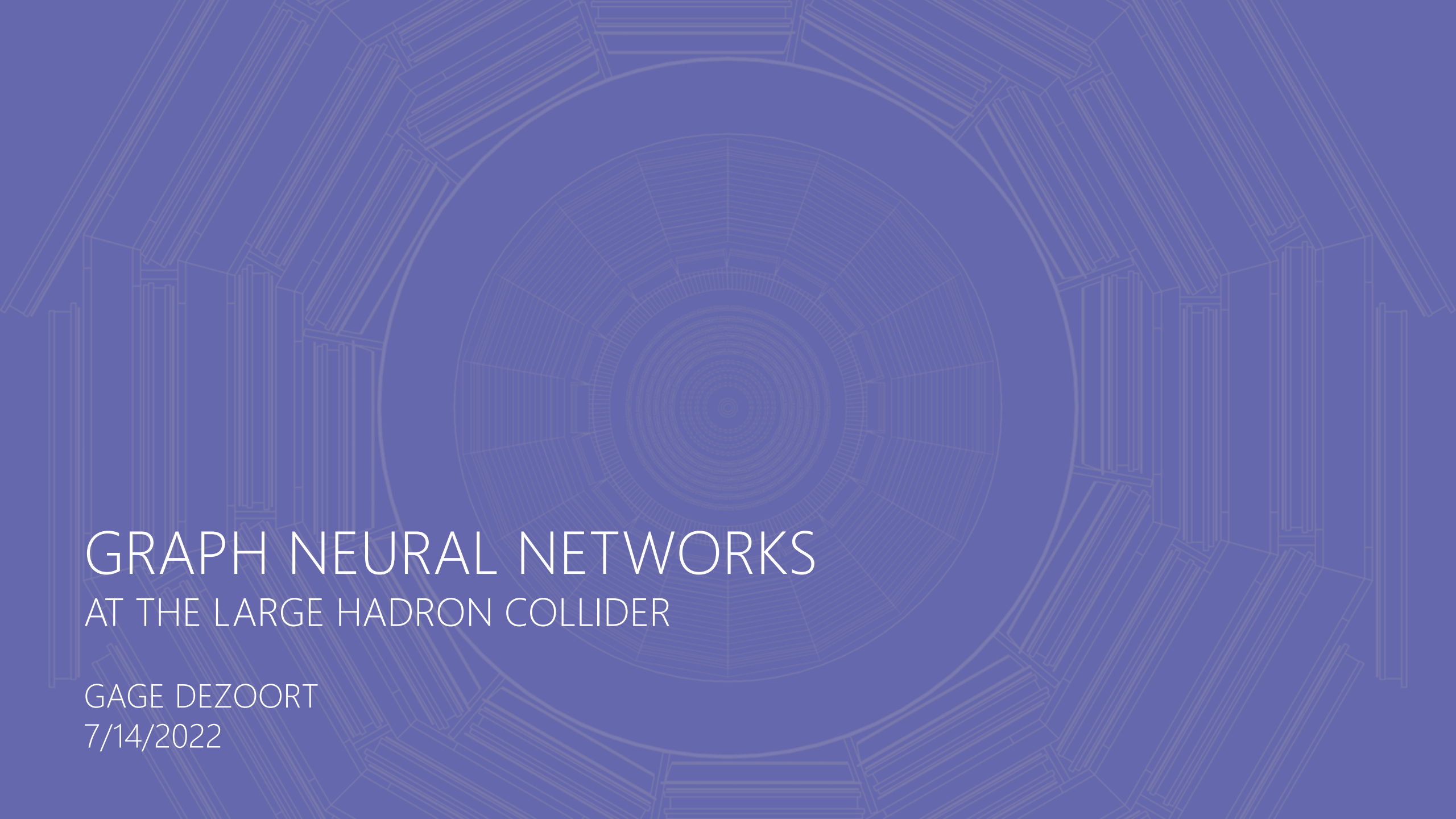


GRAPH NEURAL NETWORKS
AT THE LARGE HADRON COLLIDER

GAGE DEZOORT
7/14/2022

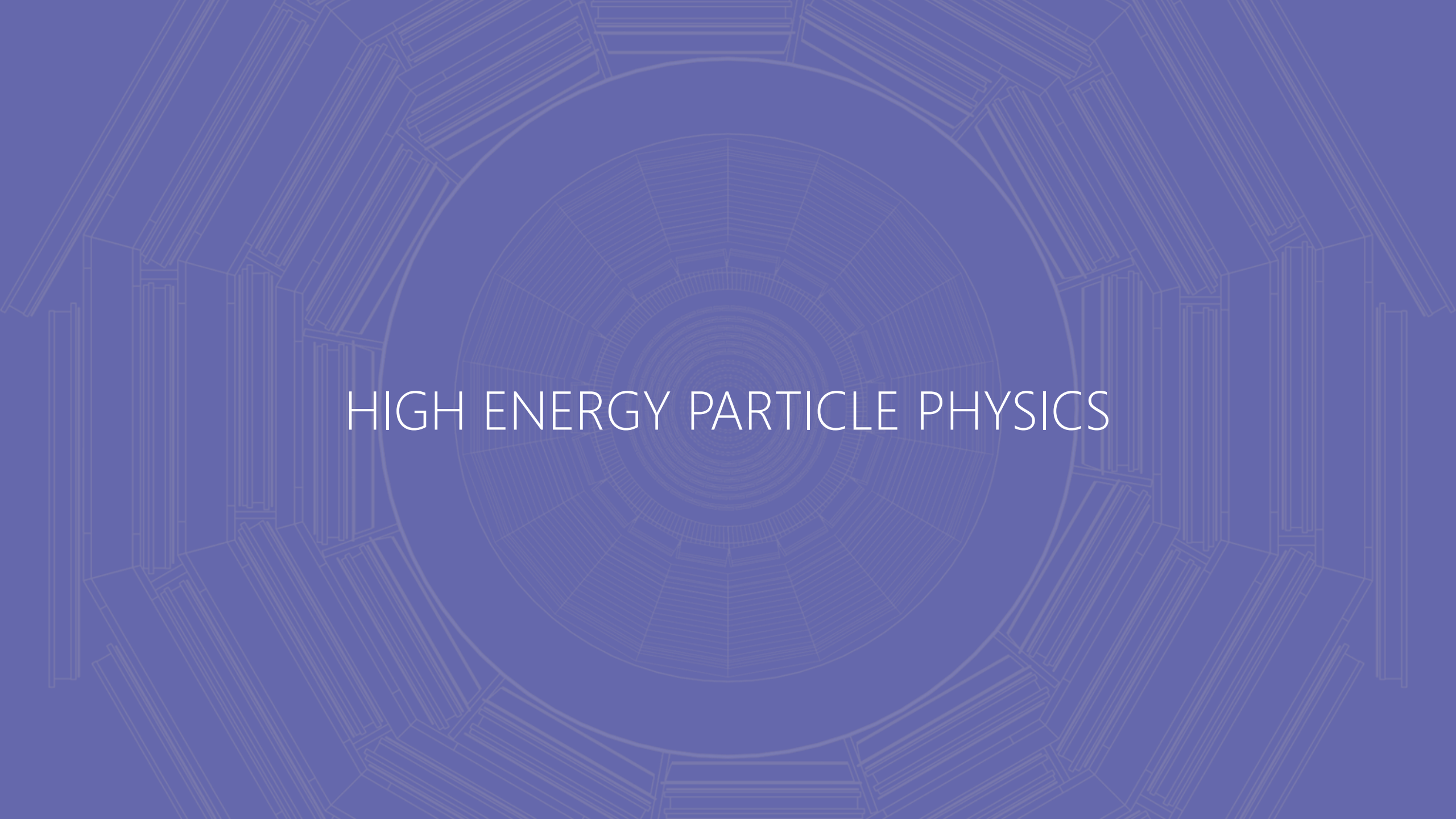
- **About me**: I'm a fourth-year Ph.D. student in the Physics department who has developed a strong interest in geometric deep learning methods applied to physics tasks

- High energy particle physics (HEP)
  - The Standard Model (SM) and beyond
  - Large Hadron Collider (LHC) physics
  - Detectors and measurements
- Graph-based learning methods
  - Graph-structured data
  - Graph neural networks (GNNs)
  - Common architectures
- GNNs at the LHC
  - Reconstruction (clustering)
  - Identification (classification)
- Conclusions

- Please ask questions; feel free to interrupt!

GNNS AT THE LHC

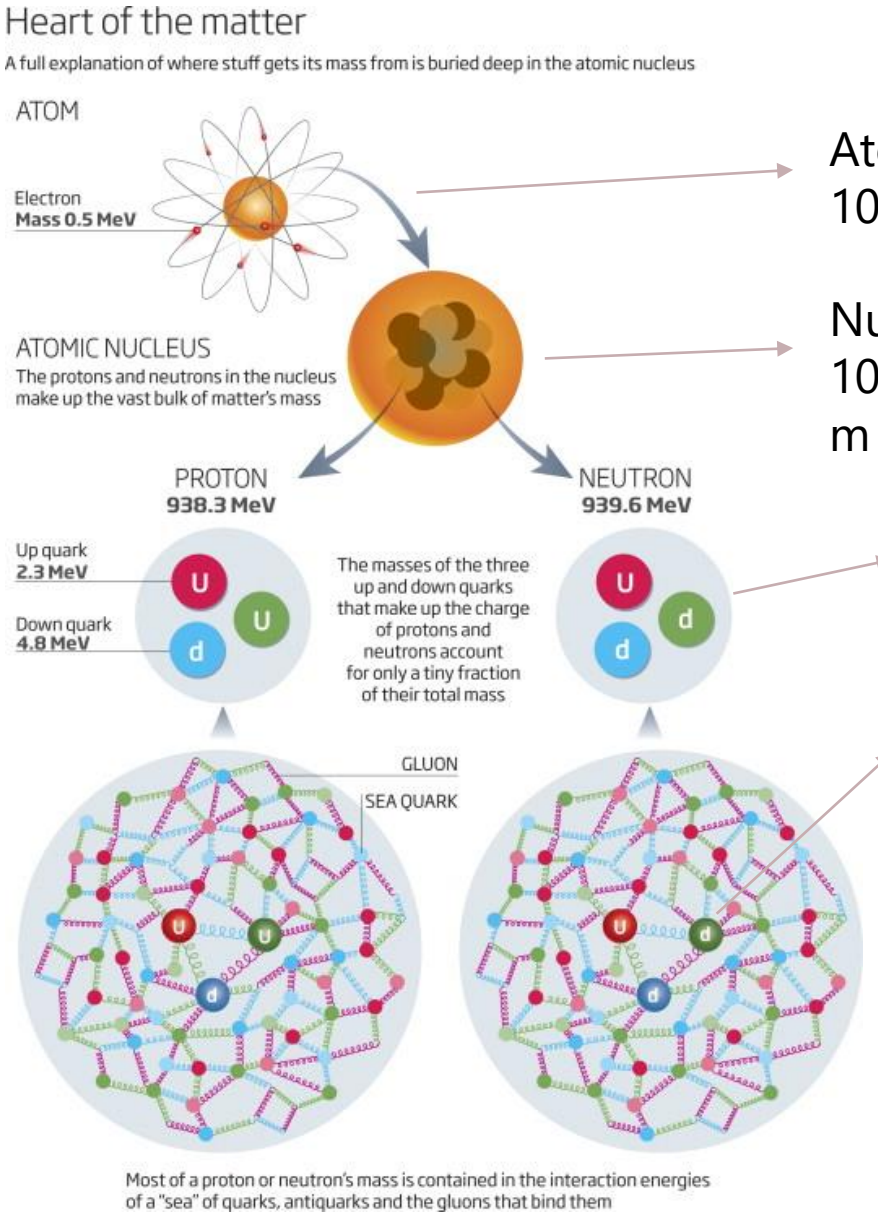
INTROS AND OVERVIEW



HIGH ENERGY PARTICLE PHYSICS

HIGH ENERGY PARTICLE PHYSICS

THE STANDARD MODEL (SM)



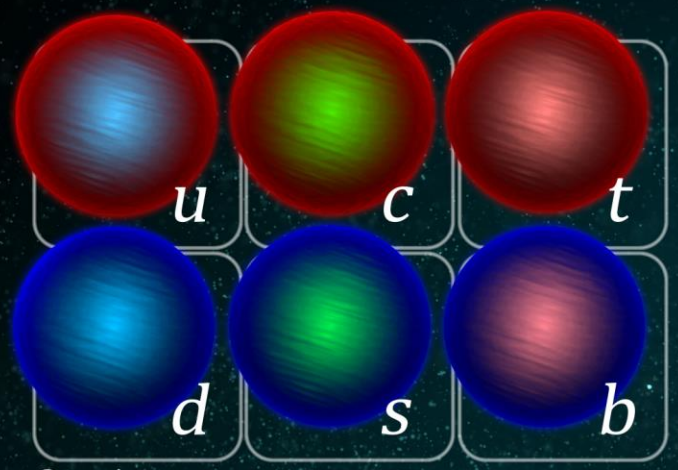
Atom:
10^{-10} m

Nuclei:
$10^{-15} – 10^{-14}$ m

Nucleon:
$<10^{-15}$ m

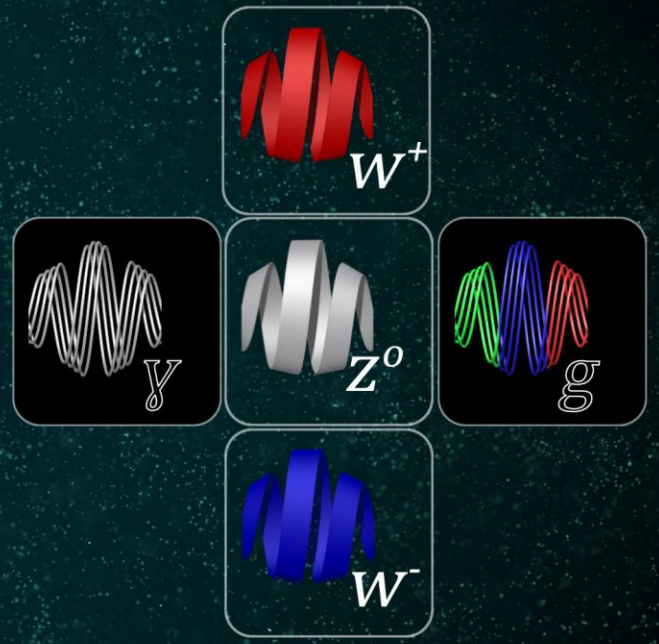
Quark:
$<10^{-19}$ m

No observed spatial structure → elementary particle!

