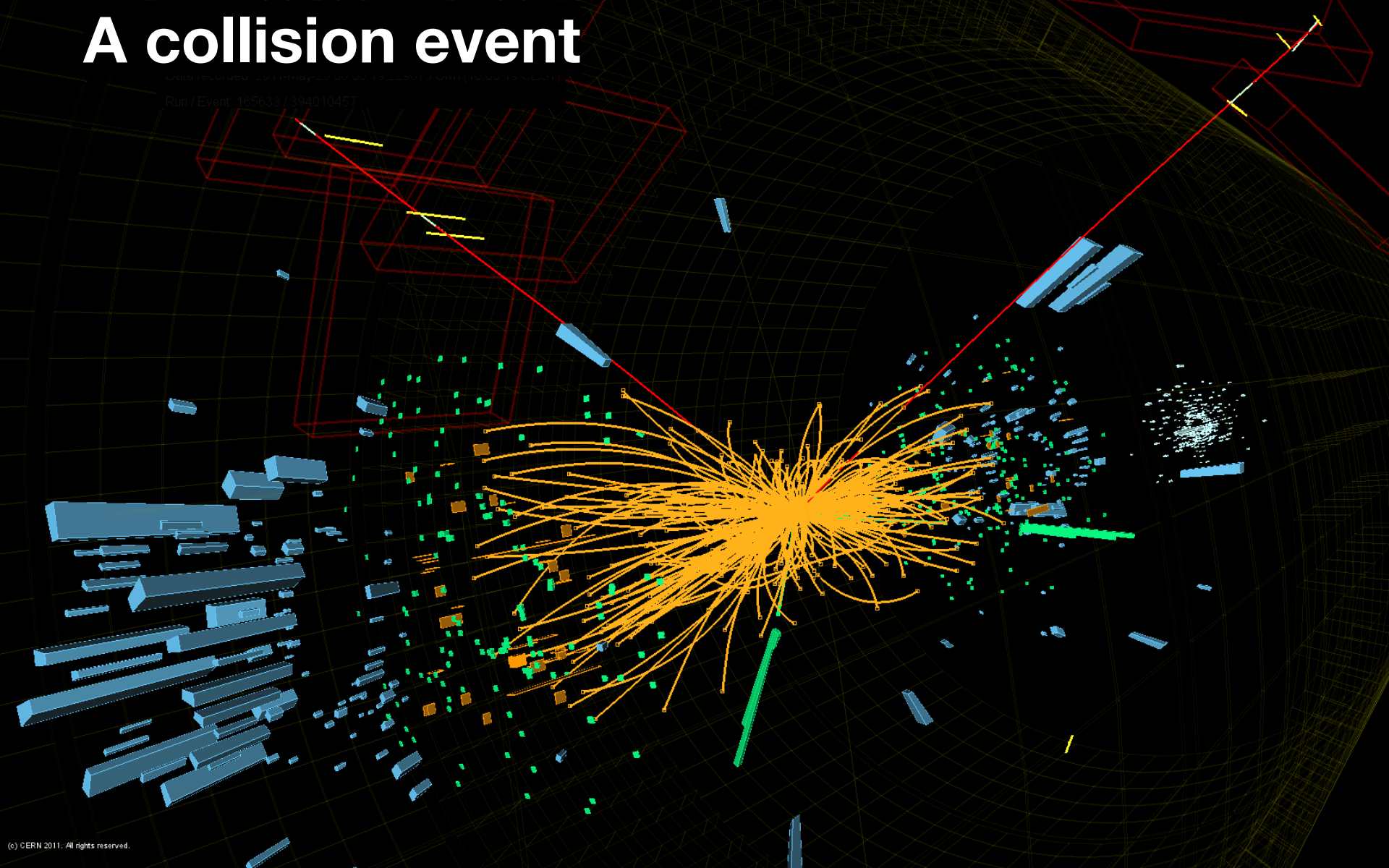




The LHC detectors [e.g., CMS]

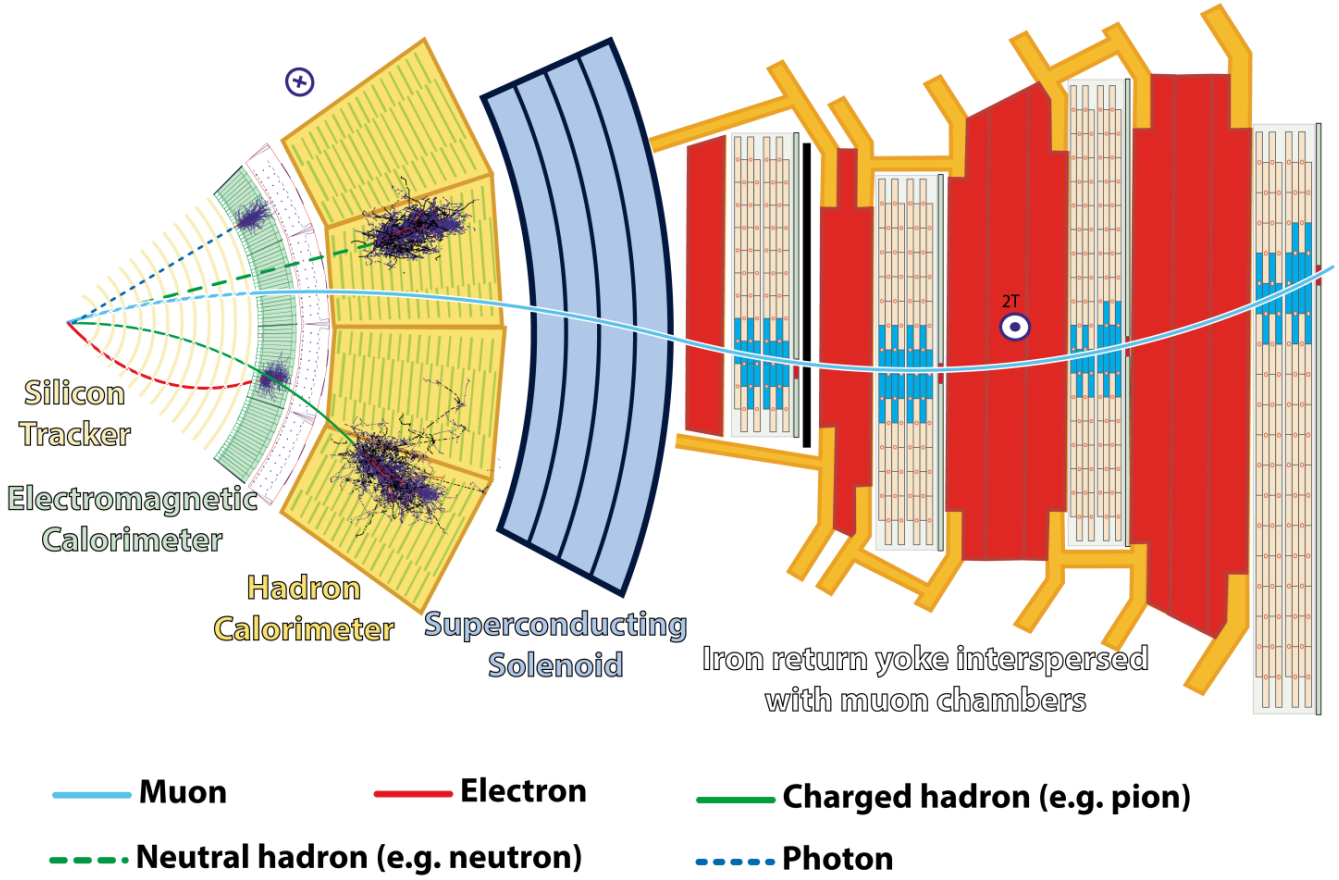


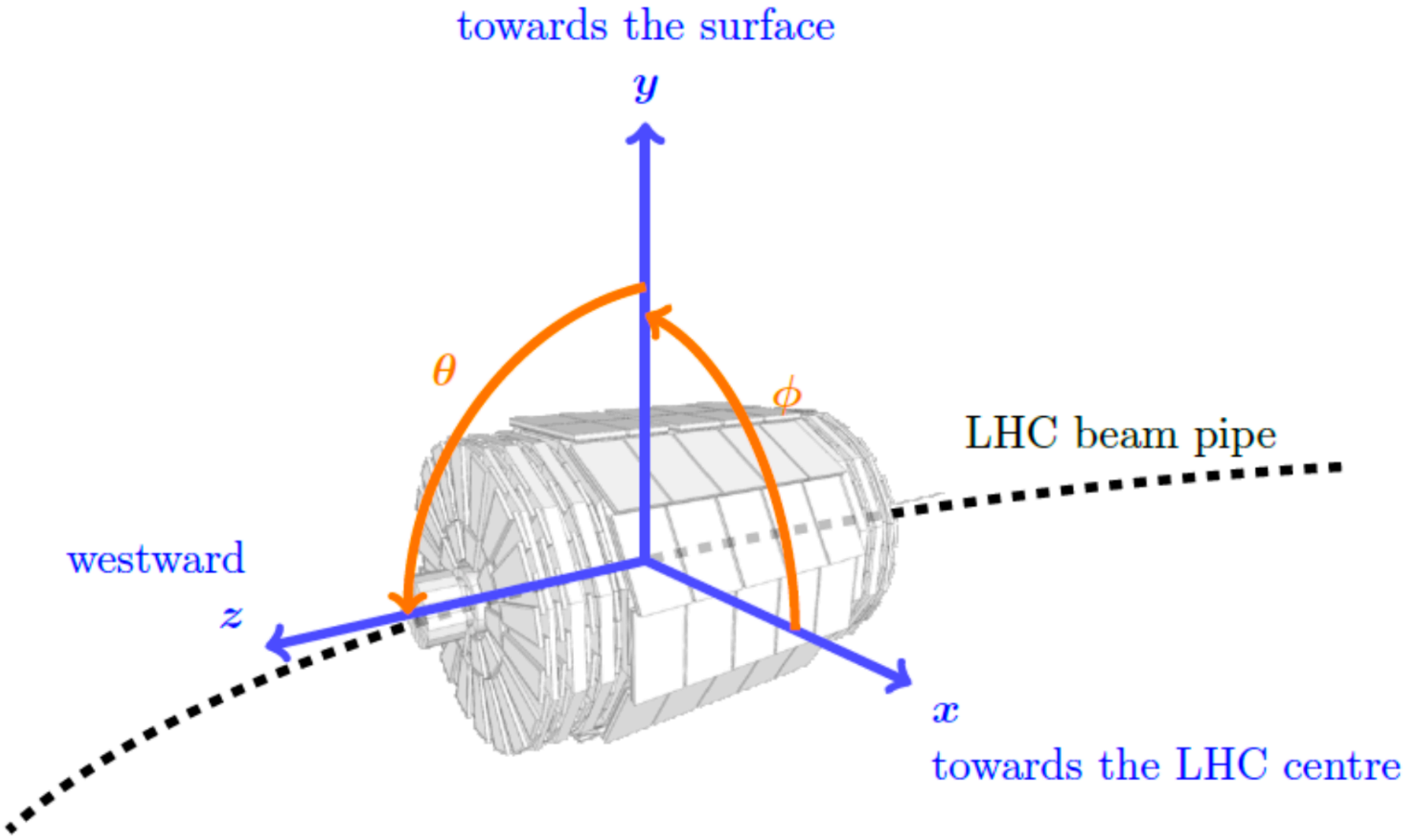
A collision event



Event reconstruction

- Combine information from all subdetectors
  - Mutually exclusive list of particles [i.e. Particle Flow]




towards the surface
y
θ
ϕ
LHC beam pipe
westward
z
x
towards the LHC centre

Key points

- **High-dimensional** and **sparse** data
  - Complex events, high-dimensional data, where most regions contain little or no signal → sparse data
- **Heterogeneous** events
  - different types of particles (e.g., muons, photons, hadrons)
  - each interacting differently with the sub-detectors
- **Long-range** dependencies
  - Physical processes often involve relationships between distant regions of the detector
    - e.g., correlations between particles traveling in different directions
- **Complex** topology
  - Particle collisions produce jets, tracks, and vertices that naturally form graph-like structures rather than regular grids

Advantages of GNNs

- Flexibility
  - Do not impose a fixed structure on the data
    → highly adaptable to various structures
    - regular grids (where they can approximate CNNs) to highly irregular networks