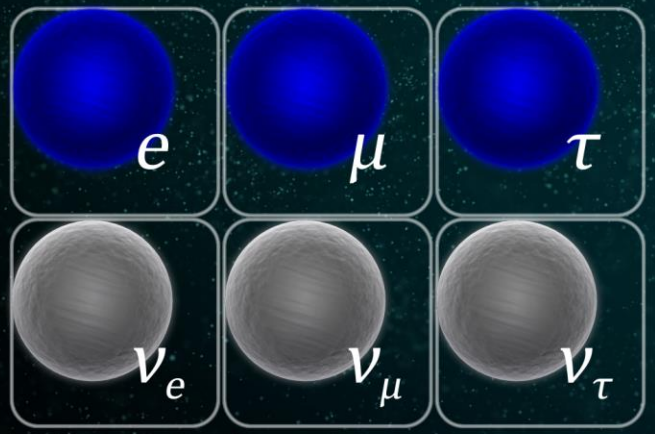


Quarks

Higgs boson



Forces

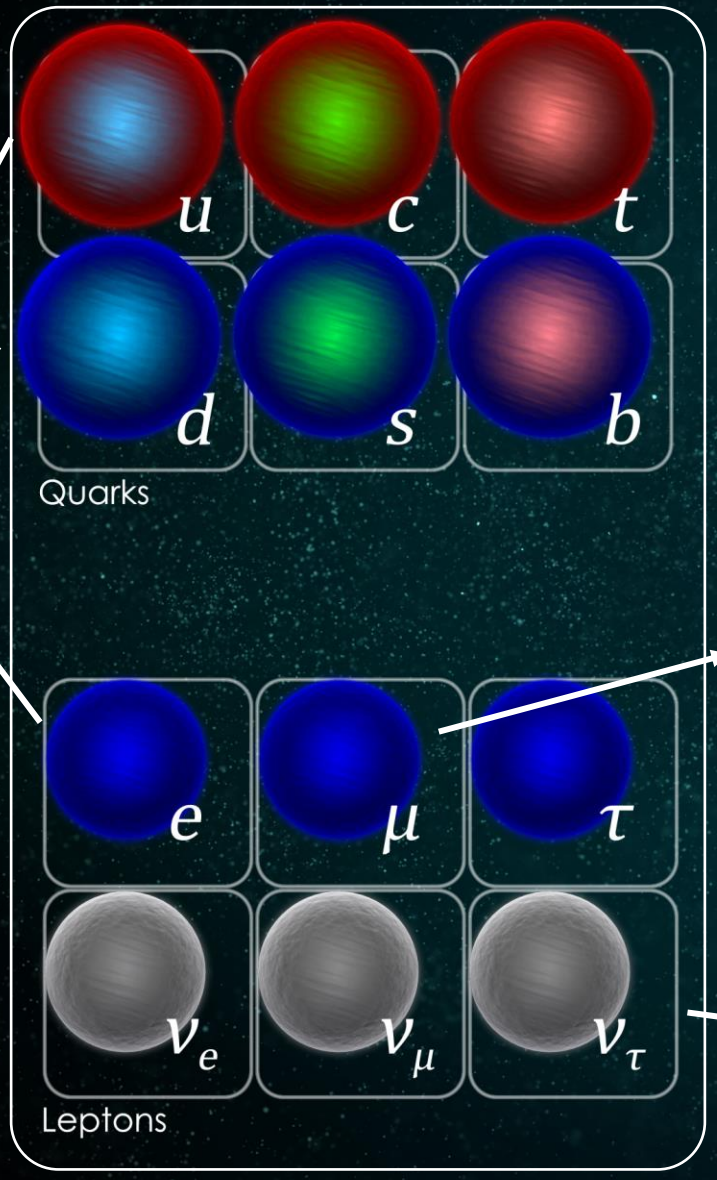


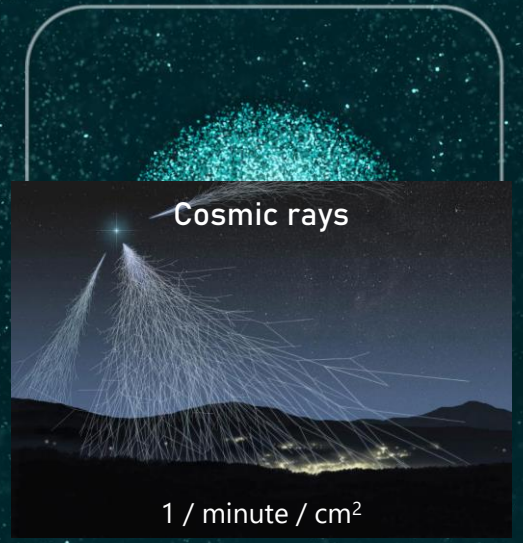
Leptons

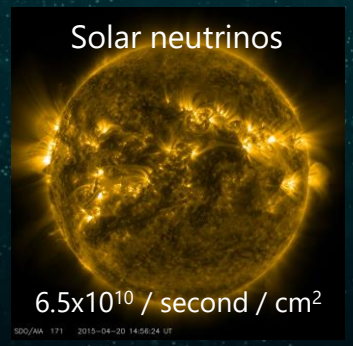
ACCELERATING SCIENCE

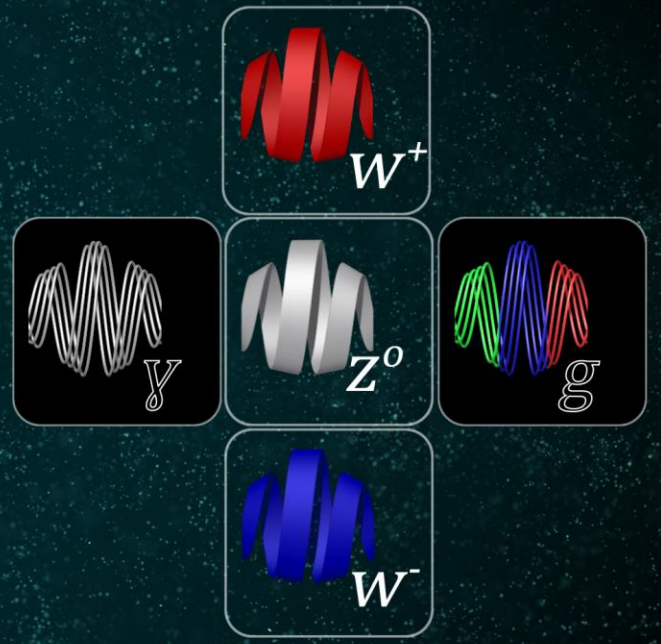
Matter Particles

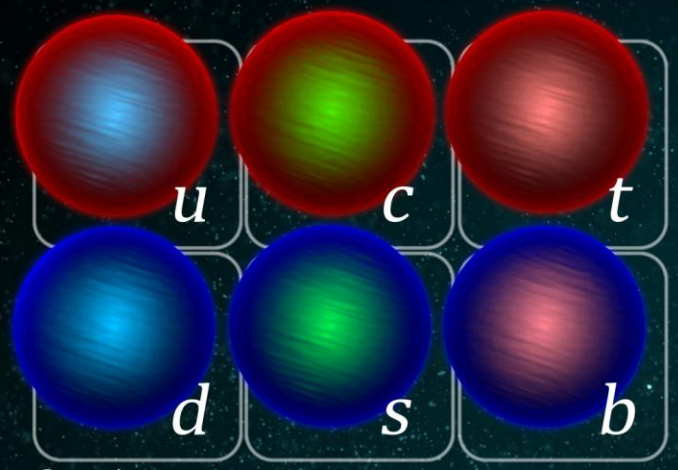
Atomic structure



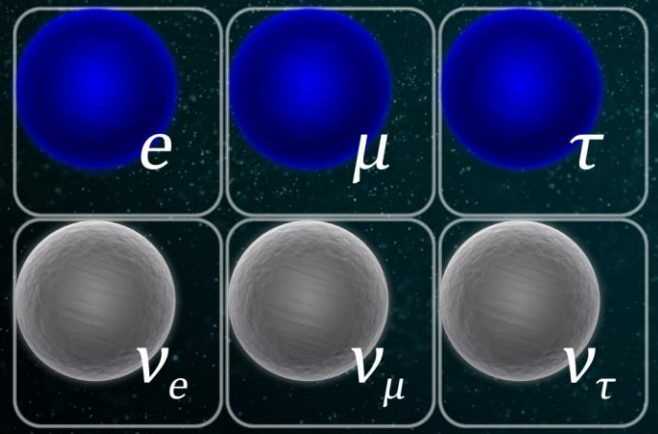








Quarks



Leptons


Higgs boson

Force Carriers

…where's gravity? The SM is incomplete!
→ wide range of Beyond the SM (BSM) theories w/ additional particles/interactions

HIGH ENERGY PARTICLE PHYSICS

THE STANDARD MODEL (SM)

BROKEN SYMMETRY AND THE MASS OF GAUGE VECTOR MESONS*

F. Englert and R. Brout
Faculté des Sciences, Université Libre de Bruxelles, Bruxelles, Belgium
(Received 26 June 1964)

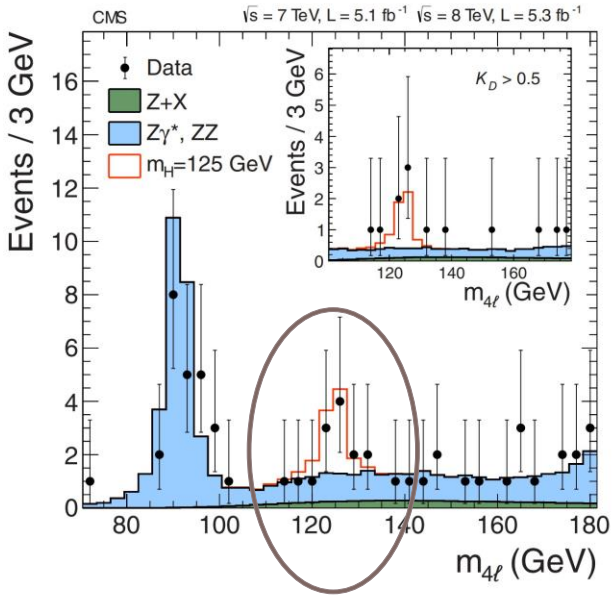
BROKEN SYMMETRIES AND THE MASSES OF GAUGE BOSONS

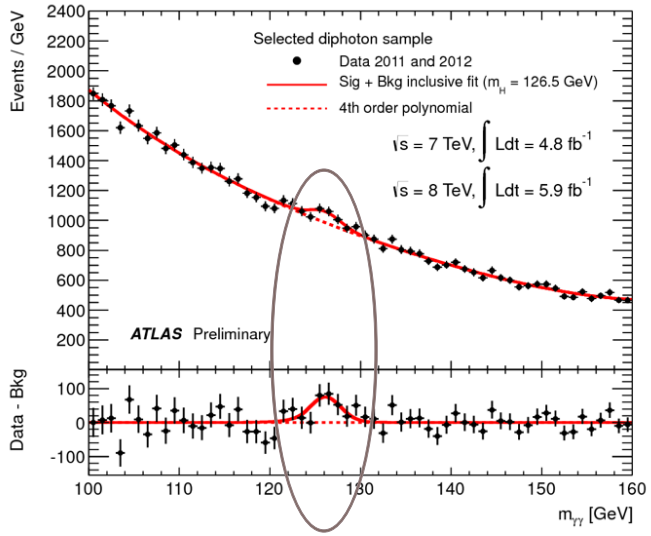
Peter W. Higgs
Tait Institute of Mathematical Physics, University of Edinburgh, Edinburgh, Scotland
(Received 31 August 1964)

GLOBAL CONSERVATION LAWS AND MASSLESS PARTICLES*

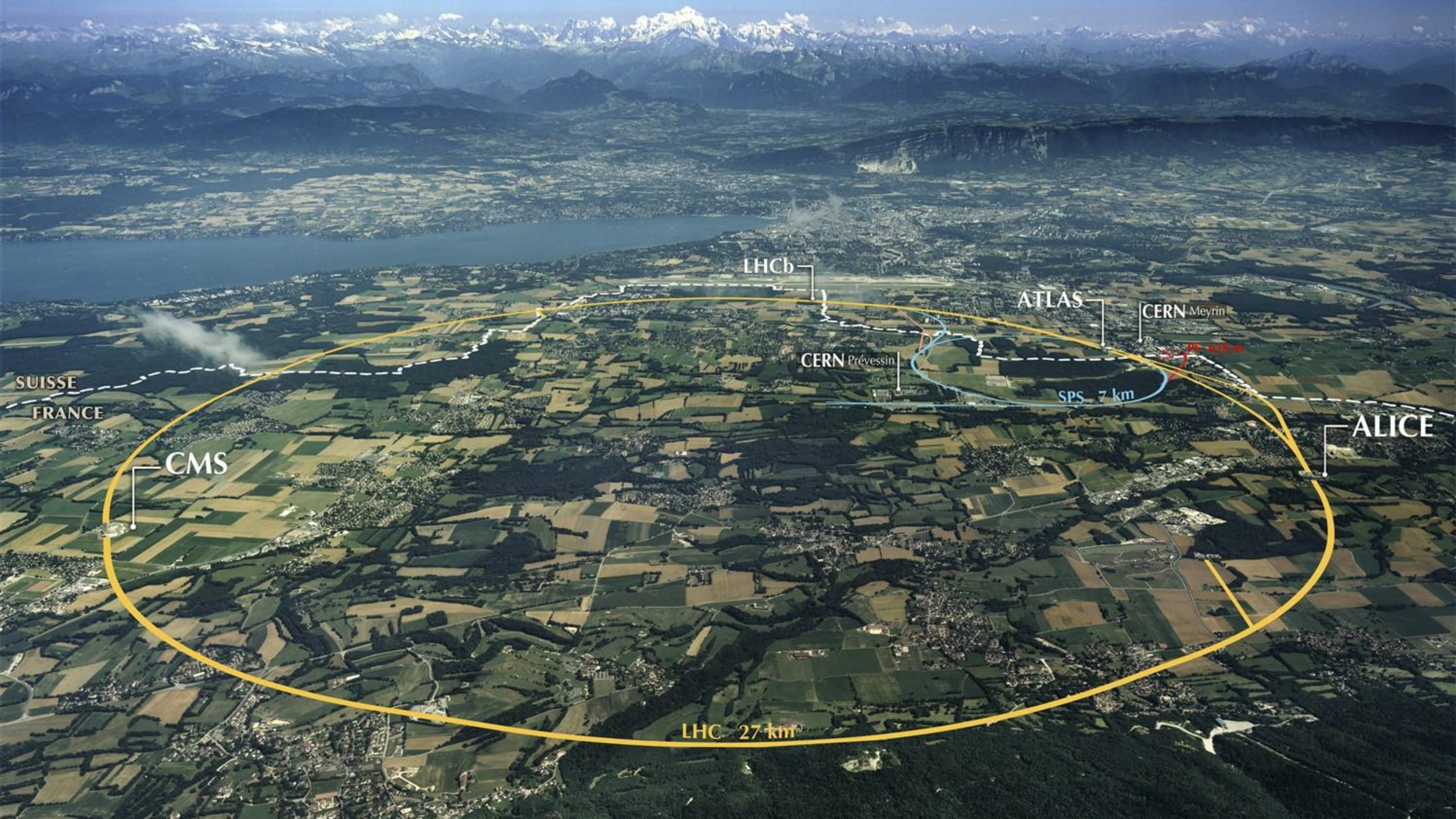
G. S. Guralnik,† C. R. Hagen,‡ and T. W. B. Kibble
Department of Physics, Imperial College, London, England
(Received 12 October 1964)