Advantages of GNNs

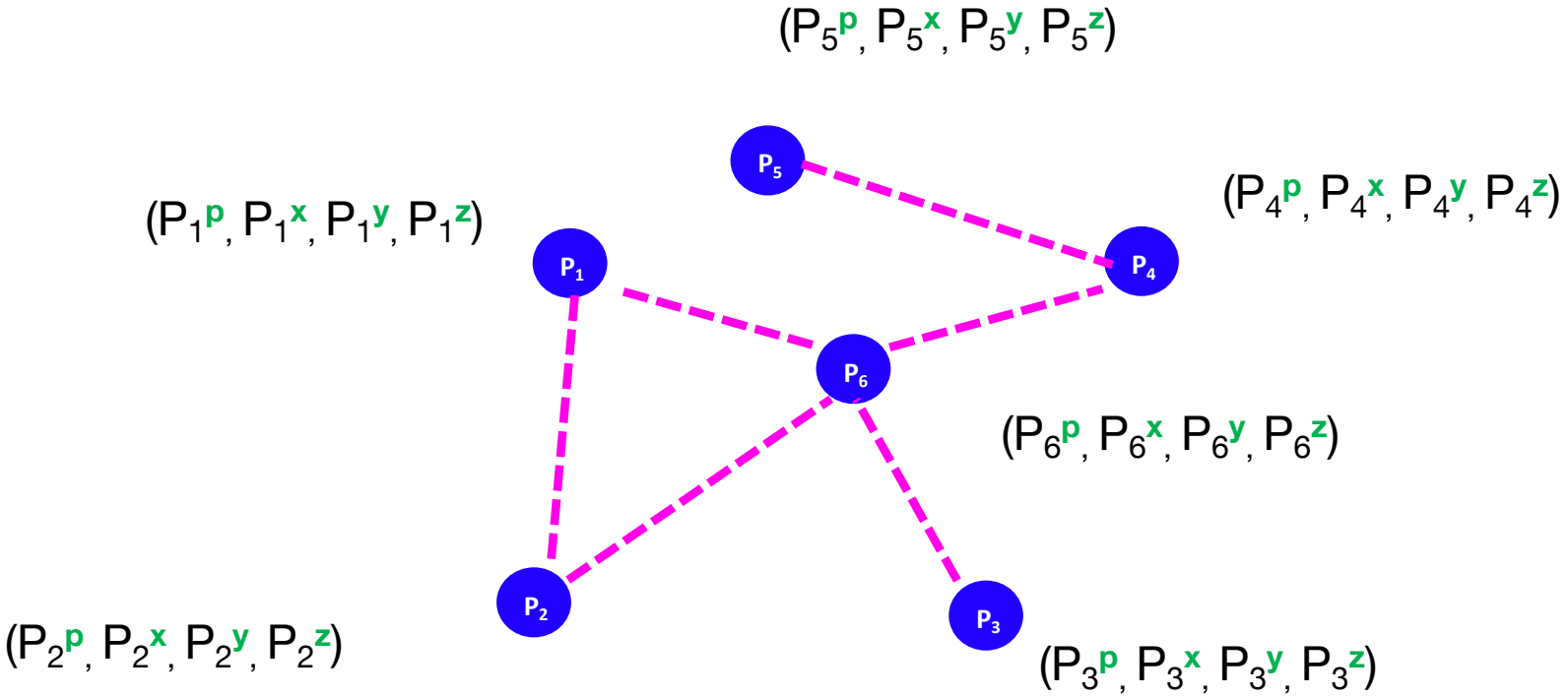
- Flexibility
  - Do not impose a fixed structure on the data
    → highly adaptable to various structures
    - regular grids (where they can approximate CNNs) to highly irregular networks
- Local patterns
  - Exploit local structure in graphs [such as CNNs].
    - GNNs: capture the properties of a node based on its surroundings, respecting the graph topology

Advantages of GNNs

- Flexibility
  - Do not impose a fixed structure on the data
    → highly adaptable to various structures
    - regular grids (where they can approximate CNNs) to highly irregular networks
- Local patterns
  - Exploit local structure in graphs [such as CNNs].
    - GNNs: capture the properties of a node based on its surroundings, respecting the graph topology
- Efficiency:
  - More Parameter-efficient than dense networks
    - GNNs focus on “relationships” between input data that matter
      - ignore the irrelevant ones

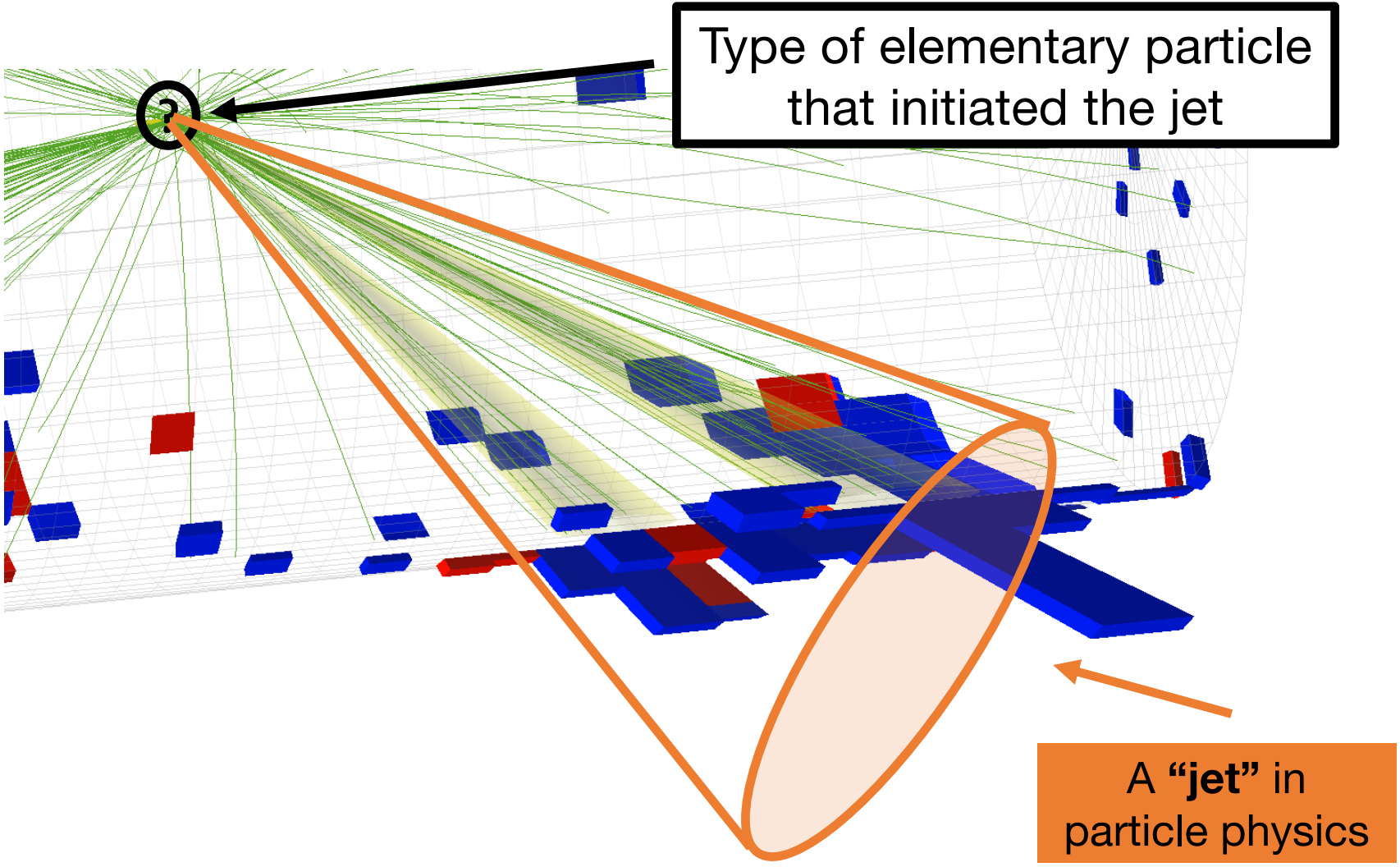
GNNs in particle physics

- **Nodes:** particles, detector signals, etc..
- **Features:** momentum, spatial coo/ntes (x,y,z), ID, ..
- **Edges:** correlations between the particles

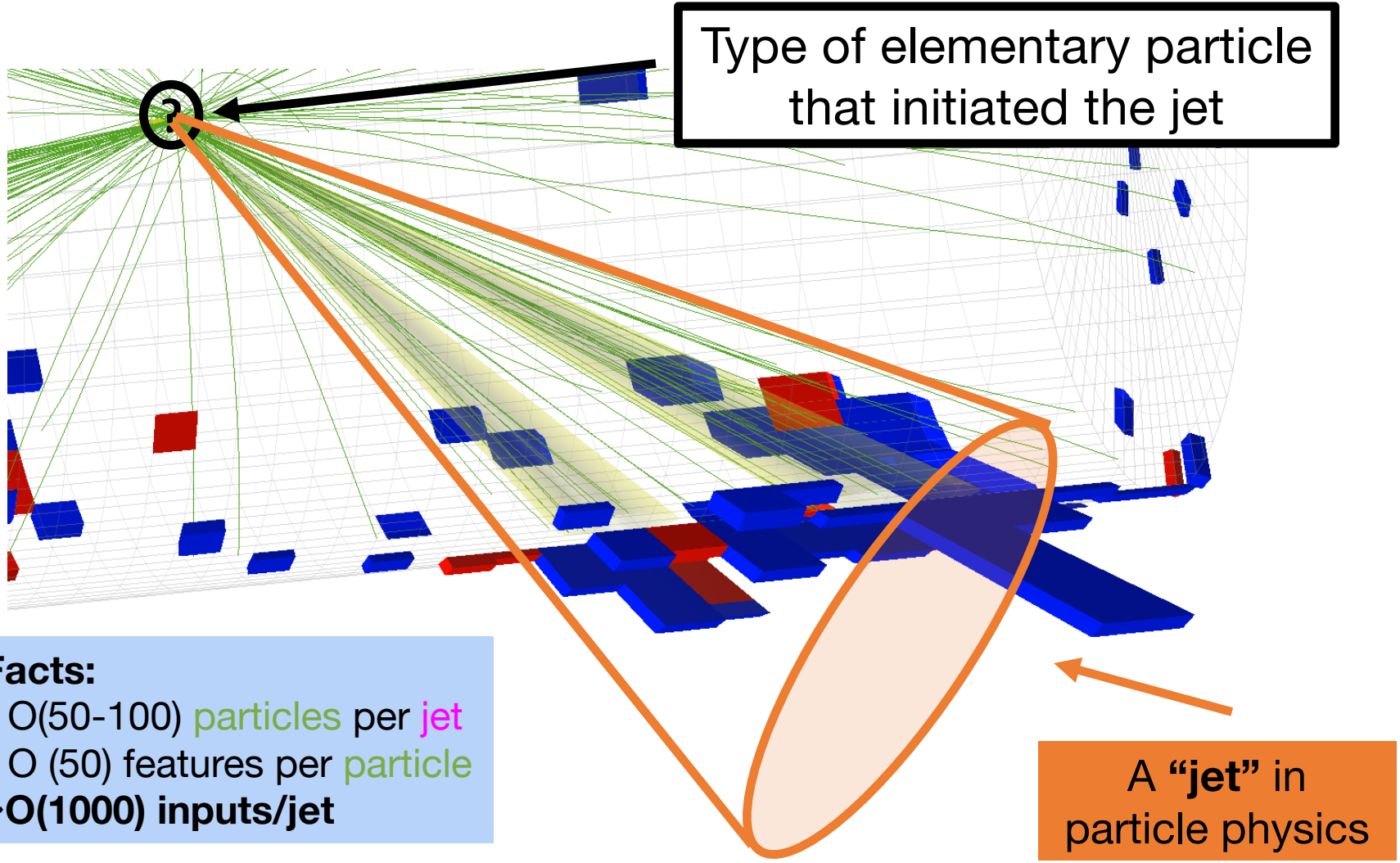


Exercise 1: Classify different particle types

Exercise 1: Classify different particle types



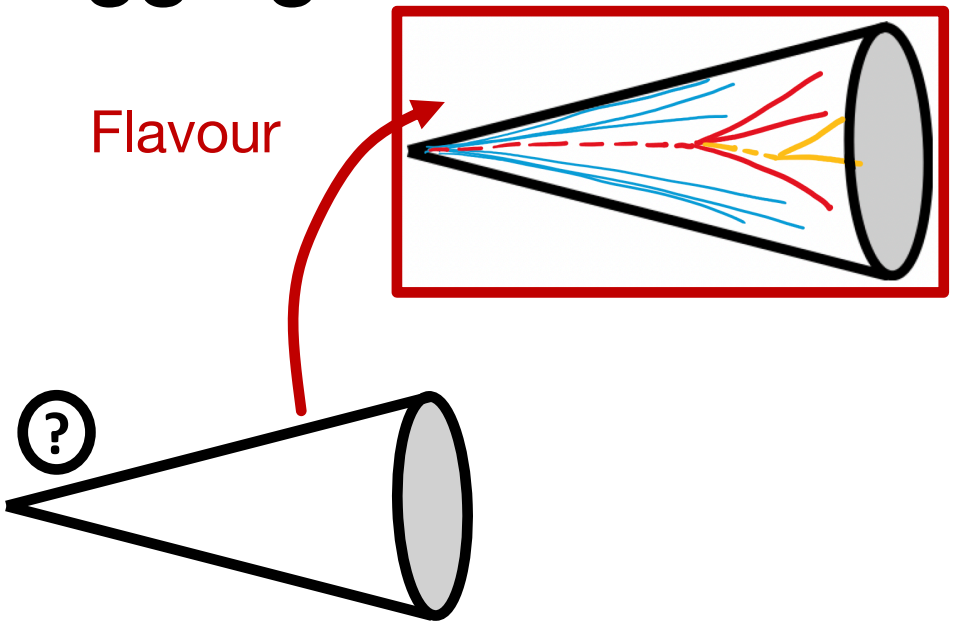
Exercise 1: Classify different particle types



Facts:
- O(50-100) particles per jet
- O (50) features per particle

~O(1000) inputs/jet

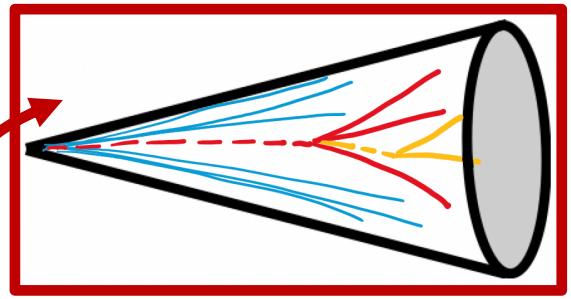
Jet tagging basics



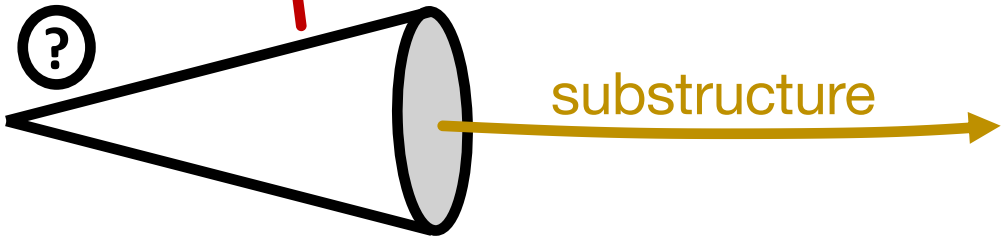
- Larger lifetime of b/c hadrons
- Displaced tracks/vertices
- Harder fragmentation
- Non-isolated leptons

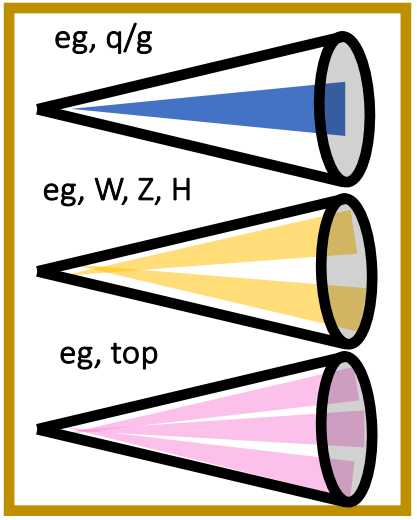
Jet tagging basics

Flavour



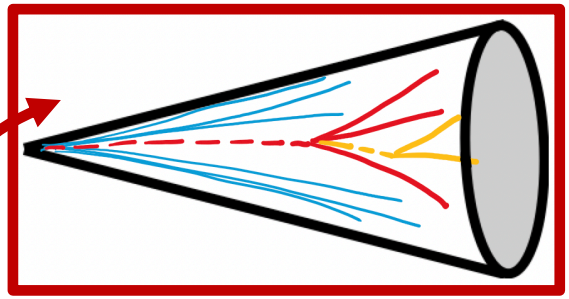
- Larger lifetime of b/c hadrons
- Displaced tracks/vertices
- Harder fragmentation
- Non-isolated leptons



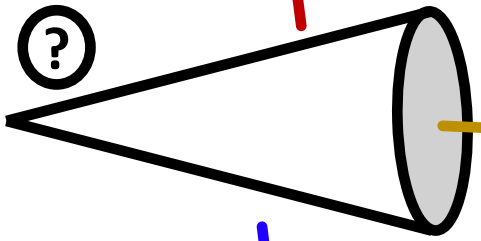


Jet tagging basics

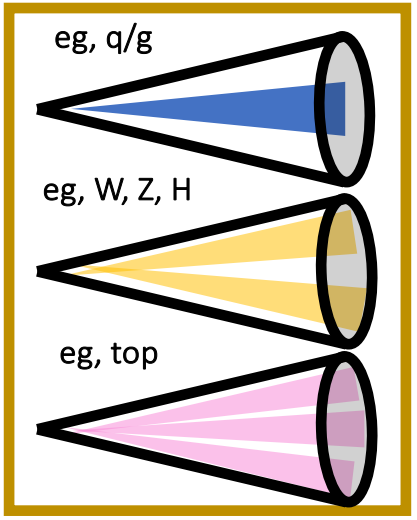
Flavour



- Larger lifetime of b/c hadrons
- Displaced tracks/vertices
- Harder fragmentation
- Non-isolated leptons



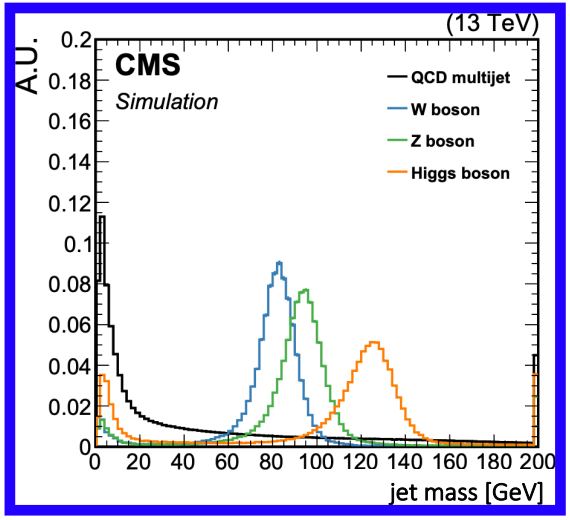
substructure



Jet mass

Light q/g: m(jet)→0 GeV

Higgs: m(jet)→125 GeV



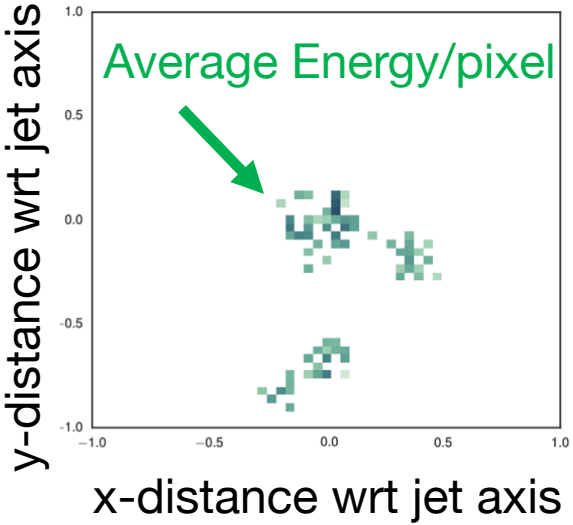
How to “represent” the jet

How to “represent” the jet

- Key for powerful AI/ML-based algorithms
  - We are going to compare two representations
    - **Jet as an Image:** Processe via Convolutional Neural Networks (CNNs)
    - **Jet as a Graph:** Process via GNNs
  - Compare them

Jets as images

Jet Image
[a top quark]

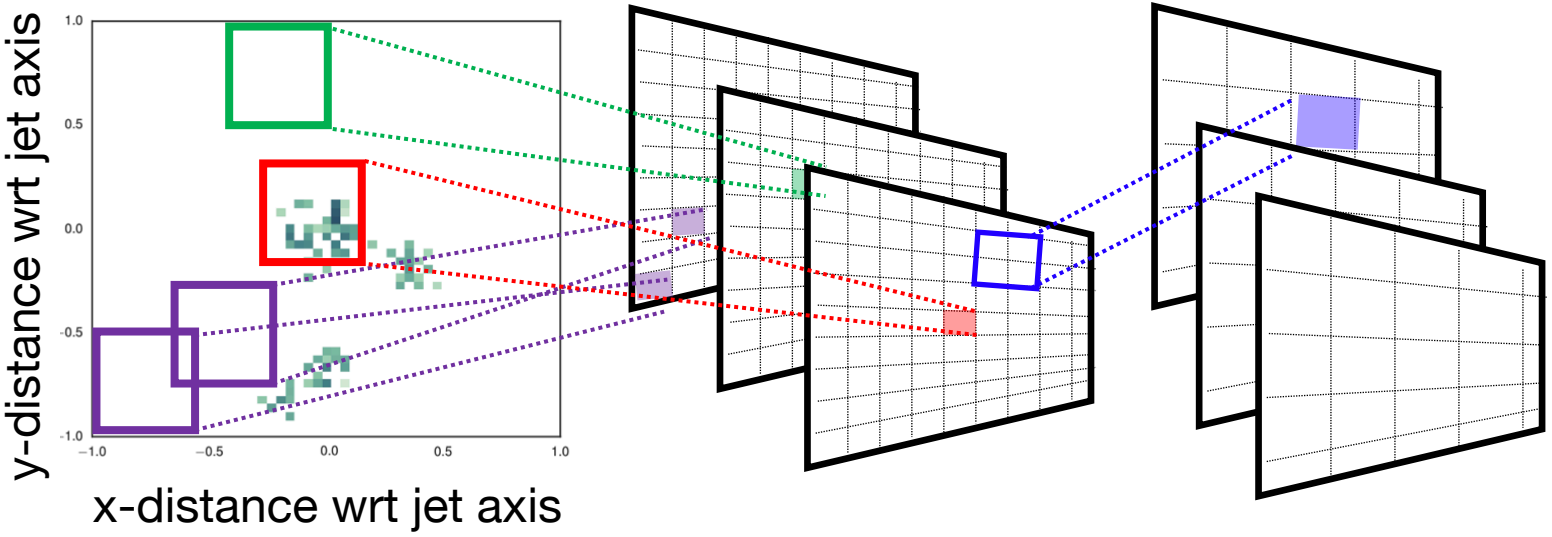


detector → camera
jet → image

Jets as images

Jet Image [a top quark]

Extract features



detector → camera
jet → image

2D-conv layers

- Convolution layers can be stacked
- Deeper layers exploit more general features

Pooling

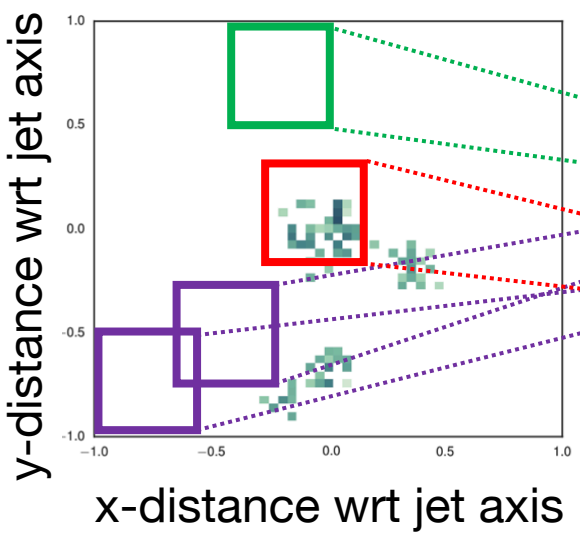
- Max, Min,… of an area [e.g., 2x2 pix] of the feature map
- Robustness
- Dimensionality reduction

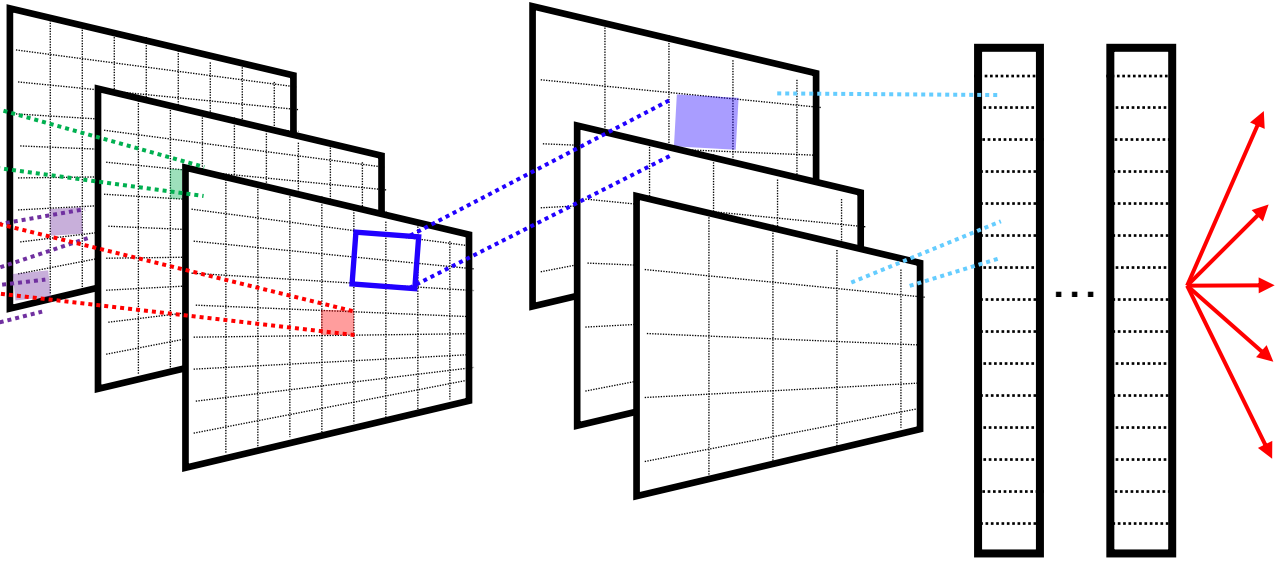
Jets as images

Jet Image [a top quark]

Extract features

Classification





detector → camera
jet → image

2D-conv layers
- Convolution layers can be stacked
- Deeper layers exploit more general features