The PRL "symmetry breaking" papers (1964) famously describe how mass arises in the SM…
"Higgs mechanism" → observable mass state (**Higgs boson**)



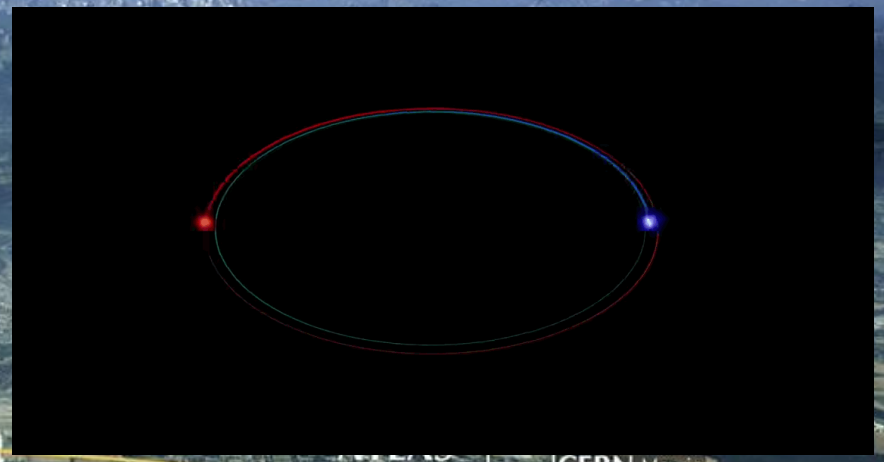


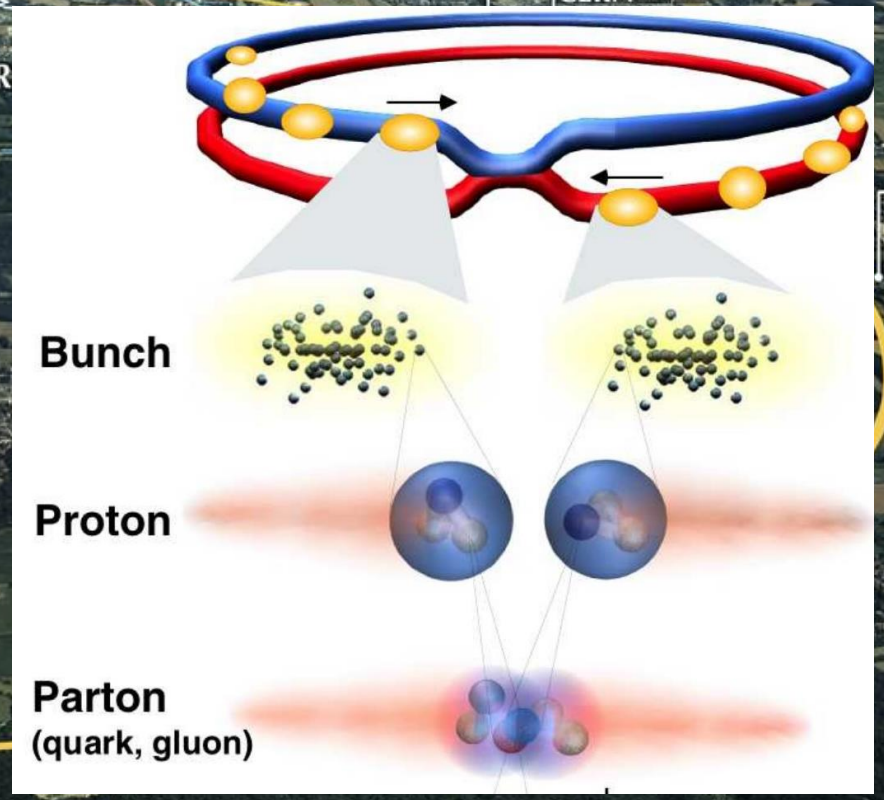
The Higgs boson was discovered at the Large Hadron Collider (LHC) in 2012!


LHCb
ATLAS
CERN Meyrin
CERN Prévessin
SPS 7 km
SUISSE
FRANCE
CMS
ALICE
LHC 27 km

PROTON COLLISIONS:

- Bunches of 1.2×10^{11} protons are accelerated in opposite directions; there are 2808 bunches in each “beam”
- Protons accelerated to energies of 6.5 TeV
  - 99.9999991% the speed of light (roughly the kinetic energy of a mosquito in flight)
  - 11,245 rotations per second
- Bunches crossed at 40 MHz, hundreds of millions of collisions produced per second

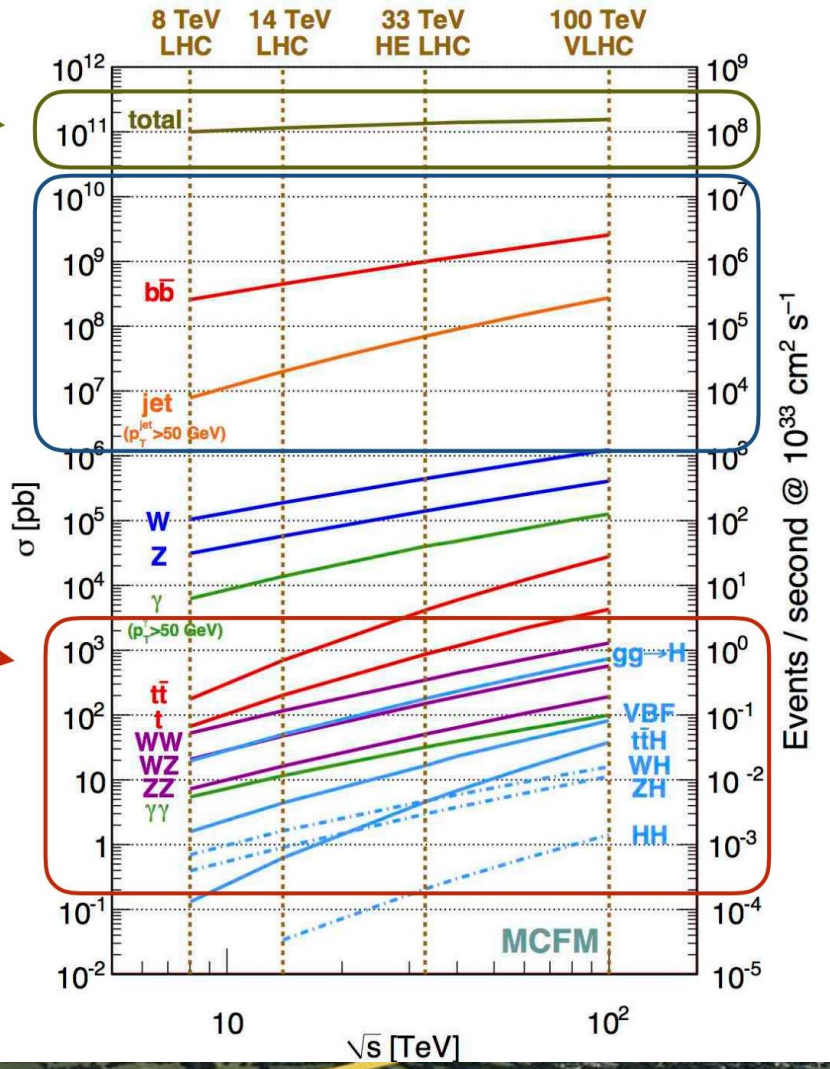




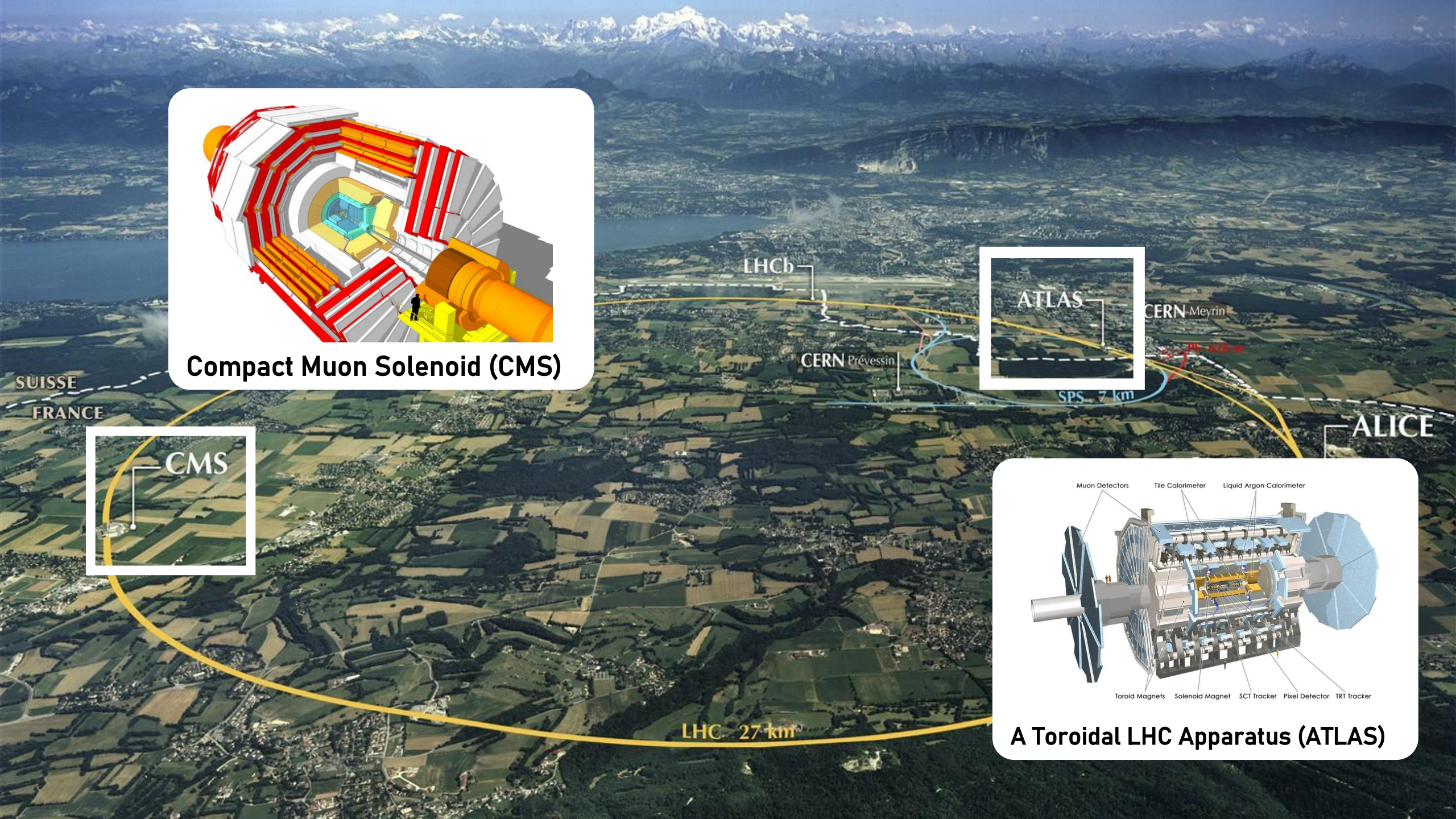
LHC total cross section

No need to store all events

- Most are low energy multi-jet events
- Interesting events are rare

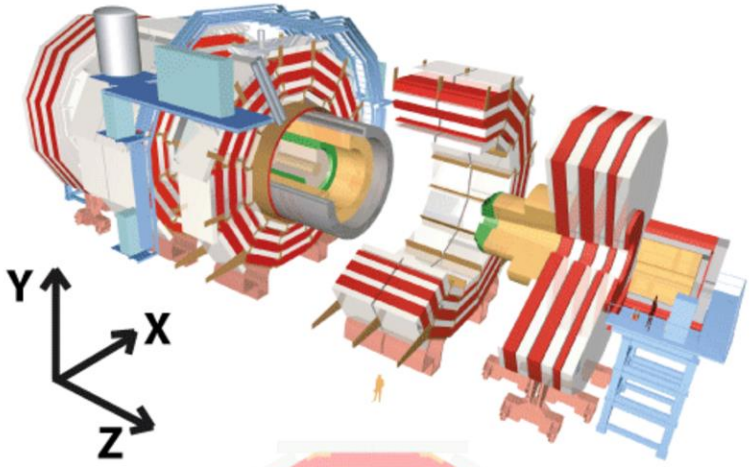


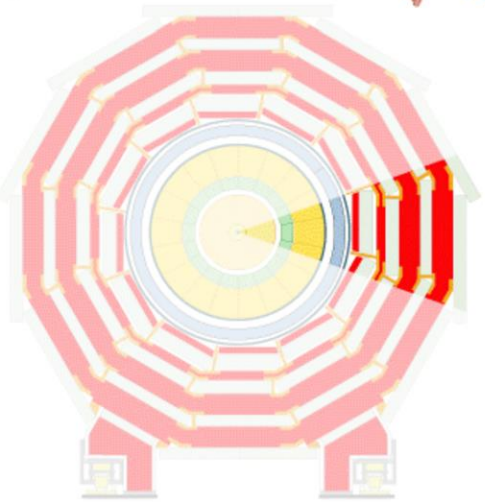
Description of the CMS Trigger Design and Performance
P. Bortignon on behalf of the CMS collaboration