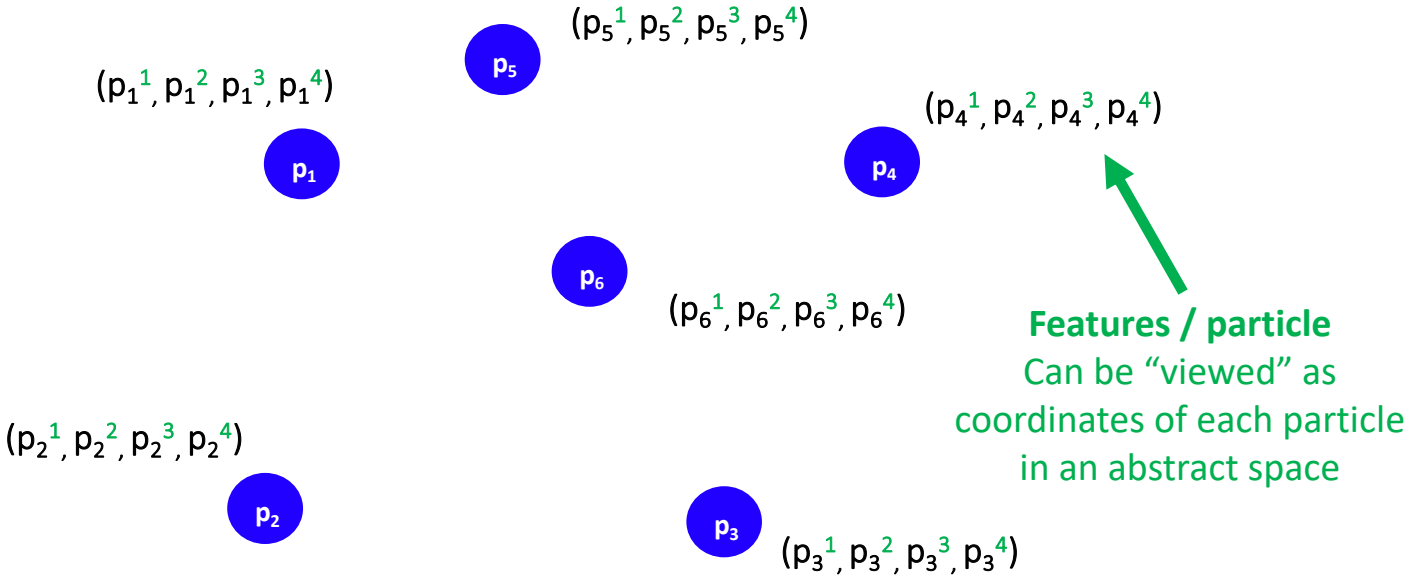
Pooling
- Max, Min,… of an area [e.g., 2x2 pix] of the feature map
- Robustness
- Dimensionality reduction

Dense NN
- Correlations & classification
- **Output**: binary or multiclass

Jet as a Graph

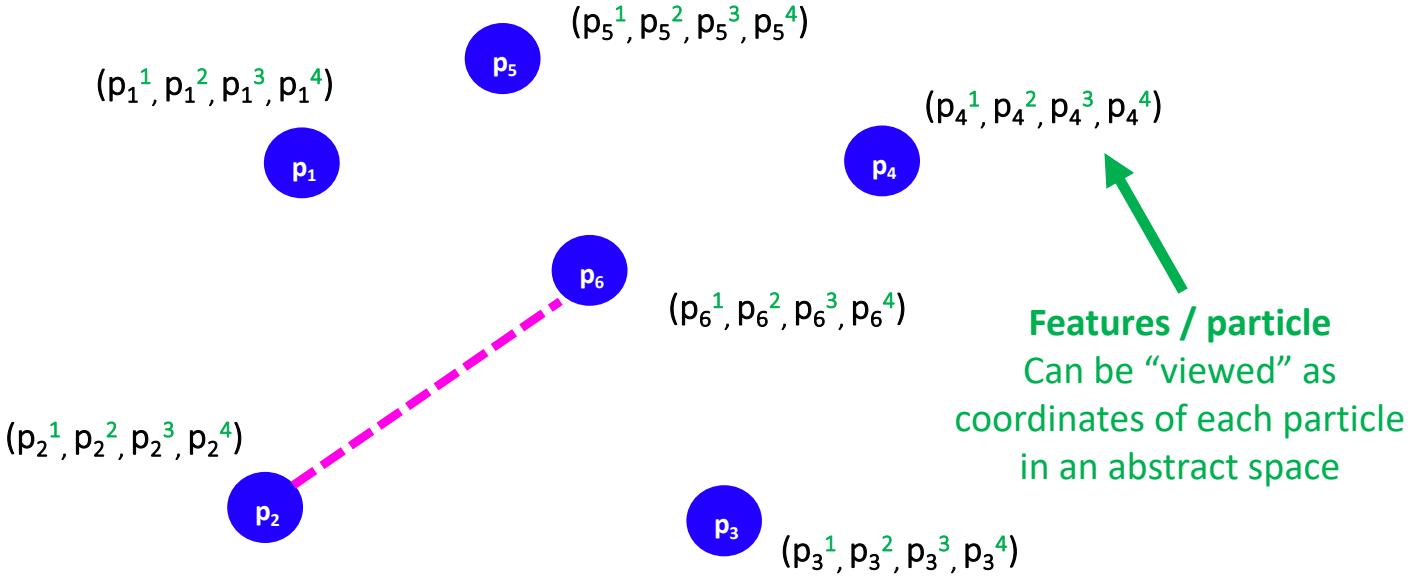
Jet as a Graph

- We form a **"Point/Particle Cloud"**
  - Each particle: **node** of the graph



Jet as a Graph

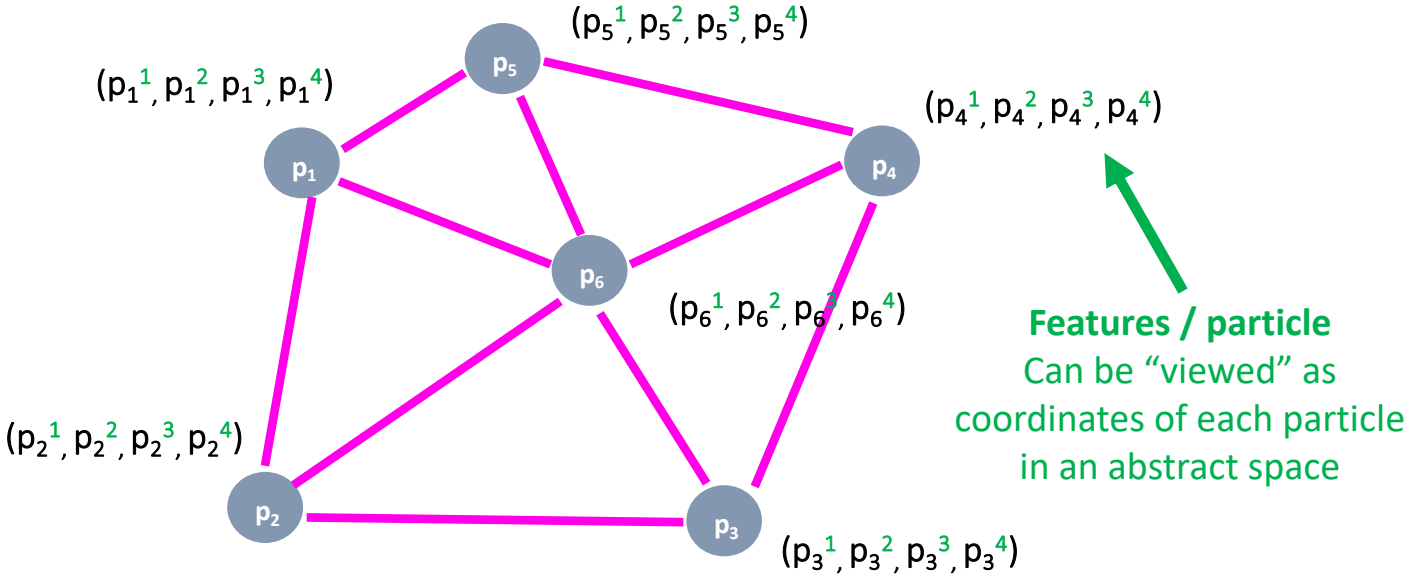
- We form a **"Point/Particle Cloud"**
  - Each particle: **node** of the graph



- Connections between particles: the **edges**

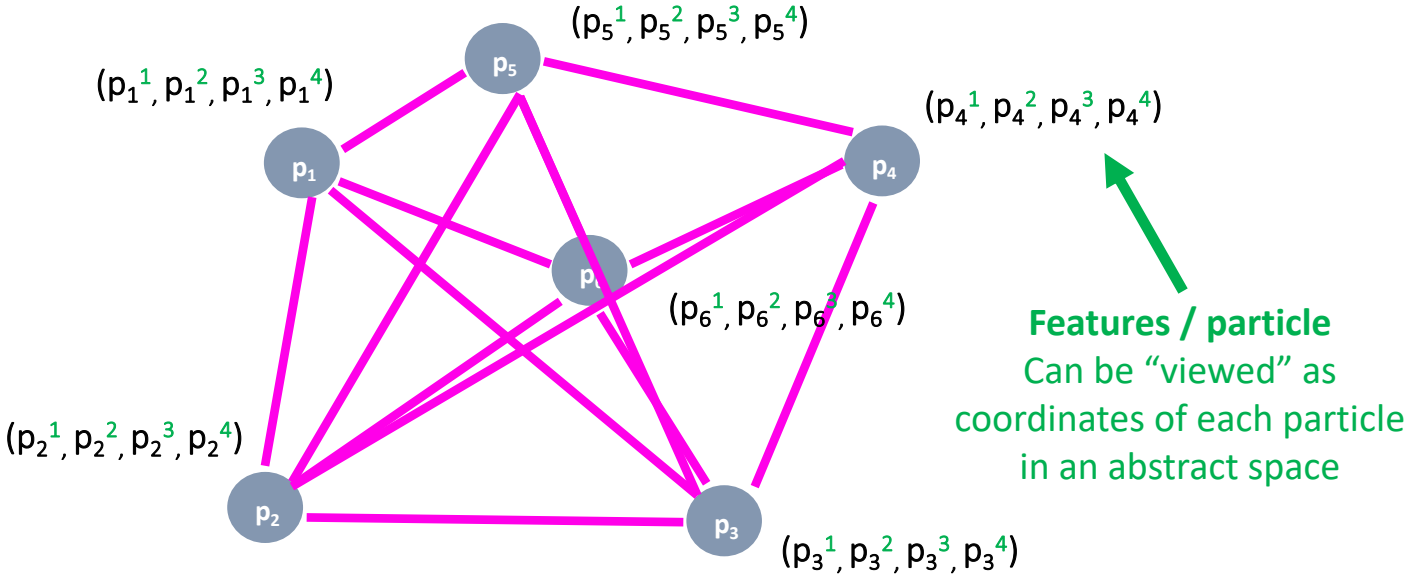
Step 1: Build the graph

- One approach: **Fully connected Graph**



Step 1: Build the graph

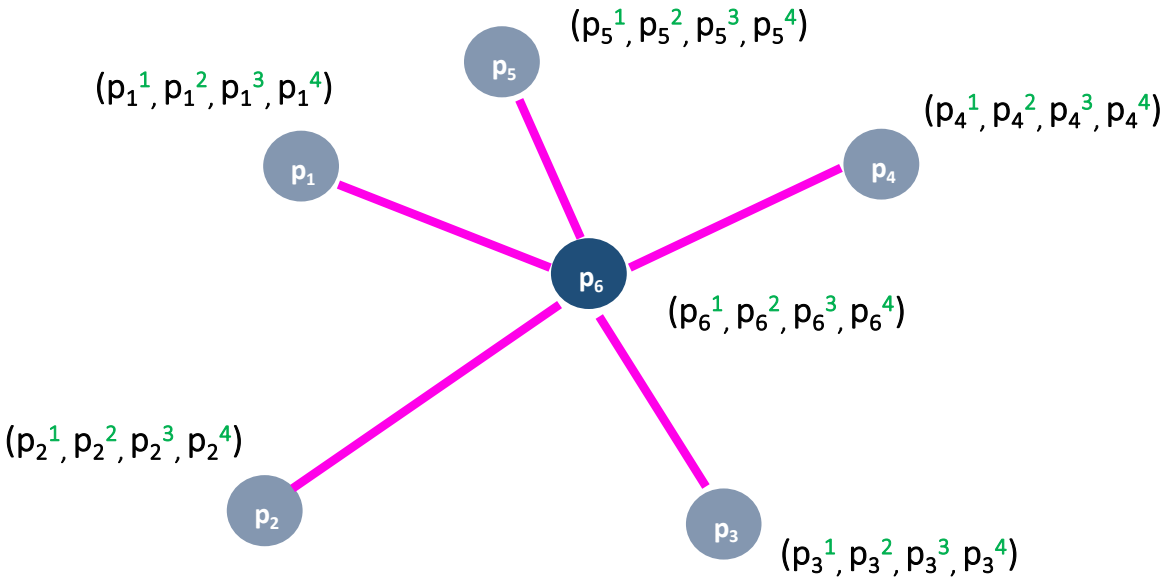
- One approach: **Fully connected Graph**