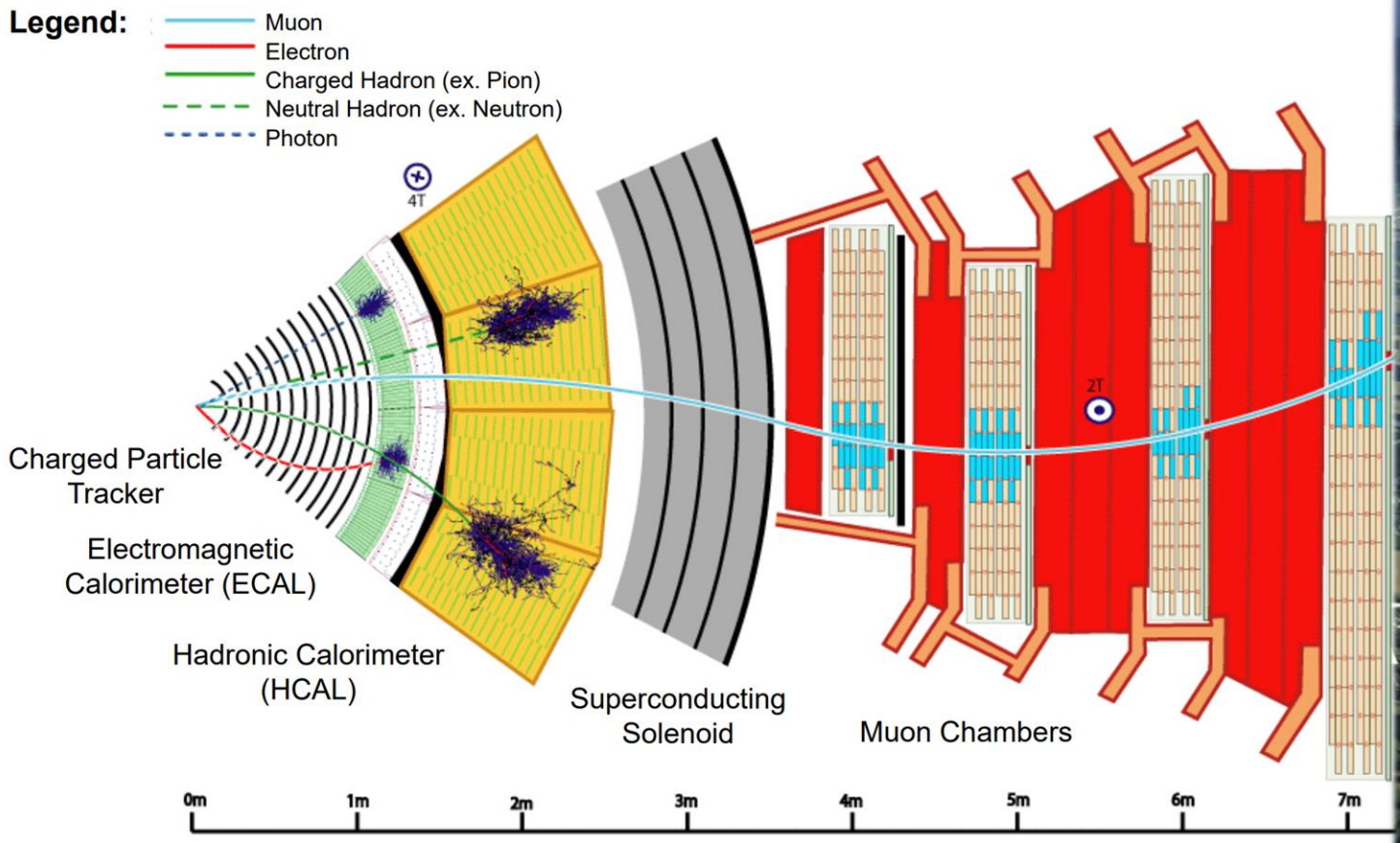


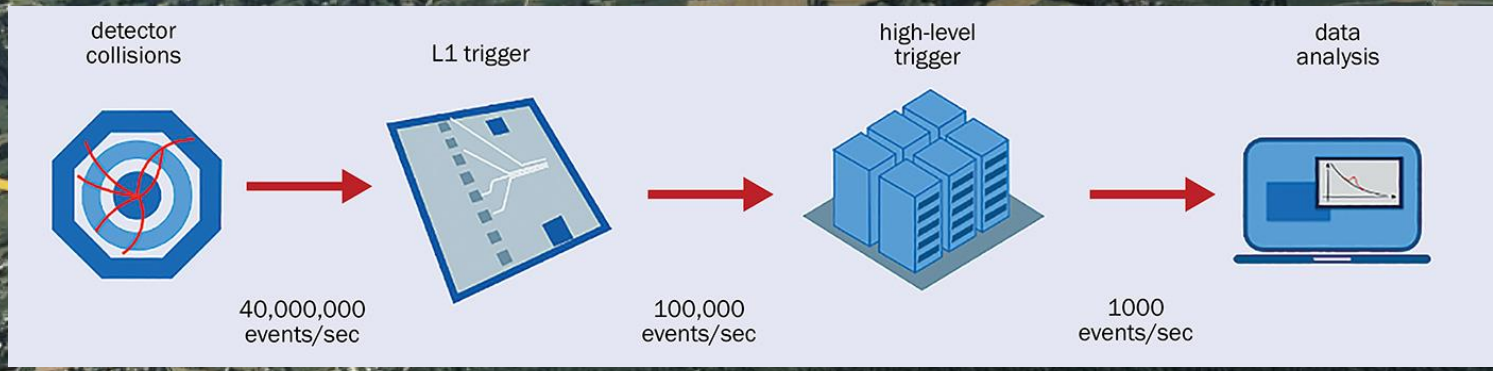
Compact Muon Solenoid (CMS)

A Toroidal LHC Apparatus (ATLAS)


Y
X
Z

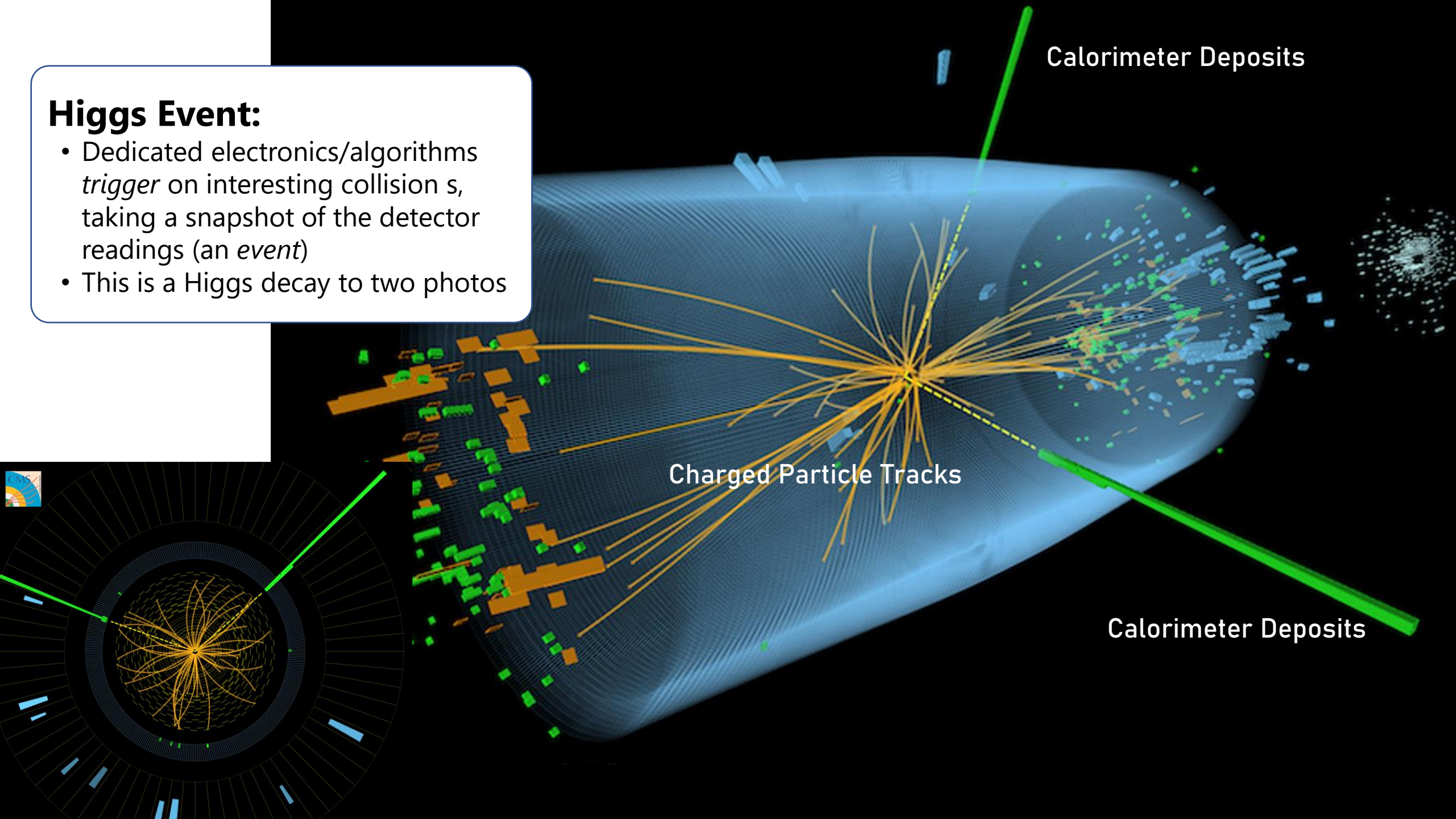

Y
X


Legend:
Muon
Electron
Charged Hadron (ex. Pion)
Neutral Hadron (ex. Neutron)
Photon
4T
2T
Charged Particle Tracker
Electromagnetic Calorimeter (ECAL)
Hadronic Calorimeter (HCAL)
Superconducting Solenoid
Muon Chambers
0m
1m
2m
3m
4m
5m
6m
7m

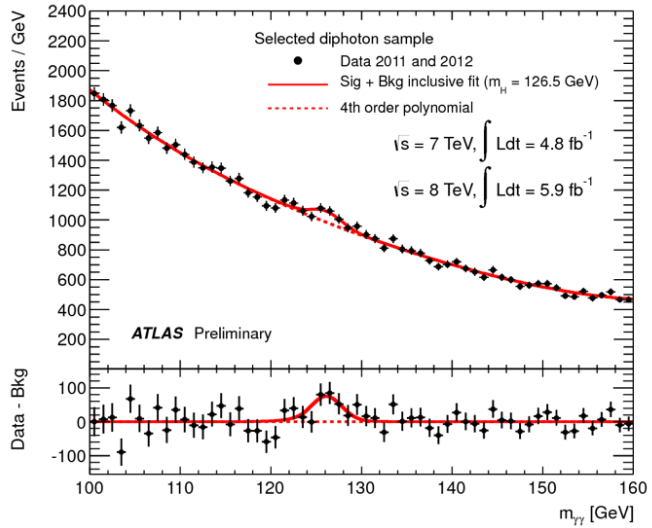

detector collisions
L1 trigger
high-level trigger
data analysis
40,000,000 events/sec
100,000 events/sec
1000 events/sec

Higgs Event:

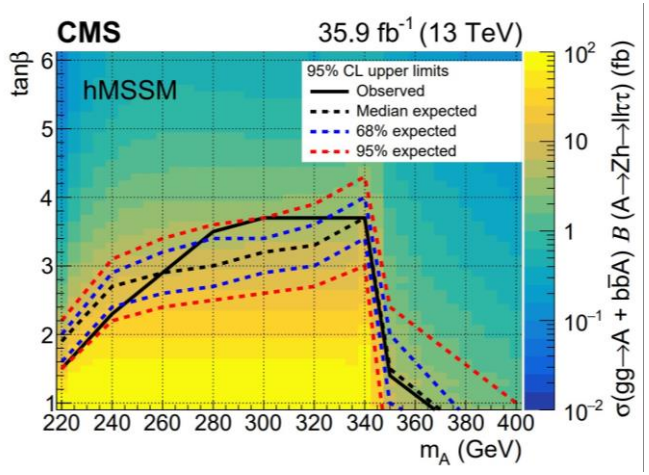
- Dedicated electronics/algorithms *trigger* on interesting collision s, taking a snapshot of the detector readings (an *event*)
- This is a Higgs decay to two photos



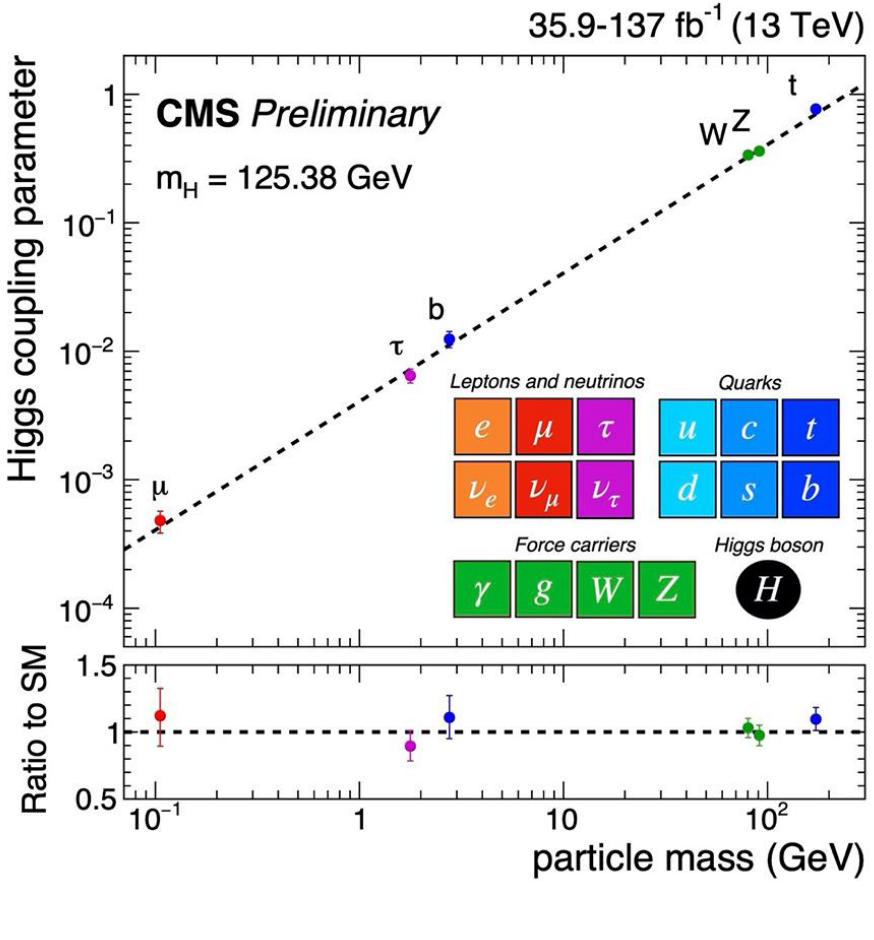
Discovering New Particles

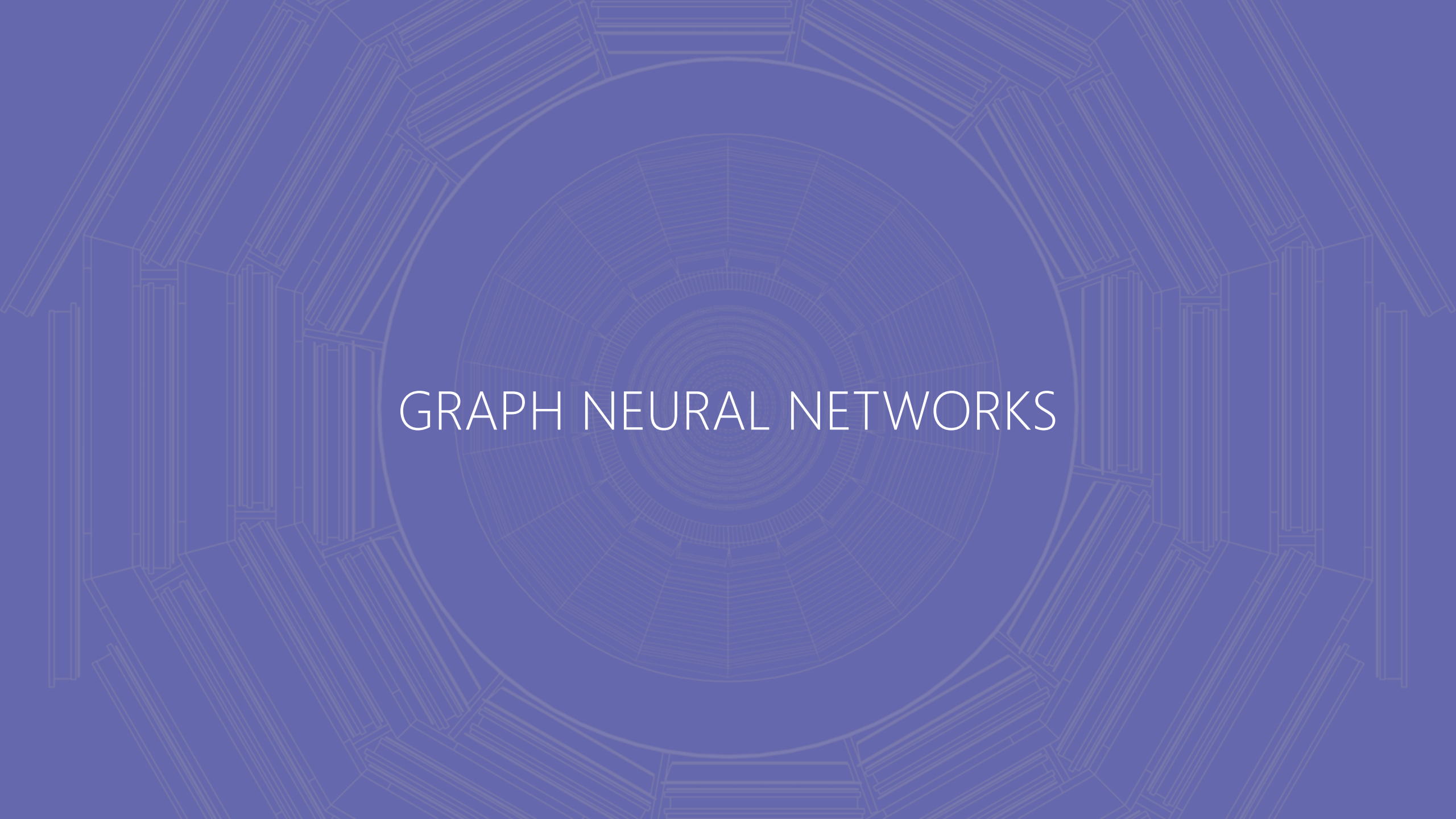


Placing Limits on New Physics



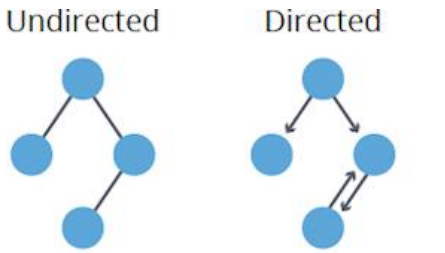
Precision Measurements



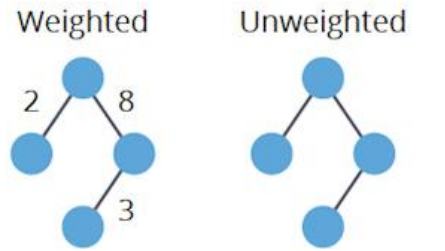

GRAPH NEURAL NETWORKS

GRAPH NEURAL NETWORKS
GRAPH-STRUCTURED DATA

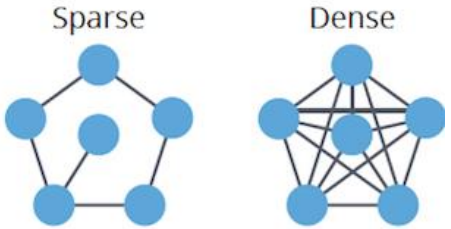
- Graphs represent relational data
  - Entities → Nodes: $u \in \mathcal{V}$
    - Node features: $\boldsymbol{x}_u \in \mathbb{R}^{d_\mathcal{V}}$
  - Relations → Edges: $(u, v) \in \mathcal{E}$
    - Edge features: $\boldsymbol{e}_{uv} \in \mathbb{R}^{d_\mathcal{E}}$



Directed edges specify an incoming and outgoing node



Edge features might be weights or otherwise more complicated attributes



Sparsity is problem-dependent… roughly, sparse if $|\mathcal{E}| \ll |\mathcal{V}|^2$

[Graph Algorithms in Neo4j: Graph Algorithm Concepts]

GRAPH NEURAL NETWORKS
GRAPH-STRUCTURED DATA

- Graphs represent relational data
  - Entities → Nodes: $u \in \mathcal{V}$
    - Node features: $\boldsymbol{x}_u \in \mathbb{R}^{d_\mathcal{V}}$
  - Relations → Edges: $(u, v) \in \mathcal{E}$
    - Edge features: $\boldsymbol{e}_{uv} \in \mathbb{R}^{d_\mathcal{E}}$

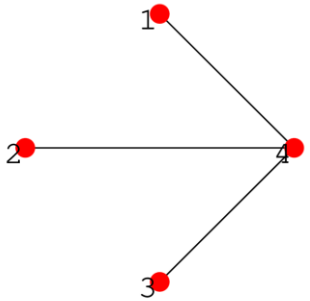
Edge Representations

PyTorch

Adjacency Matrices

$A_{adjacency} \in \mathbb{R}^{d_\mathcal{V} \times d_\mathcal{V}}$

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



Incidence Matrices

$A_{incidence} \in \mathbb{R}^{d_\mathcal{V} \times d_\mathcal{E}}$

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

PyTorch geometric *optimized for sparse adjacency structure*

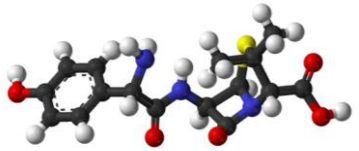
Sparse Index Lists (COO)

$I_{COO} \in \mathbb{R}^{2 \times d_\mathcal{E}}$

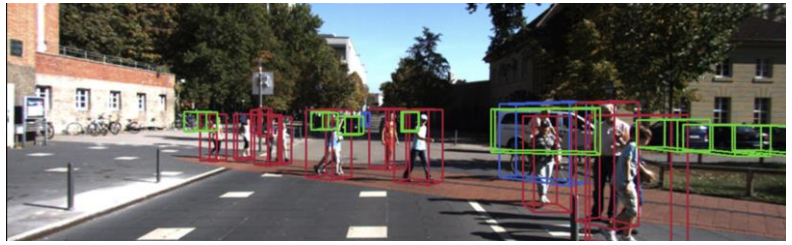
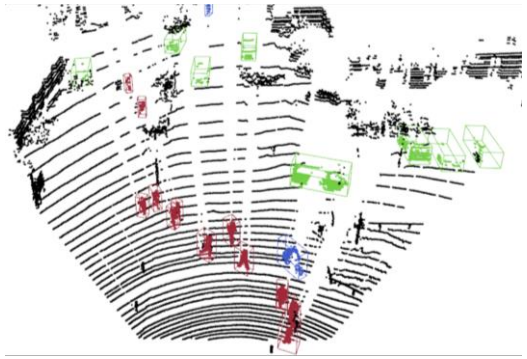
$$\begin{bmatrix} [1 & 2 & 3] \\ [4 & 4 & 4] \end{bmatrix}$$

Graph Algorithms in Neo4j: Graph Algorithm Concepts

Drug Discovery

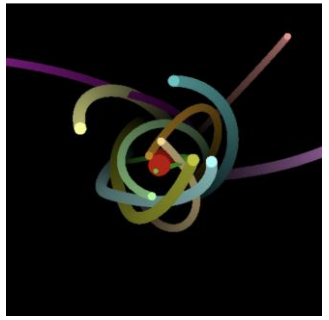
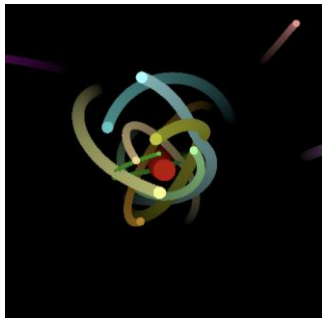
$GNN($  $) \rightarrow$ global molecular property

Instance Segmentation

$GNN($  $) \rightarrow$ 

2003.01251.pdf (arxiv.org)

Physics Simulation

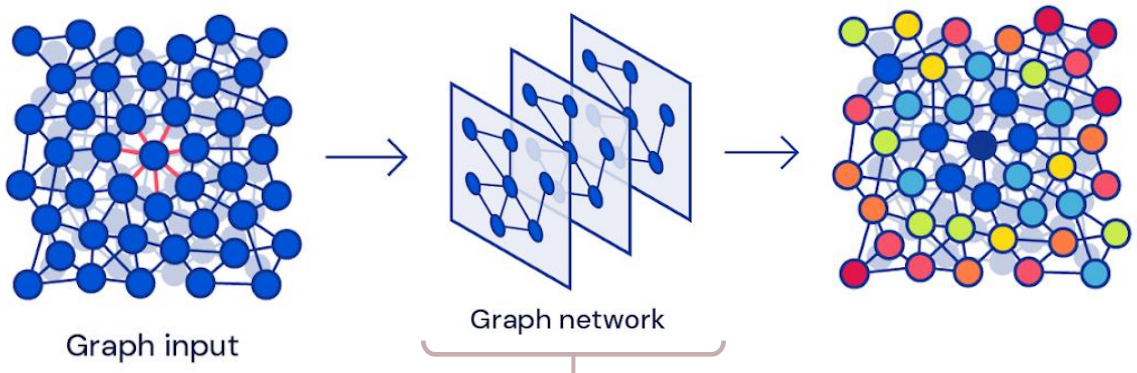
$GNN($  $) \rightarrow$ 

[1612.00222] Interaction Networks for Learning about Objects, Relations and Physics (arxiv.org)

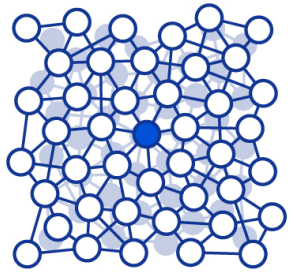
GRAPH NEURAL NETWORKS

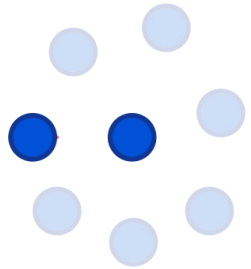
HIGH-LEVEL EXAMPLES

GNNs: High-Level View



Propagation Modules:
Neural Message Passing





"Messages" computed from each node's neighborhood are used to update graph features… *k* iterations → info from *k*-hop neighborhoood

Figure Source: https://deepmind.com/blog/article/Towards-understanding-glasses-with-graph-neural-networks

GRAPH NEURAL NETWORKS
GNN OVERVIEW

GRAPH NEURAL NETWORKS
LEARNING ON SETS

Set Learning

- Many real-world objects don't have a natural ordering
- Why not DNNs on sets? Many different orderings of the inputs to consider… need a permutation symmetric function of inputs!