But computationally very expensive

Step 1: Build the graph

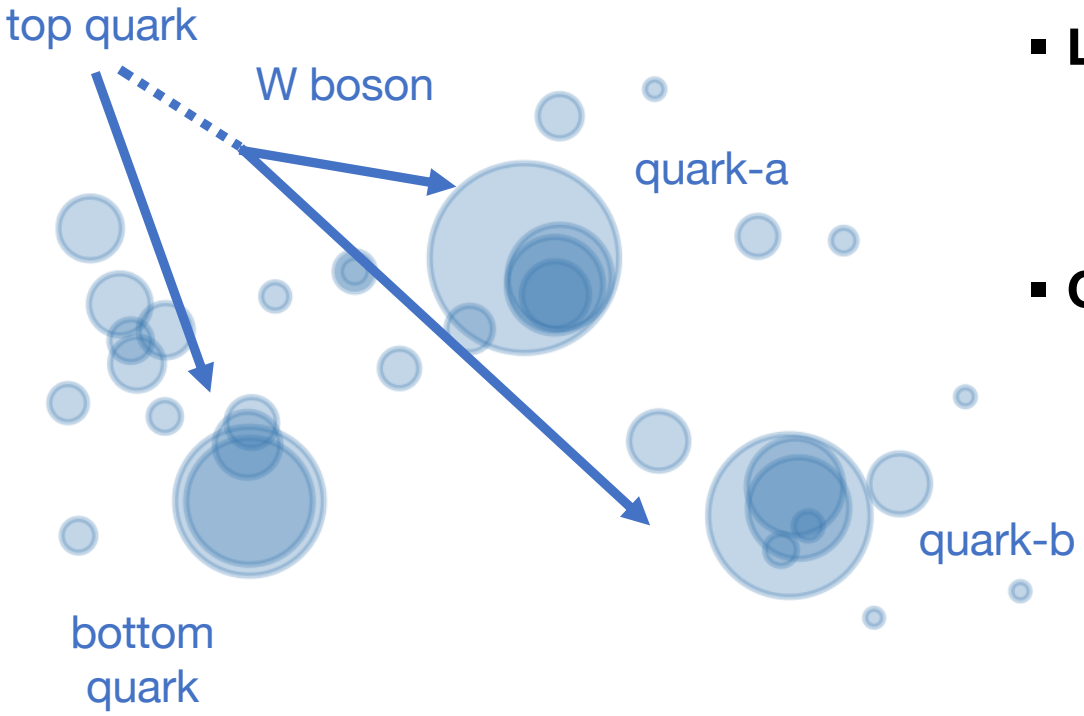
- One approach: **Fully connected Graph**
- **Another option:** apply some criteria
  - eg, select the k-Nearest Neighbors to each point, or within some distance, or



Step 2: Learn from the Graph

PRD 101 056019 (2020)

- Follow a **hierarchical learning** approach:
  - **First learn local structures** and **then more global** ones

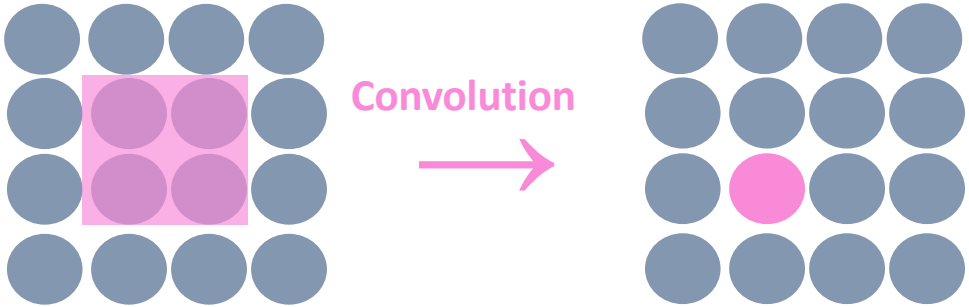


- **Local** structures:
  - Identify the bottom quark and the two quarks from the W-boson separately
- **Global** info:
  - Presence of two quarks stemming from W-boson decay
  - Three quarks within jet, etc..

Convolution operations proven to be very powerful

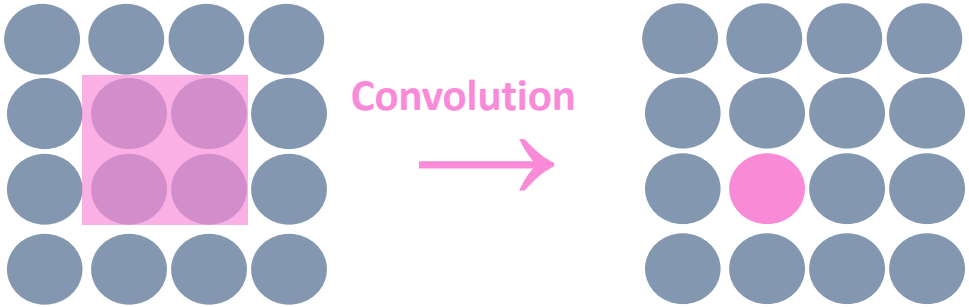
Convolution on fixed grids [CNN-2D]

Fixed grid:

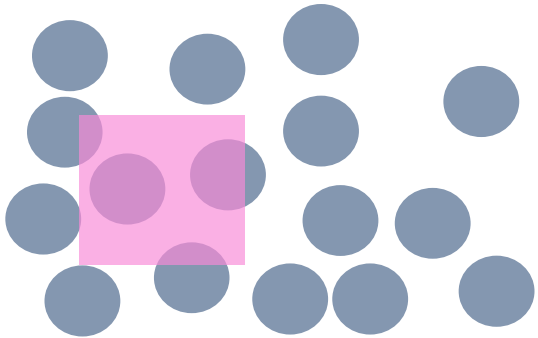


Convolution on Particle Clouds [?]

Fixed grid:



point/particle cloud:

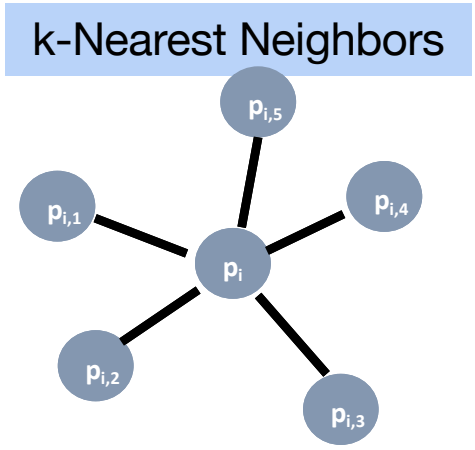


… but not straight forward on point/particle clouds
- Irregular and unordered sets
- Requires a permutation invariant convolutional operation

Convolution on particle clouds

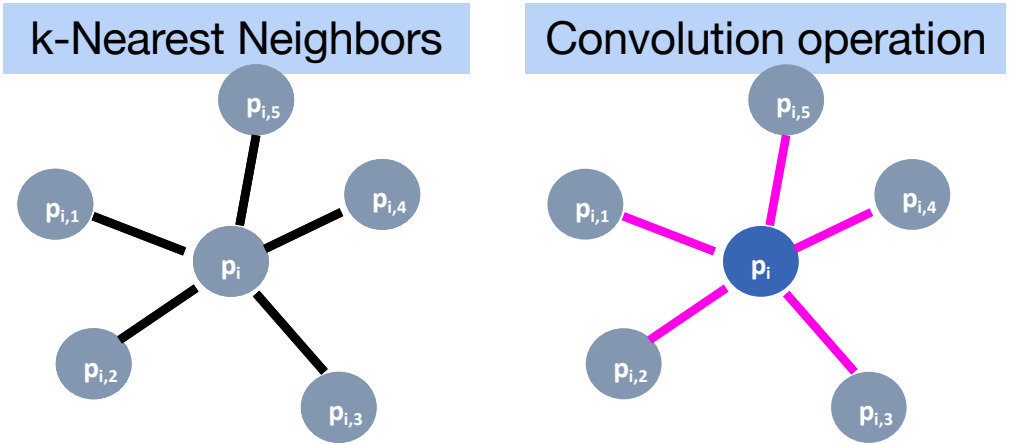
arxiv:1801.07829

- Find the ***k*-nearest neighbors** of each point



Convolution on particle clouds

- Find the ***k*-nearest neighbors** of each point
- Permutation invariant **convolution operation**
  - **Edge feature** func.→ **aggregate** edge feat. w/ symmetric func.



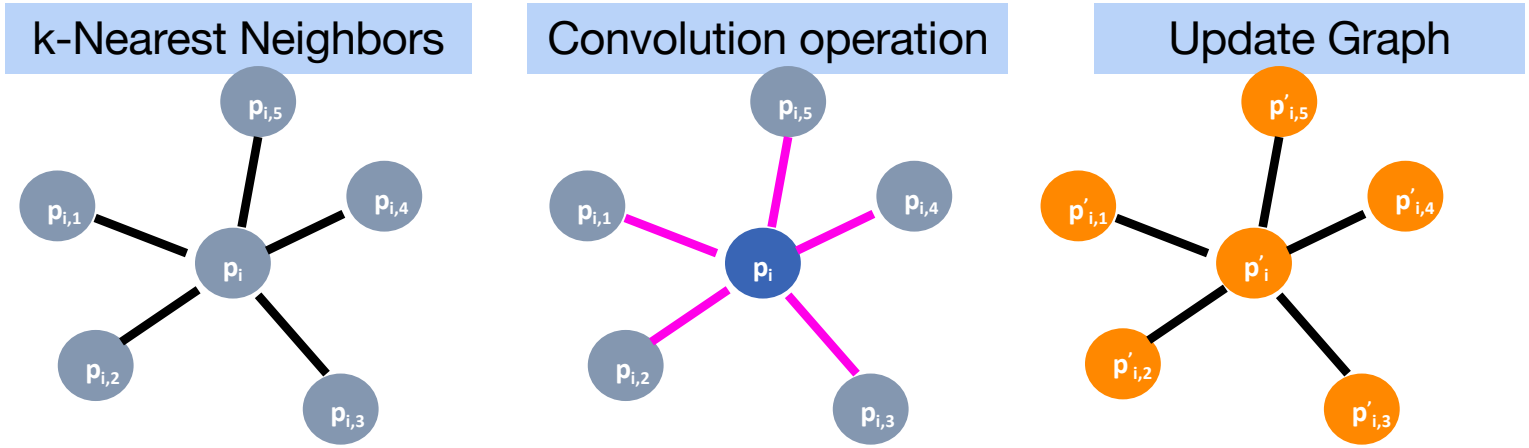
- In a nutshell:

$$p_i' = \square_{j=1}^{k} h_\theta(p_i, p_{ij} - p_i)$$

h_Θ: MLP [shared across edges]
: average over all *k*-NN

Convolution on particle clouds

- Find the ***k*-nearest neighbors** of each point
- Permutation invariant **convolution operation**
  - **Edge feature** func.→ **aggregate** edge feat. w/ symmetric func.
- **Update Graph (ie Dynamic Graph CNN, DGCNN):**
  - Using *k*NN in the feature space produced after convolution



- In a nutshell:

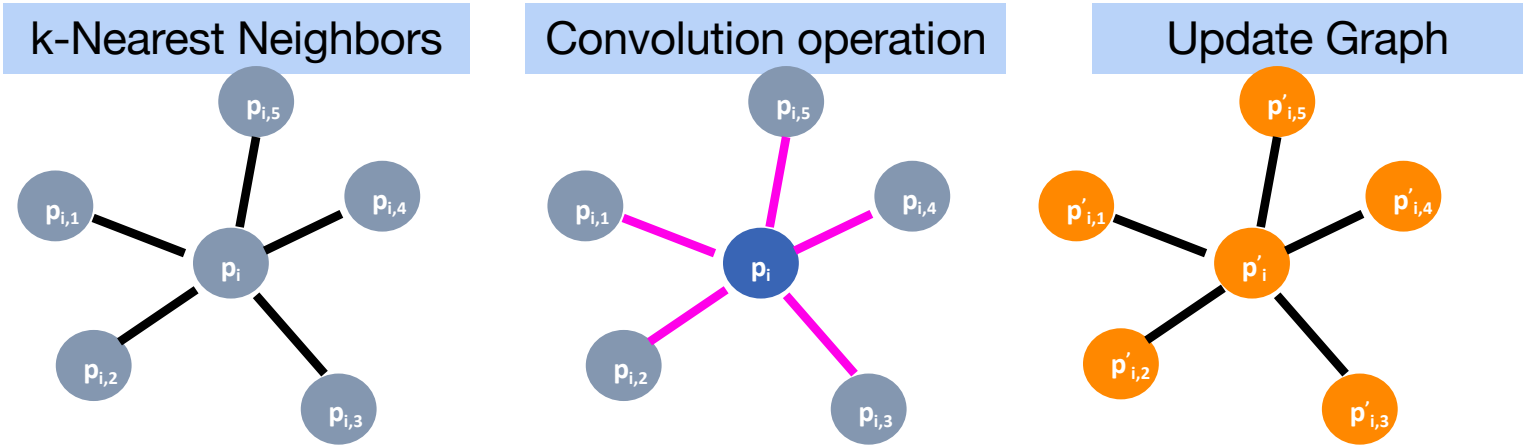
$$p_i' = \square_{j=1}^{k} h_\theta(p_i, p_{ij} - p_i)$$

h_Θ: MLP [shared across edges]
: average over all *k*-NN

Convolution on particle clouds

A simplified version during the hands-on session ☺

- Find the ***k*-nearest neighbors** of each point
- Permutation invariant **convolution operation**
  - **Edge feature** func.→ **aggregate** edge feat. w/ symmetric func.
- **Update Graph (ie Dynamic Graph CNN, DGCNN):**
  - Using *k*NN in the feature space produced after convolution



- In a nutshell:

$$p_i' = \square_{j=1}^{k} h_\theta(p_i, p_{ij} - p_i)$$

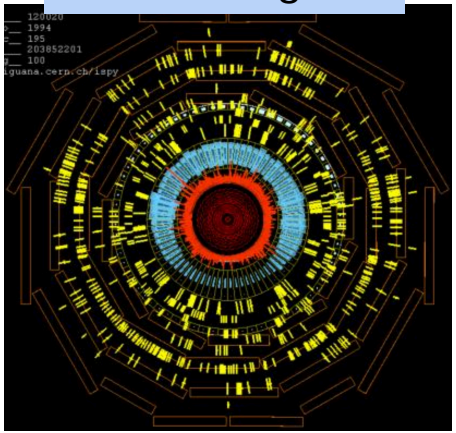
h_Θ: MLP [shared across edges]
: average over all *k*-NN

Exercise 1: Hands-on part

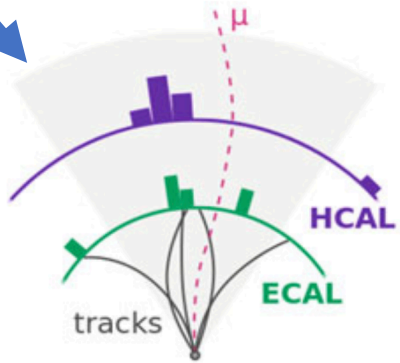
Exercise 2: Clustering

Particle/event reco chain

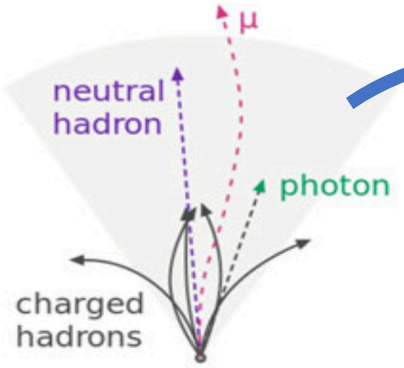
What we see:
Detector signals



Cluster signals in each sub-detector [Local RECO]

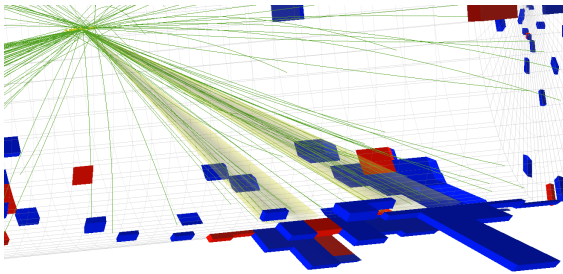


Combine info from different sub-detectors



Output: γ, e, μ, charged & neutral hadrons

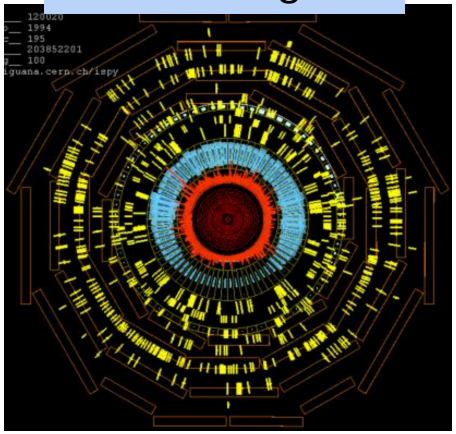
Target:
Jet flavour



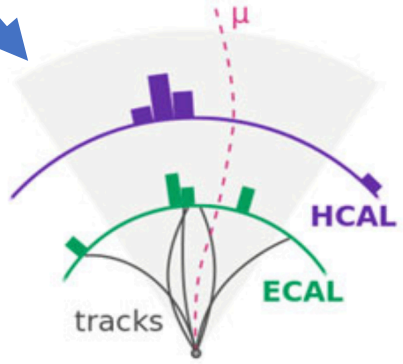
lower-level ⟶ **higher-level**

Particle/event reco chain

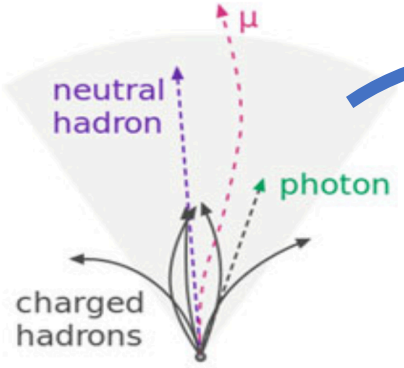
What we see:
Detector signals



Cluster signals in each sub-detector [Local RECO]

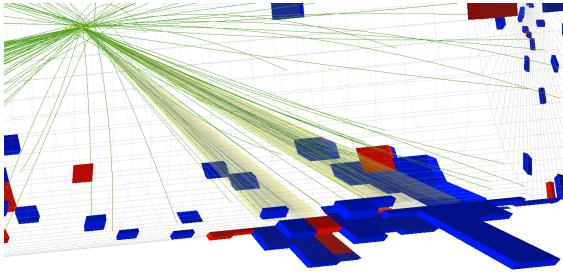


Combine info from different sub-detectors



Output: γ, e, μ, charged & neutral hadrons

Target:
Jet flavour



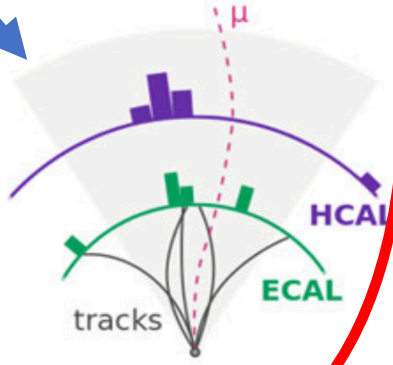
Exercise 1

lower-level ⟶ **higher-level**

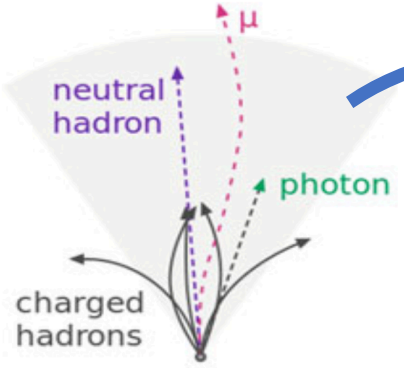
Particle/event reco chain

What we see:
Detector signals

Cluster signals in each sub-detector [Local RECO]

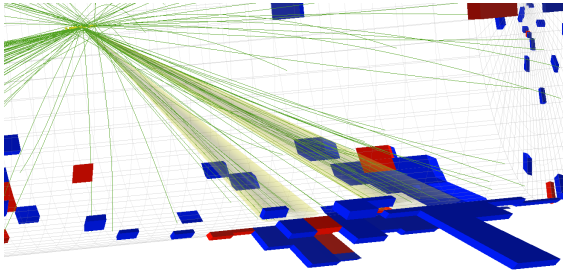


Combine info from different sub-detectors



Output: γ, e, μ, charged & neutral hadrons

Target:
Jet flavour



Exercise 1

Exercise 2

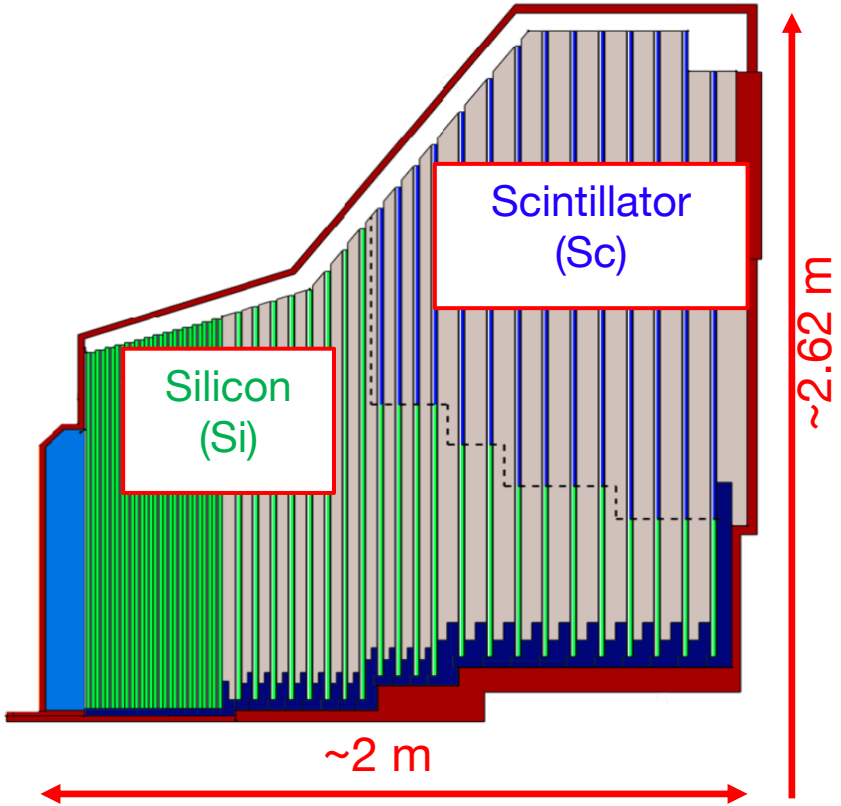
lower-level → higher-level

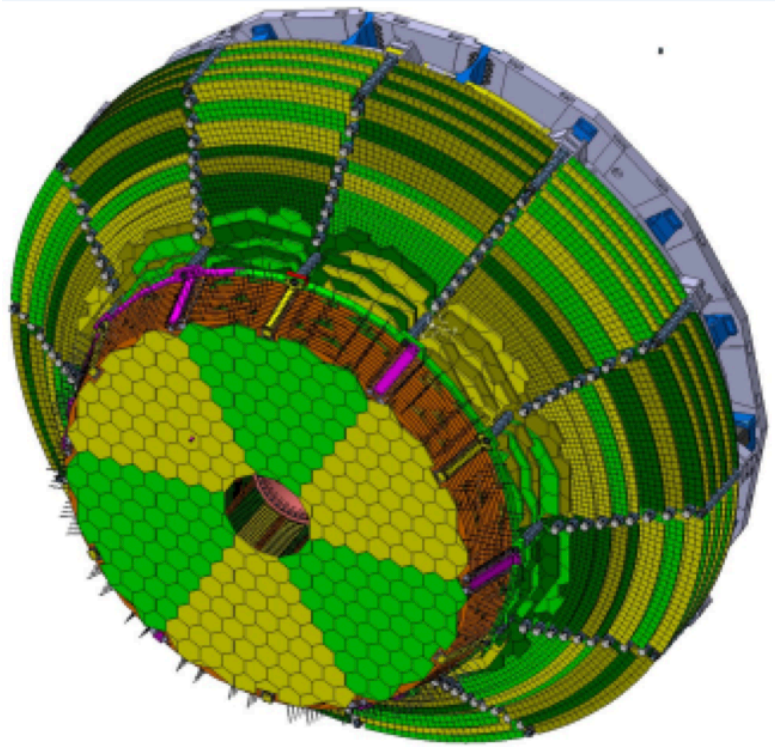
Exercise 2: Clustering

- Explore AI to cluster/group signals (e.g., energy deposits) in the detector
  - A particle traversing the detector give rise to signal in several channels
  - We aim to cluster deposits corresponding to each particle

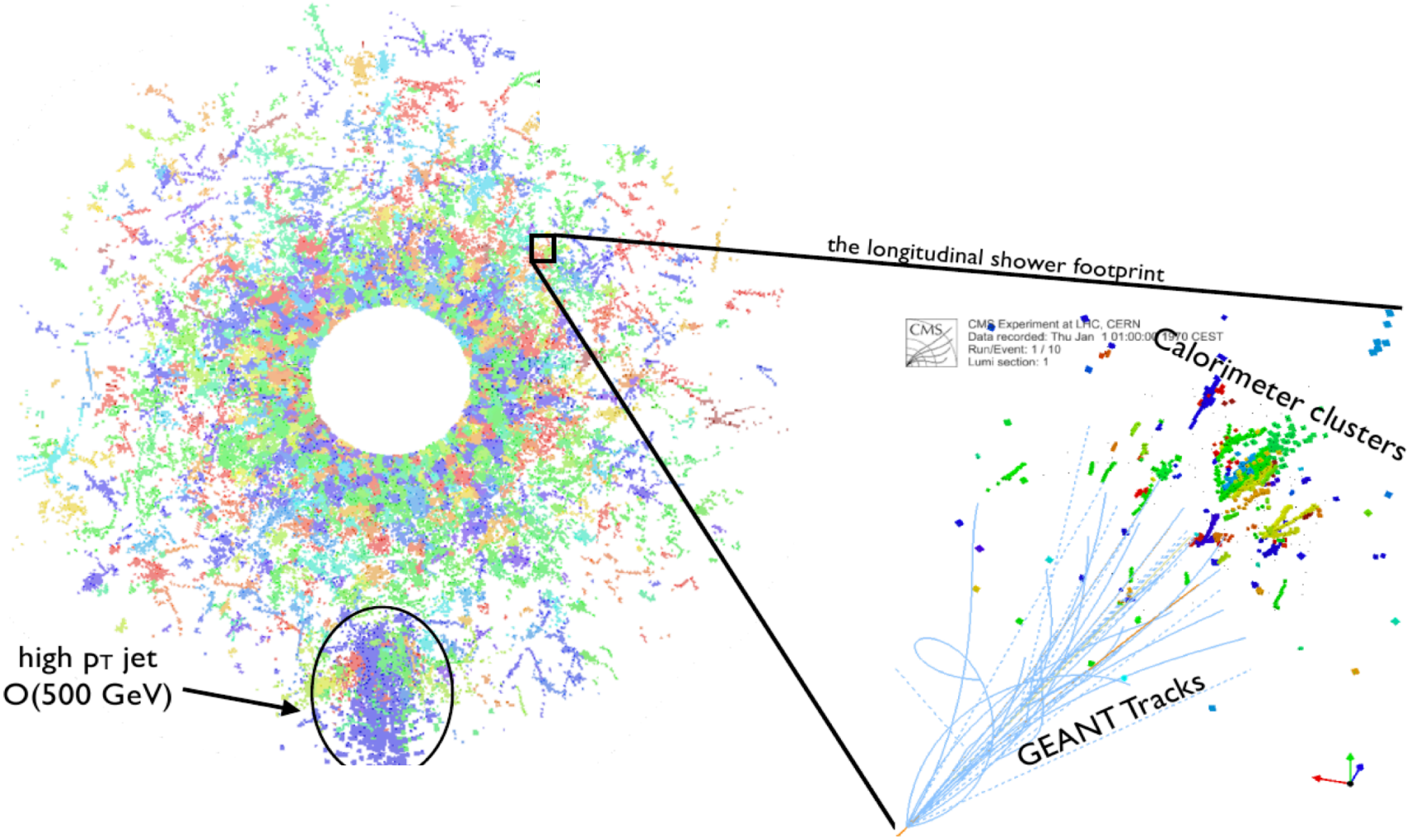
Exercise 2: High Granularity Calorimeter