Permutation invariance: $f(PX) = f(X)$

e.g. DeepSets: take a set X and two approximators (MLPs) ϕ, ψ

$$f(X) = \phi\left(\sum_{x \in X} \psi(x)\right)$$

any permutation invariant function of countable sets

sum aggregation is permutation-invariant

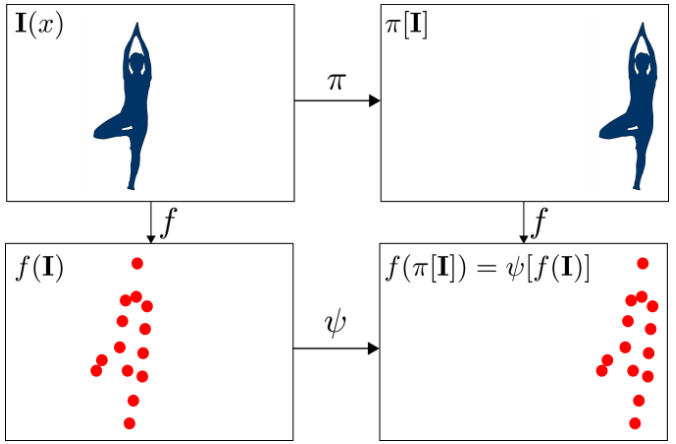
latent representation of each element

- These are "global" predictions on a set
  (returns number(s) aggregated from the whole set)
- In graphs, we have more structure (edges) and might require per-node or per-edge predictions…

[1703.06114] Deep Sets (arxiv.org)
[1806.01261] Relational inductive biases, deep learning, and graph networks (arxiv.org)

GRAPH NEURAL NETWORKS
LEARNING ON GRAPHS

Equivariance Illustration



Graph Learning Template:

- Graphs support arbitrary pairwise relations between nodes
- Relational inductive bias: input graphs explicitly define relations for the learning model to leverage

Node Neighborhoods:

- Neighborhood of u: $N(u) = \{v : (u, v) \in E\}$
- Neighborhood node features: $X_{N(u)} = \{\{x_v : v \in N(u)\}\}$
- Neighborhood edge features: $E_{N(u)} = \{\{e_{uv} : (u, v) \in E\}\}$

Permutation invariance: $f(\boldsymbol{PX}, \boldsymbol{PAP}^T) = f(\boldsymbol{X}, \boldsymbol{A})$ → graph-level predictions

Permutation equivariance: $f(\boldsymbol{PX}, \boldsymbol{PAP}^T) = \boldsymbol{P}f(\boldsymbol{X}, \boldsymbol{A})$ → node-level predictions

Permutation equivariant function of graphs:

$$f(X, A) = \begin{pmatrix} - & g(\boldsymbol{x}_1, X_{N(1)}, E_{N(1)}) & - \\ - & g(\boldsymbol{x}_2, X_{N(2)}, E_{N(2)}) & - \\ & \dots & \\ - & g(\boldsymbol{x}_{|V|}, X_{N(|V|)}, E_{N(|V|)}) & - \end{pmatrix}$$

equivariance enforced by applying g to all nodes equally

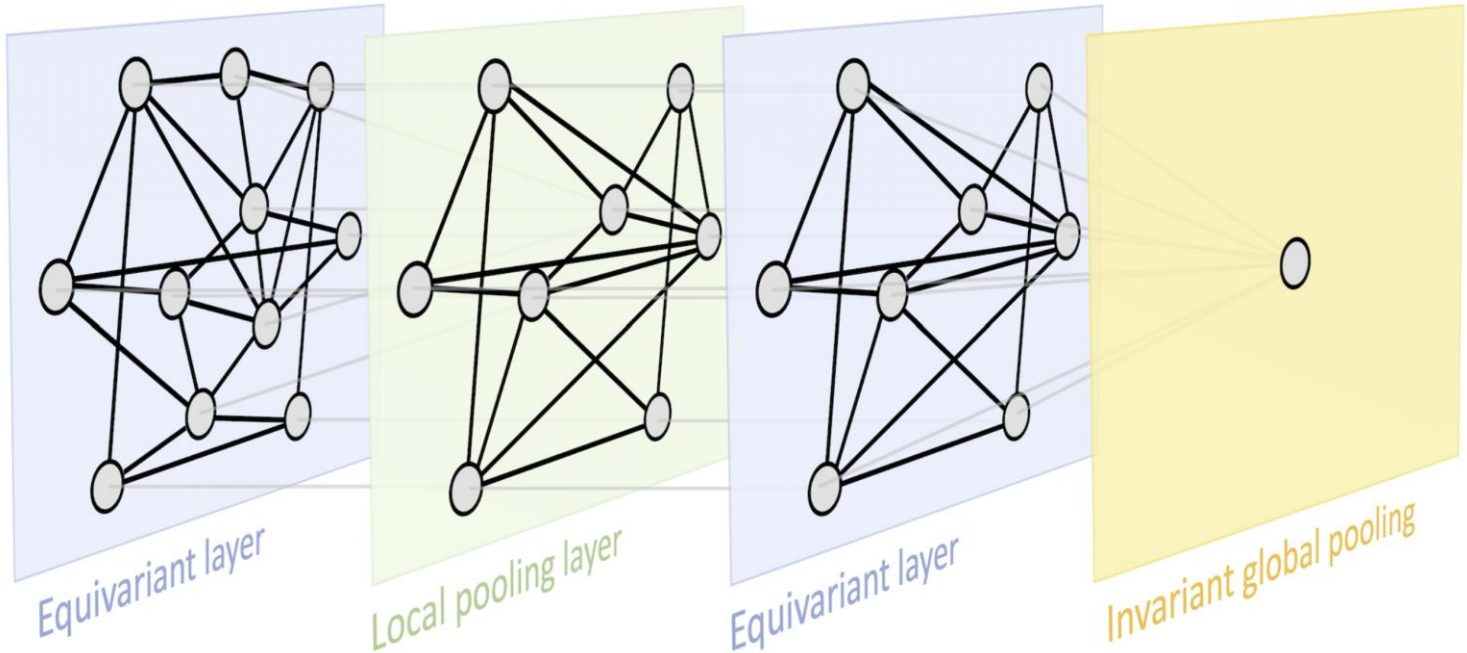
permutation equivariant function of graphs

local function operating on each node's neighborhood… needs to be permutation invariant!

GRAPH NEURAL NETWORKS
GNN LAYERS

Applying Equivariant Layers:

- Equivariant layers update the state of each node while preserving the structure of the graph
- Pooling layers subsample or otherwise combine graph nodes
  - Global pooling is used for graph-level predictions



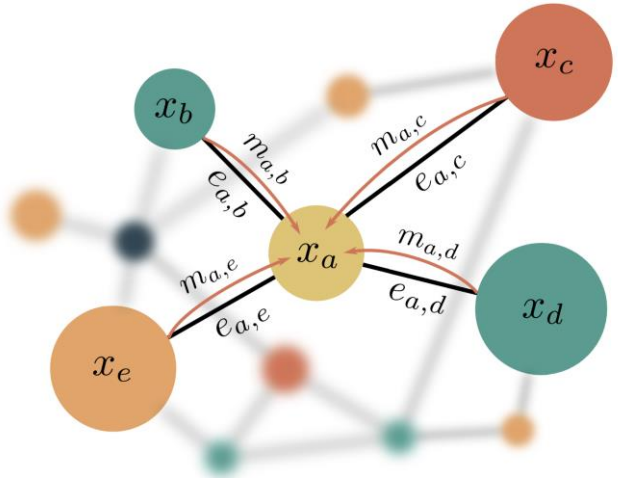
2104.13478.pdf (arxiv.org)

GRAPH NEURAL NETWORKS

NEURAL MESSAGE PASSING

Message Passing (MPNN) Layers:

Framework for many equivariant graph updates



At each layer *k*, compute messages in each node's neighborhood:

$$\boldsymbol{m}_{uv}^{(k)} = \psi^{(k)}\left(\boldsymbol{h}_u^{(k-1)}, \boldsymbol{h}_v^{(k-1)}, \boldsymbol{e}_{uv}^{(k-1)}\right)$$

Aggregate messages in a permutation-invariant way:

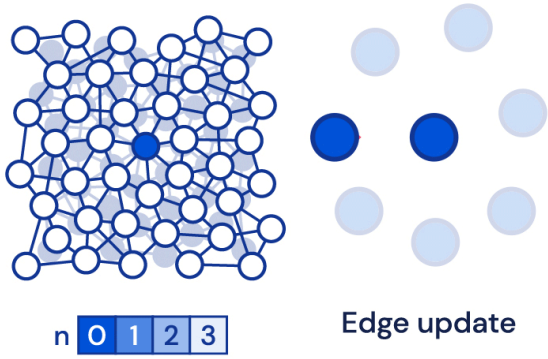
$$\boldsymbol{a}_u^{(k)} = \bigoplus_{v \in N(u)} \boldsymbol{m}_{uv}^{(k)}$$

Messages passed only from u's direct neighbors

Any permutation invariant operation (e.g. sum, mean, max)

Update the node's state based on the messages it received:

$$\boldsymbol{h}_u^{(k)} = \phi^{(k)}(\boldsymbol{h}_u^{(k-1)}, \boldsymbol{a}_u^{(k)})$$

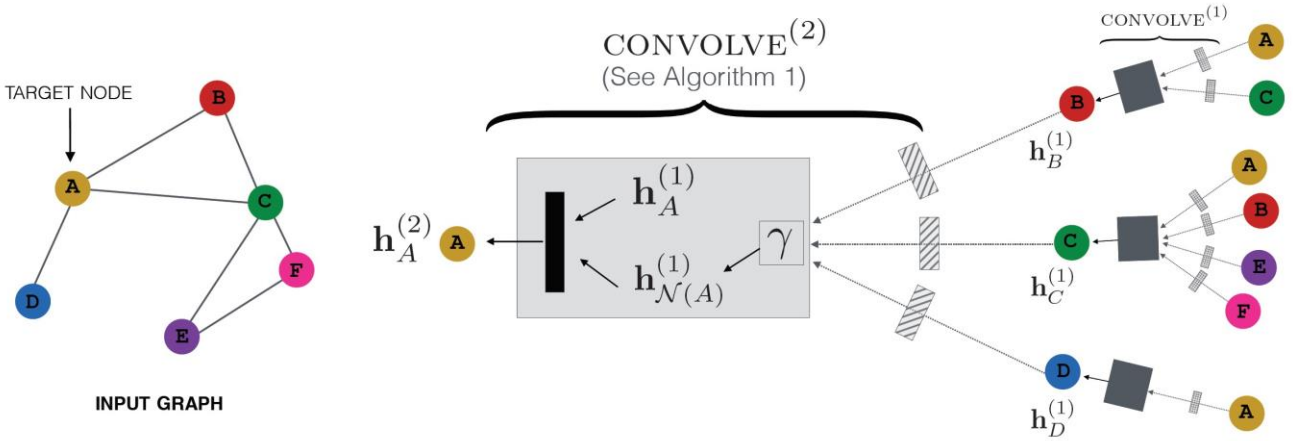


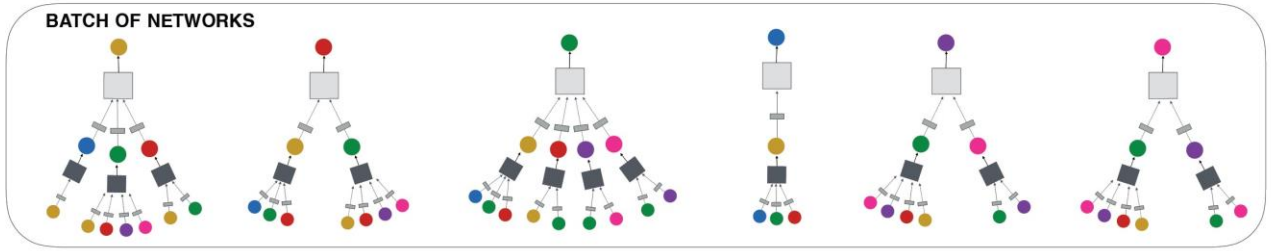
GRAPH NEURAL NETWORKS
REPEATED MESSAGE PASSING

Generic MPNN Layers:

$$\boldsymbol{h}_u^{(k)} = \phi^{(k)}\left[\boldsymbol{h}_u^{(k-1)}, \bigoplus_{v \in N(u)} \psi^{(k)}\left(\boldsymbol{h}_u^{(k-1)}, \boldsymbol{h}_v^{(k-1)}, \boldsymbol{e}_{uv}^{(k-1)}\right)\right]$$

Node Updates: collecting info from each node's *k*-hop neighborhood at the k^{th} layer





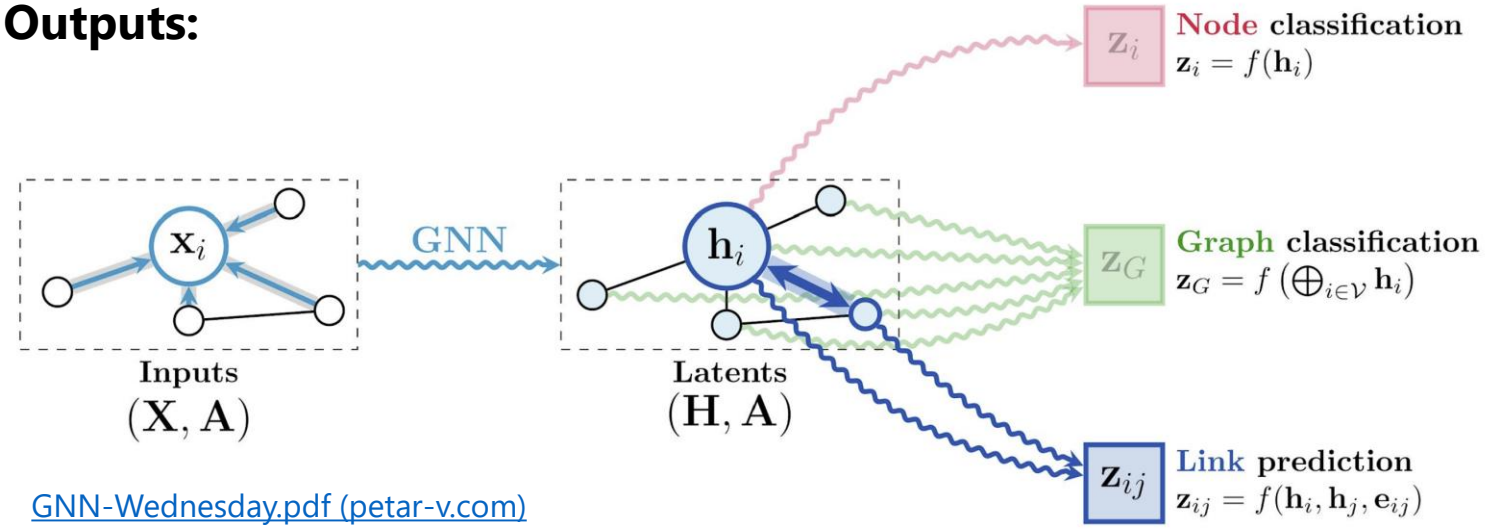
https://doi.org/10.1145/3219819.3219890

GRAPH NEURAL NETWORKS
PREDICTIONS, LIMITATIONS

Generic MPNN Layers:

$$\boldsymbol{h}_u^{(k)} = \phi^{(k)}\left[\boldsymbol{h}_u^{(k-1)}, \bigoplus_{v \in N(u)} \psi^{(k)}\left(\boldsymbol{h}_u^{(k-1)}, \boldsymbol{h}_v^{(k-1)}, \boldsymbol{e}_{uv}^{(k-1)}\right)\right]$$