- Use a novel and quite ambitious detector project
  - The CMS High Granularity Endcap Calorimeter
    - Currently under construction; Target: 2030s





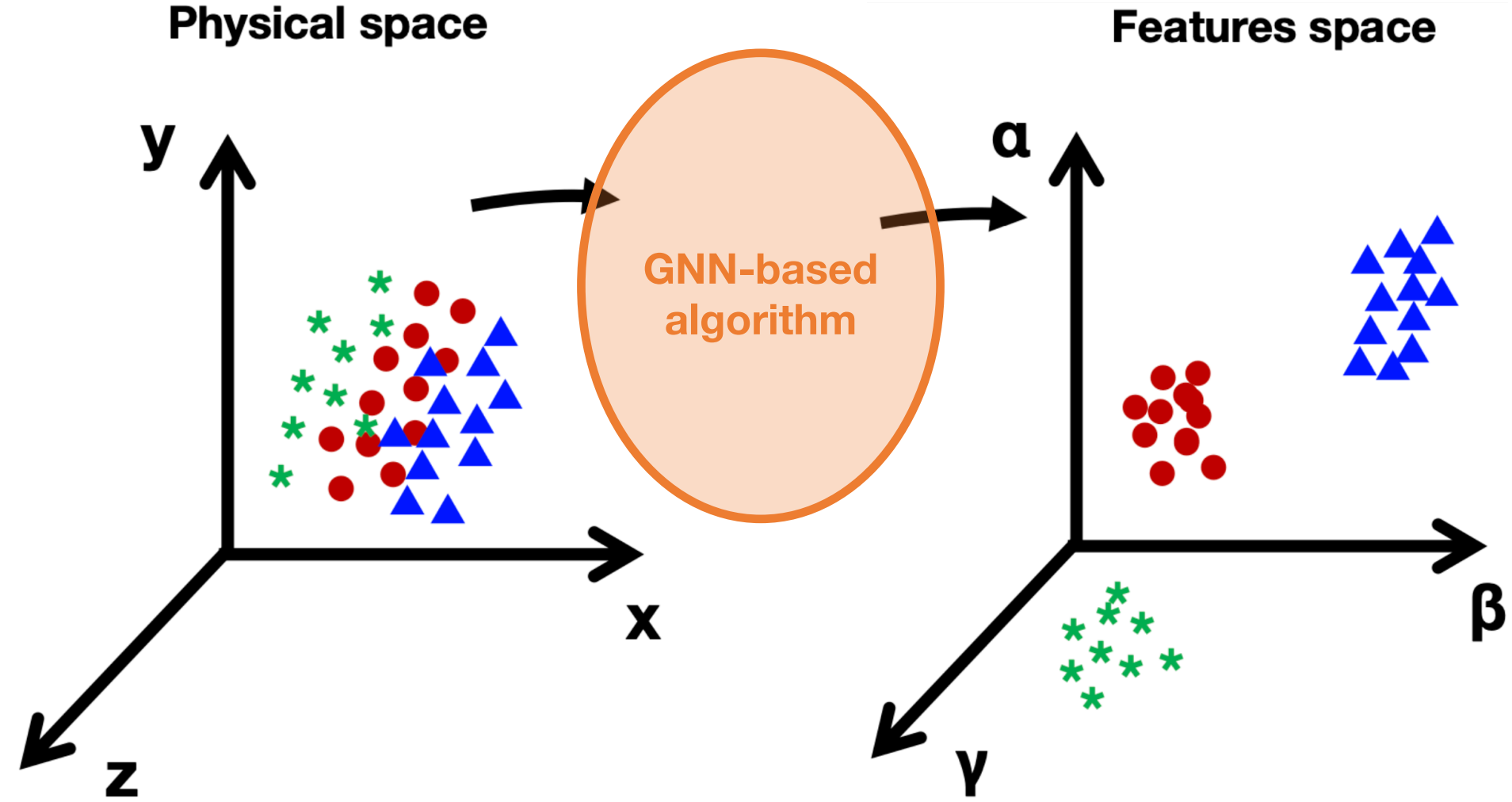
Simulated event



Clustering energy deposits

- Huge potential but extremely challenging using “traditional” (i.e., not ML) algorithms
  - each signal has a set of features: Energy, position, timing…
    - using these “raw” information is difficult to separate the deposits from different particles (i.e., different colors in the previous plot)
- GNN-based approach
  - Extract more discriminating set of features
    - in the latent/abstract space
  - Group them using a clustering algorithm
    - For simplicity: use k-Means in this exercise

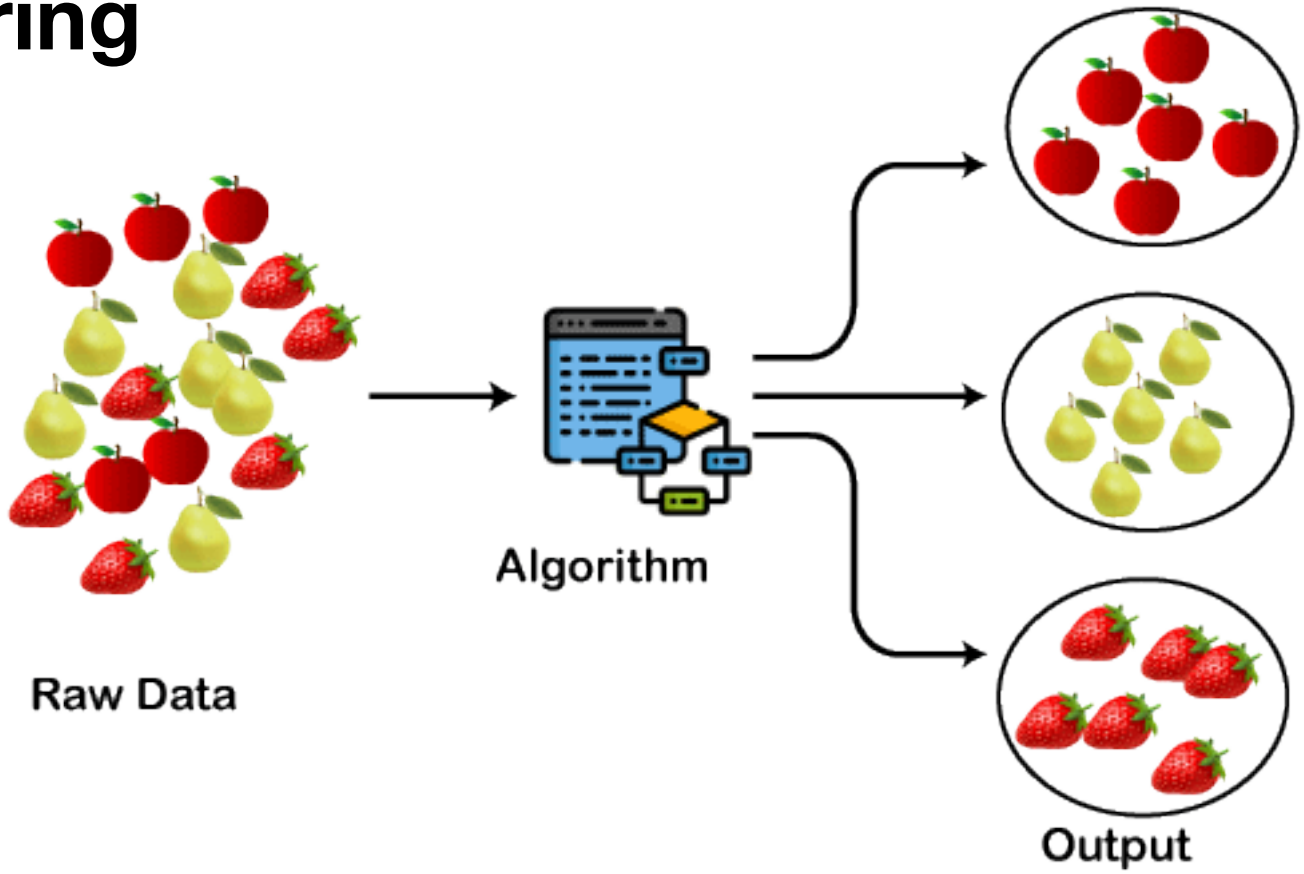
Clustering energy deposits



Clustering

- **Unsupervised** (or weakly-/self-supervised) learning method
  - Divides a set of points/entities/elements in groups with similar properties
- **Goal:** identify inherent structures/patterns within a dataset without using pre-labeled classes (i.e., no supervision).
  - Relies on a **similarity** measure
    - depends on the context and the data
- Important:
  - Since it is un-supervised learning we cannot use criteria such as accuracy, AUC, ..
    - challenging and highly subjective

Clustering



source

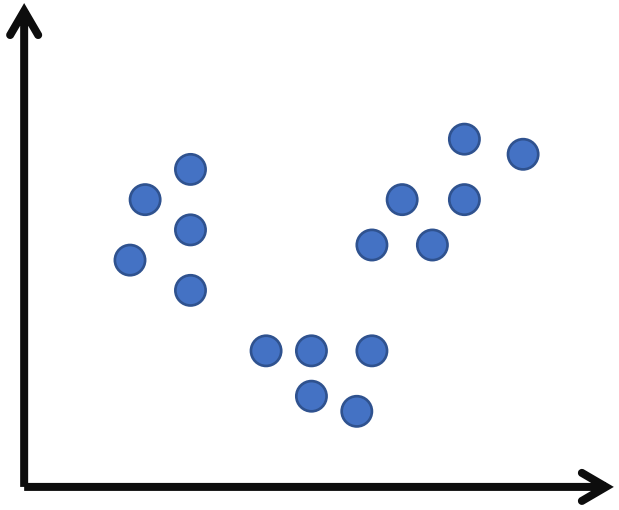
NB: No labelling [or very little]

Clustering algorithms aim to find patterns without us explicitly pointing them out.

Partitioning methods: *k*-Means

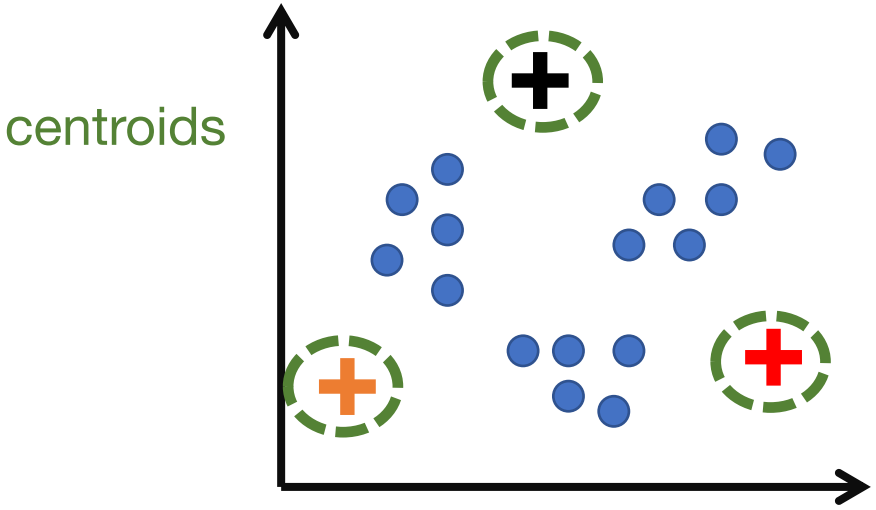
- Partition ***n* observations** into ***k*** clusters
  - each observation belongs to the cluster with the nearest mean
  - minimize an **objective function**

k-Means steps



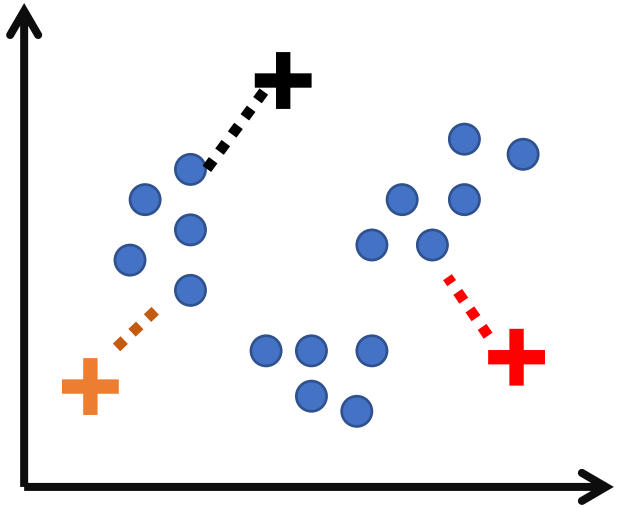
k-Means steps (I)

- **Initialization**: Select ***k*** initial centroids [randomly or not]



k-Means steps (II)

- **Assignment**: Assign each data point to the nearest centroid, forming k-clusters:

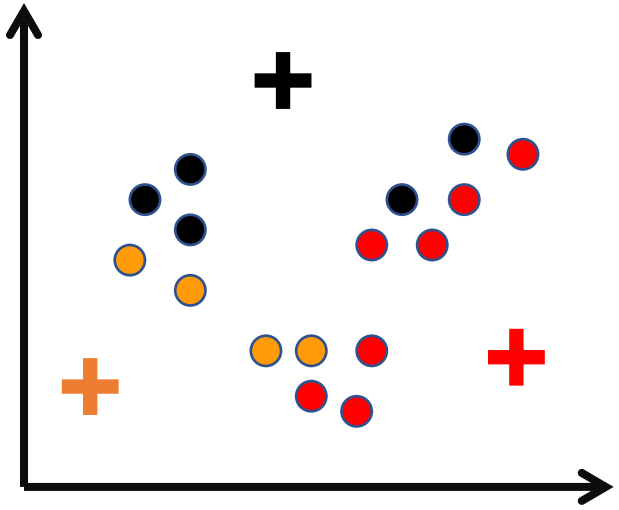


$$w_{ij} = \begin{cases} 1 & \text{if } j = \arg\min_{j'} ||x_i - \mu_{j'}||^2 \\ 0 & \text{otherwise} \end{cases}$$

(Euclidian distance)^2

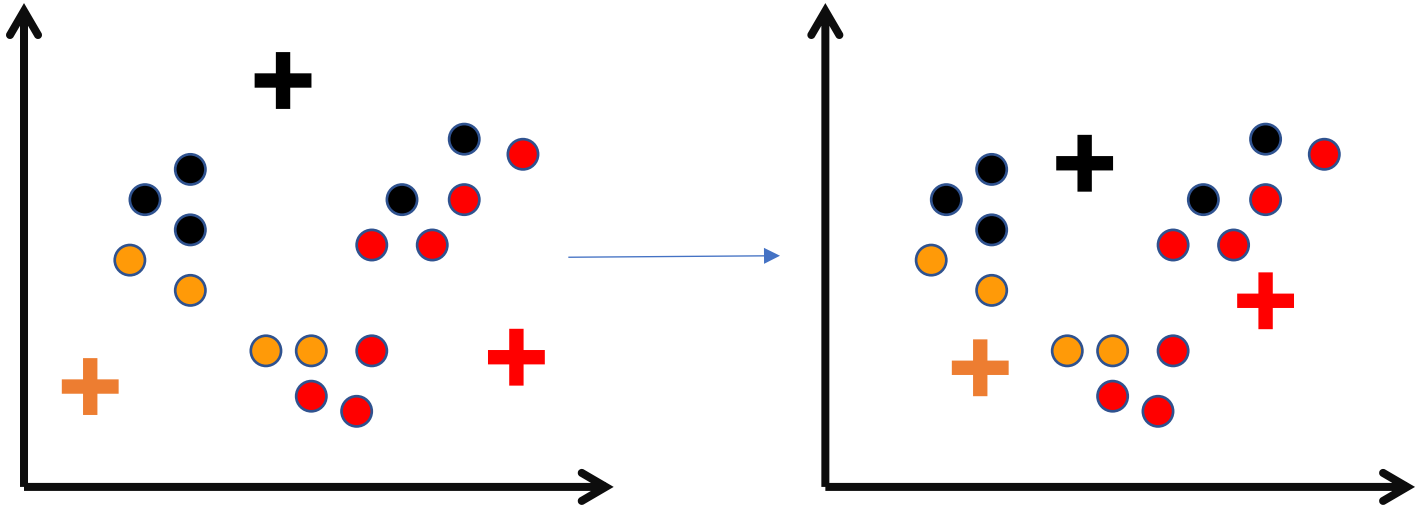
k-Means steps (III)

- **Update**: Recompute the centroid of each cluster as the mean of all points assigned to that cluster



k-Means steps (III)

- **Update**: Recompute the centroid of each cluster as the mean of all points assigned to that cluster



$$\mu_j = \frac{\sum_{i=1}^{n} w_{ij} x_i}{\sum_{i=1}^{n} w_{ij}}$$

k-Means steps (IV)

- **Repeat**: Assignment and Update steps repeated until centroids do not change significantly
  - minimize **objective function**:

$$J = \sum_{i=1}^{n} \sum_{j=1}^{k} w_{ij} ||x_i - \mu_j||^2$$

1: if x_i assigned to cluster j
0: otherwise

k-Means steps (IV)

- **Repeat**: Assignment and Update steps repeated until centroids do not change significantly
  - minimize **objective function**:

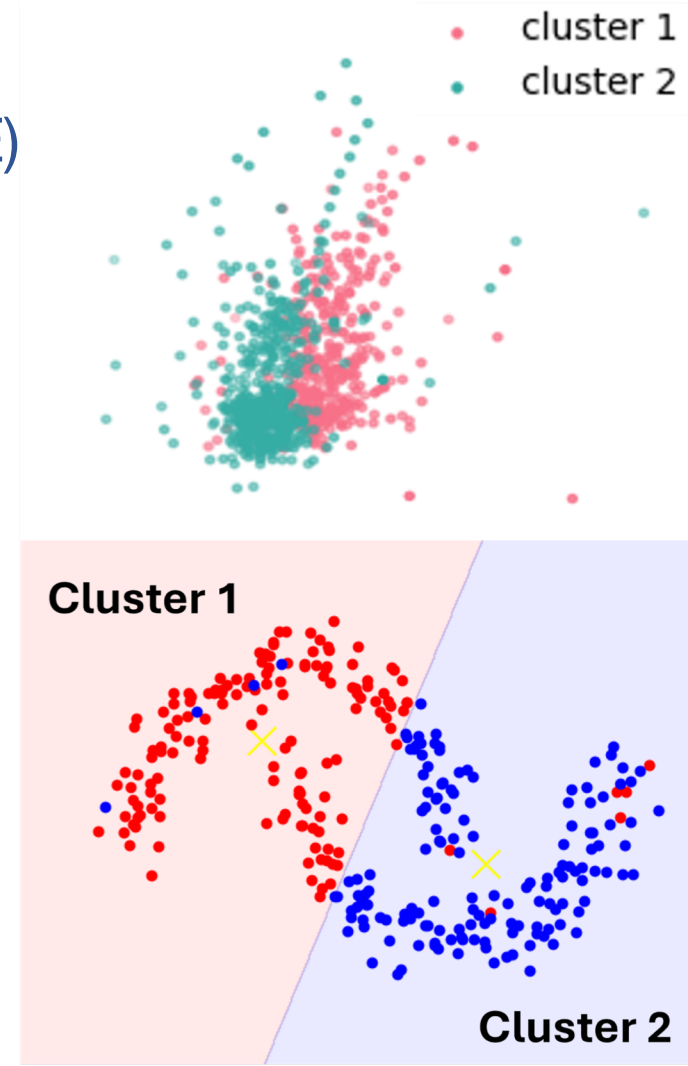
$$J = \sum_{i=1}^{n} \sum_{j=1}^{k} w_{ij} ||x_i - \mu_j||^2$$

1: if x_i assigned to cluster j
0: otherwise

- Procedure stops when centroids no longer change significantly from one iteration to the next
  - algorithm has converged and objective function reached a local minimum

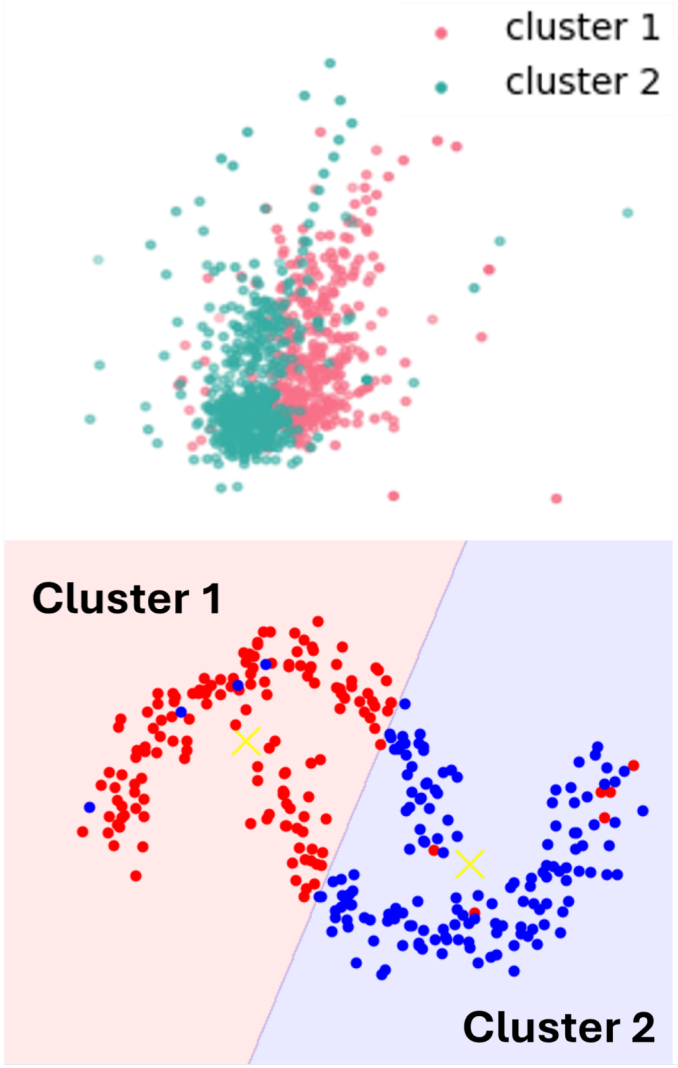
Hands-on 2: Algorithm Structure

- First:
  - Transform Input features (x,y,z,E) to new space through a GNN-based model
- Then:
  - Cluster with K-Means on model output



Hands-on 2: Alignment Diffusion Loss

- Minimize angle b/w connected vectors in the model's output
- Spread away angles with unconnected vectors



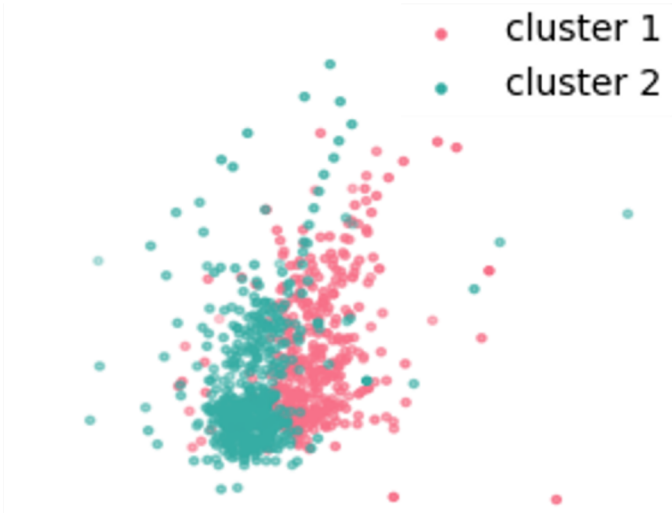
Hands-on 2: Metrics

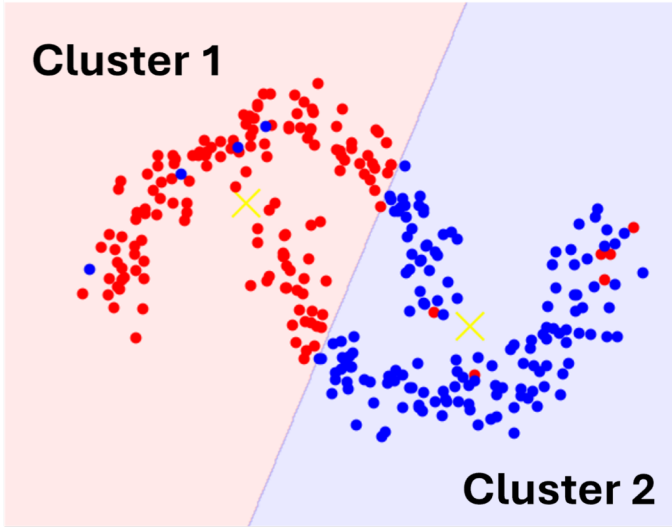
Evaluate how good the clusters are

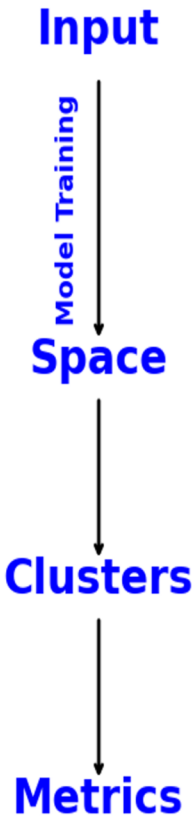
- Purity
- Energy-weighted purity

$$p = \frac{\text{predicted}}{\text{actual}} \times 100\%$$

$$ewp = \frac{E_{\text{correct}}}{E_{\text{actual}}} \times 100\%$$







Exercise 2: hands-on