Outputs:



GNN-Wednesday.pdf (petar-v.com)

Limitations:

- Not every problem is easily mapped to a graph
- MPNNs aren't guaranteed to solve every problem
  (e.g. discerning some non-isomorphic graphs, generic MPNN's representational power upper bounded by the 1-WL test)

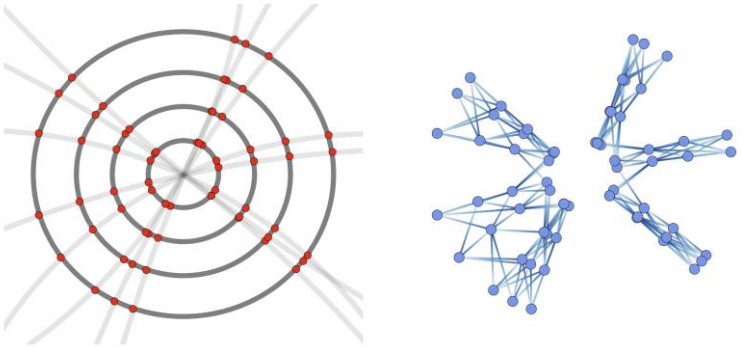
GNNS AT THE LHC

GNNS AT THE LHC

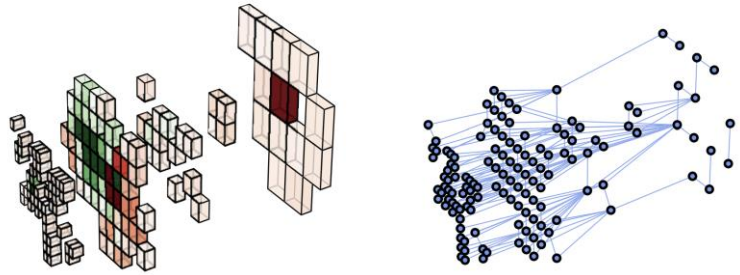
COMMON APPLICATIONS

Low-Level Reconstruction Tasks

Combine detector signals to form "building blocks" for various particle types



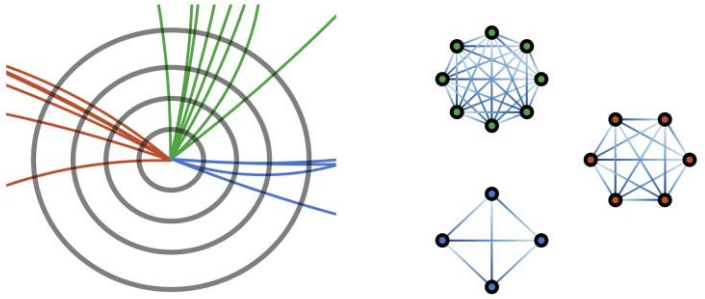
Track Reconstruction



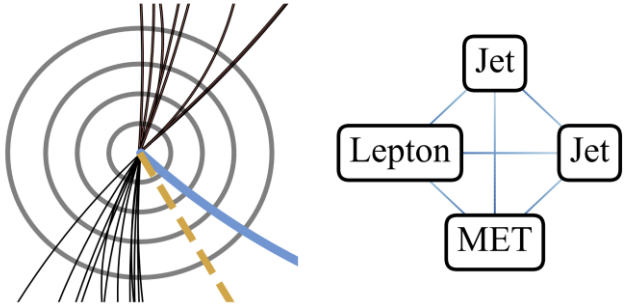
Calorimeter Segmentation

Higher Level Particle-based Tasks

Given a set of particles, can we combine them to represent a specific decay? Can we identify a physics signal?



Jet Identification



Event Classification

GNNS AT THE LHC

WHY GNNS?

Common Justifications (task-dependent)

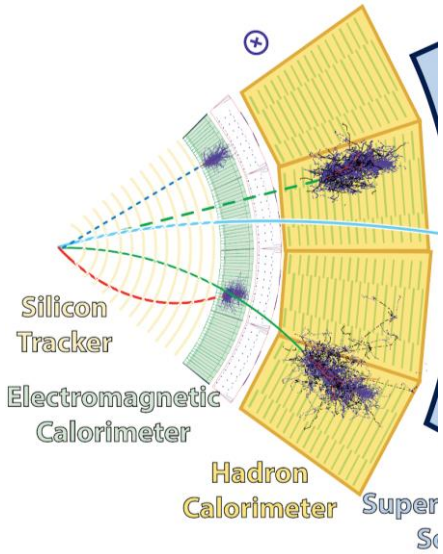
- Many LHC datasets have inherent relational structure and/or no inherent ordering
- Grids, sequences, etc. cannot naturally represent irregular detector geometries
  - A small fraction of sensors are activated in any given event ➔ data is sparse
  - Many different data sizes (particle counts, sensor readings, etc.)
- LHC data is heterogeneous
  - Data recorded from multiple subdetectors
  - Different types of particles
- Excellent performance
  - Relational inductive bias
  - Message passing leverages low-level detector info in addition to global (or otherwise human-devised) info
  - Generally smaller architectures (qualitatively speaking)

GNN TRACKING

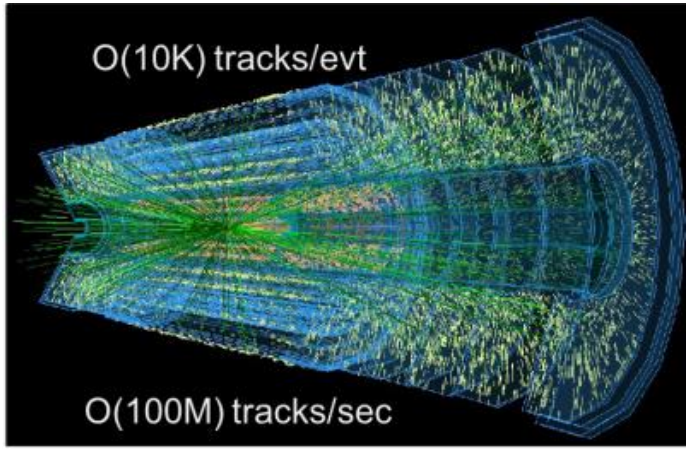
TRADITIONAL TRACKING METHODS

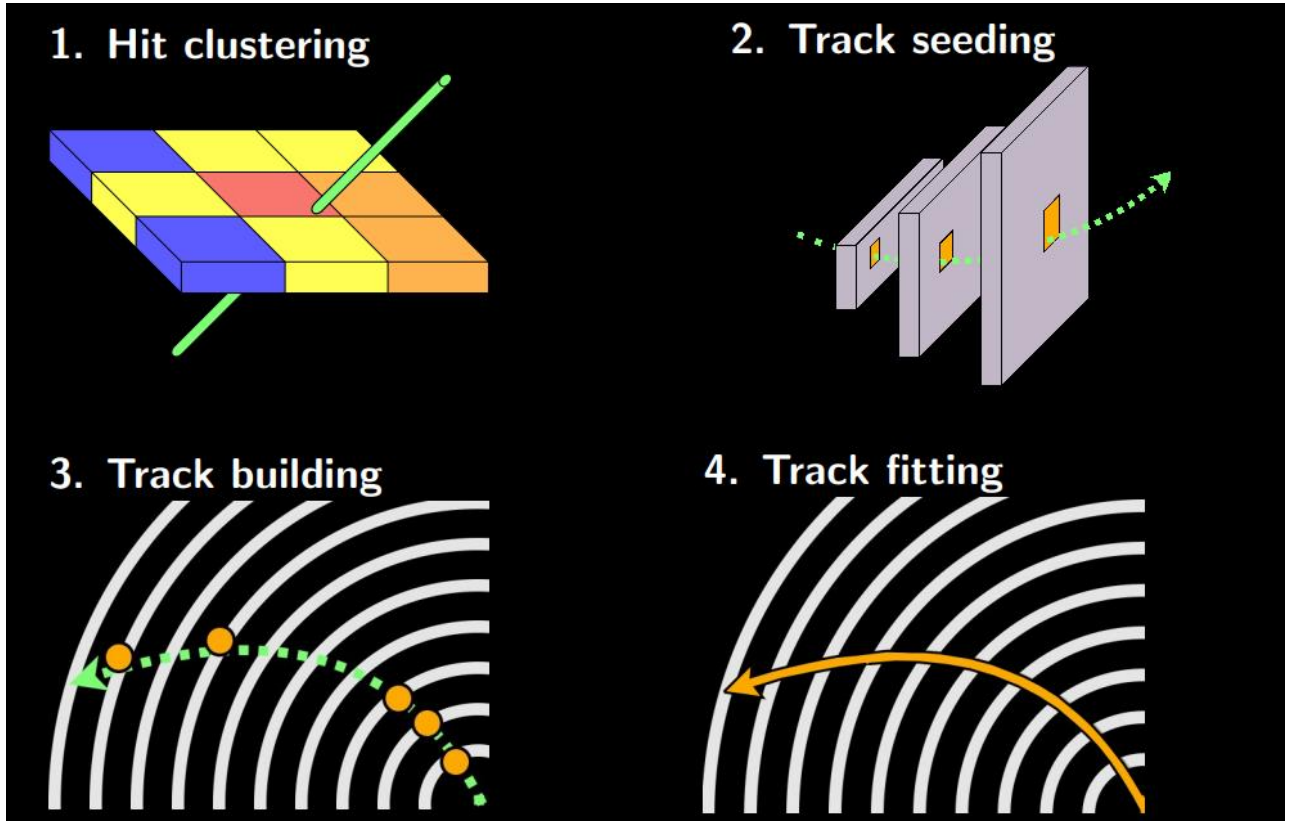
Trackers are the innermost detector layers responsible for sampling particle trajectories

Tracks allow us to measure *momentum*, *direction*, *origin*, and *charge*



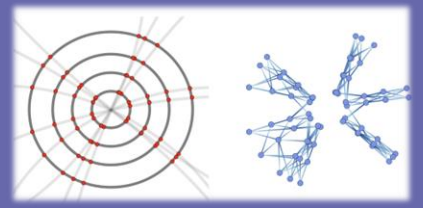
Challenge: This is where we're headed… many, many tracks per event!





Tracking: rebuilding particle trajectories from spatial measurements, traditionally an iterative process … doesn't scale with increasing detector activity!

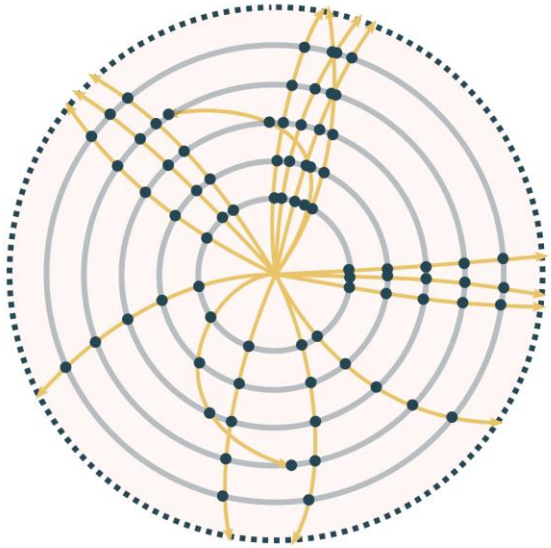
Connecting the dots: applying deep learning techniques in HEP | EP News (cern.ch)



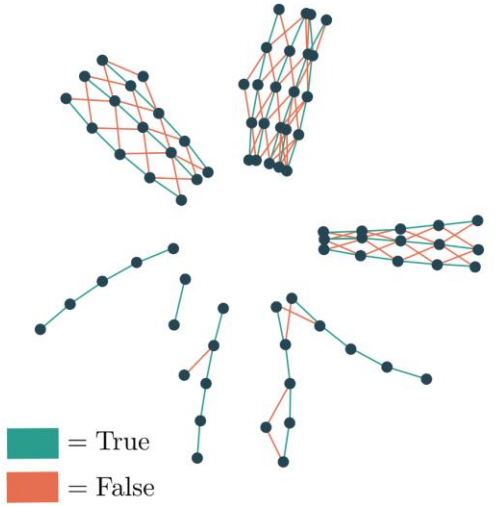
GNN TRACKING
POSED AS AN ML TASK

Edge Classification Task

- Draw edges to hypothesize various particle trajectories, train a GNN to classify edges

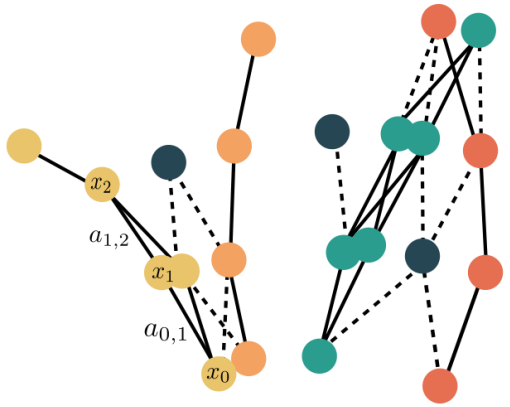


Input data is a 3D *point cloud*



Train on graphs with edge truth labels

- **Key steps** (general to many GNN workflows)
  1) Graph construction from underlying data
  2) GNN inference
  3) Post-processing of GNN predictions



Node Labels
- $l_i = 0$
- $l_i = 1$
- $l_i = 2$
- $l_i = 3$
- $l_i = 4$

Edge Labels
- $y_{ij} = 0$
- $y_{ij} = 1$

Input Graph

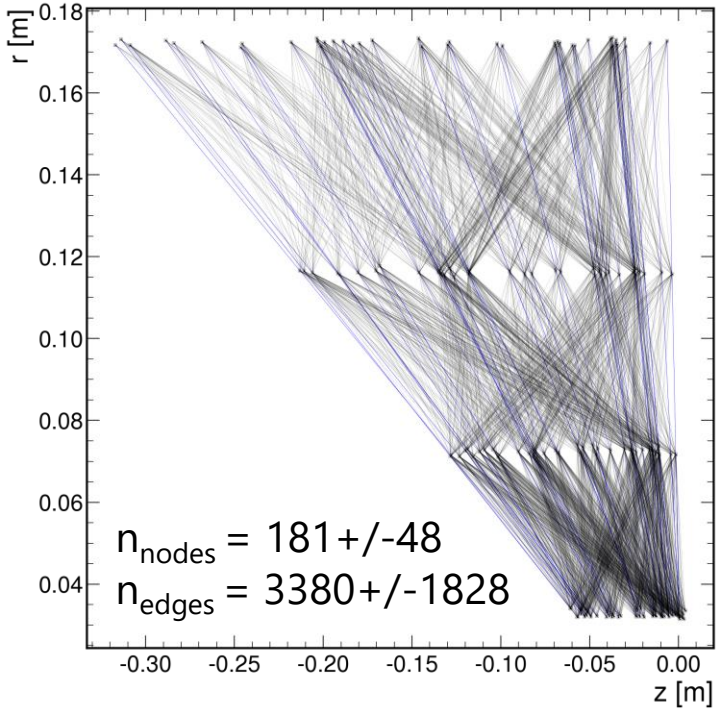
Node Features: $x_i = (r_i, \phi_i, z_i)$

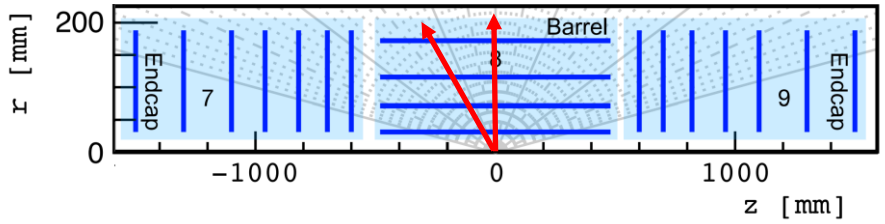
Edge Features: $a_{ij} = (\Delta r_{ij}, \Delta\phi_{ij}, \Delta\eta_{ij}, \Delta R_{ij})$

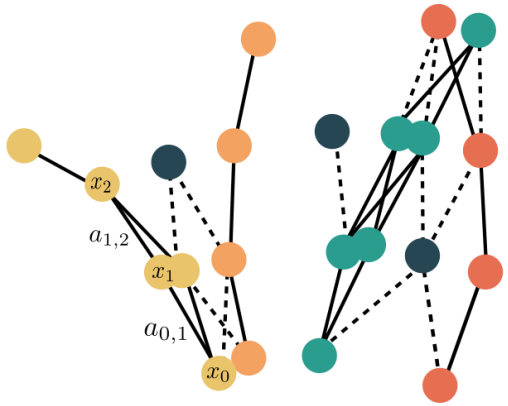
Example GNN-based tracking workflow

1) Graph Construction

Draw edges between particles detector hits based on some initial constraints, clustering, etc.







Node Labels
- $l_i = 0$
- $l_i = 1$
- $l_i = 2$
- $l_i = 3$
- $l_i = 4$

Edge Labels
- $y_{ij} = 0$
- $y_{ij} = 1$

Input Graph

Node Features: $x_i = (r_i, \phi_i, z_i)$

Edge Features: $a_{ij} = (\Delta r_{ij}, \Delta\phi_{ij}, \Delta\eta_{ij}, \Delta R_{ij})$

Edge Classifier
(updates edge features)

Node Features: $x_i = (r_i, \phi_i, z_i)$

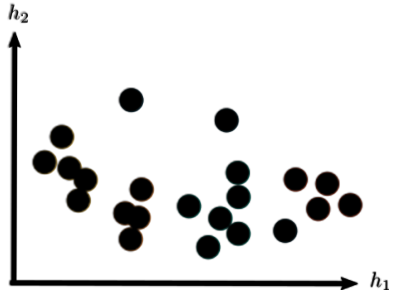
Edge Features:
$\tilde{a}_{ij} = (w_{ij}, \Delta r_{ij}, \Delta\phi_{ij}, \Delta\eta_{ij}, \Delta R_{ij})$

Opacity ~ Edge Score (w_{ij})

Object Condensation
(coordinates in learned clustering space)

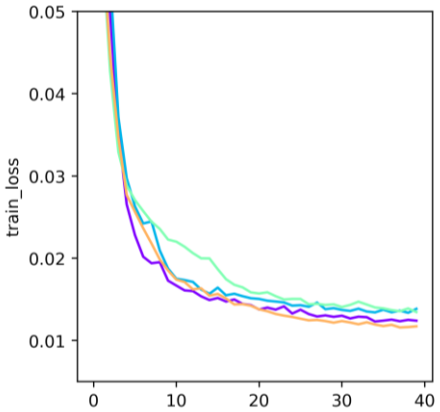
New Coordinates: $h_i \in \mathbb{R}^{d_{\text{out}}}$

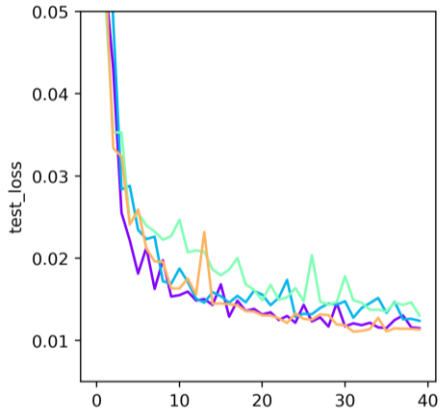
Condensation Strength: $\beta_i \in (0, 1)$



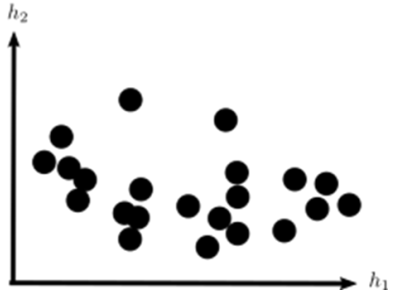
2) GNN Inference

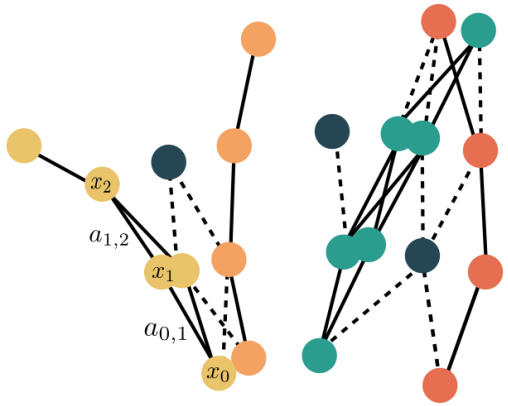
Train a GNN to classify the edges (binary cross entropy) **and** cluster signals belonging to the same particle (object condensation)





Output: track hit coordinates in a learned clustering space





Node Labels
- $l_i = 0$
- $l_i = 1$
- $l_i = 2$
- $l_i = 3$
- $l_i = 4$

Edge Labels
- $y_{ij} = 0$
- $y_{ij} = 1$

Input Graph

Node Features: $x_i = (r_i, \phi_i, z_i)$

Edge Features: $a_{ij} = (\Delta r_{ij}, \Delta\phi_{ij}, \Delta\eta_{ij}, \Delta R_{ij})$

Edge Classifier
(updates edge features)

Node Features: $x_i = (r_i, \phi_i, z_i)$

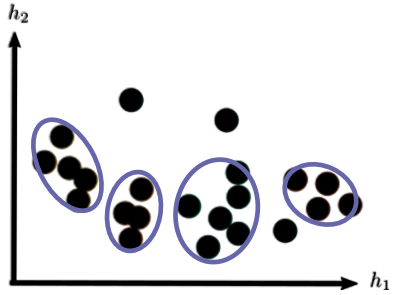
Edge Features:
$\tilde{a}_{ij} = (w_{ij}, \Delta r_{ij}, \Delta\phi_{ij}, \Delta\eta_{ij}, \Delta R_{ij})$

Opacity ~ Edge Score (w_{ij})

Object Condensation
(coordinates in learned clustering space)

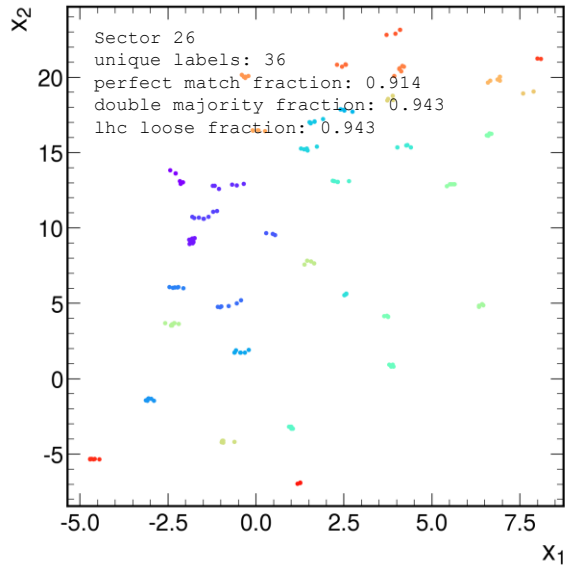
New Coordinates: $h_i \in \mathbb{R}^{d_{\text{out}}}$

Condensation Strength: $\beta_i \in (0, 1)$

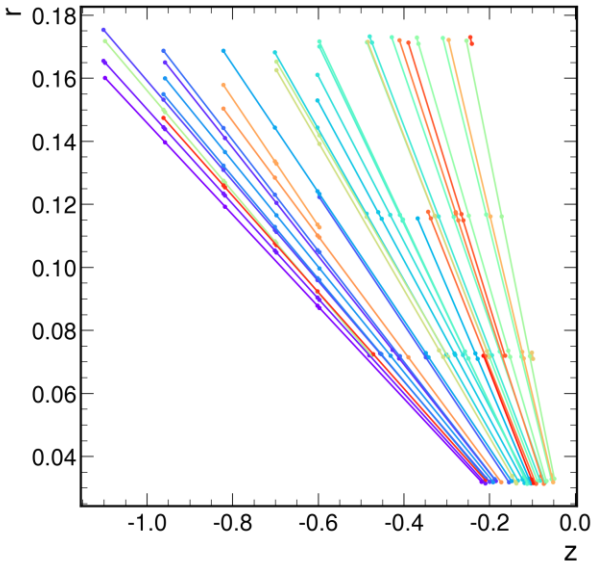


3) Postprocessing

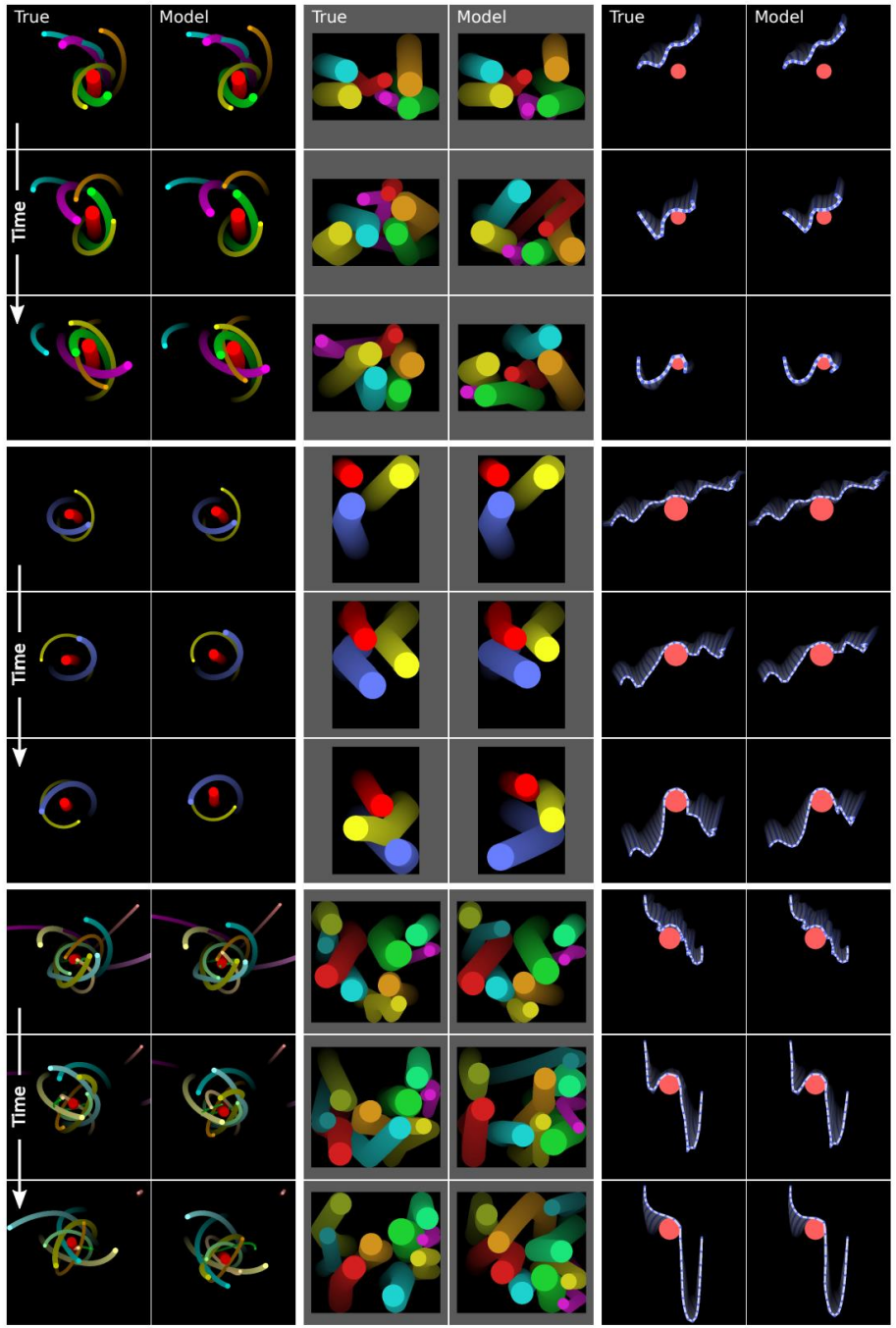
Cluster hits in the learned coordinate space to form track candidates!



GNN outputs 2D coordinates for each hit



Colors are cluster labels generated by DBSCAN (not the GNN)



Interaction Networks:

[1612.00222] Interaction Networks for Learning about Objects, Relations and Physics (arxiv.org)

Physics-motivated MPNNs suitable for graphs with pre-constructed edges (originally applied to "next timestep" physics simulations)

- (**Edge Block**) compute an interaction between two entities:

$$e_{uv}^{(k)} = MLP_{\psi}^{(k)}\left(\left[h_u^{(k-1)}, h_v^{(k-1)}, e_{u,v}^{(k-1)}\right]\right)$$

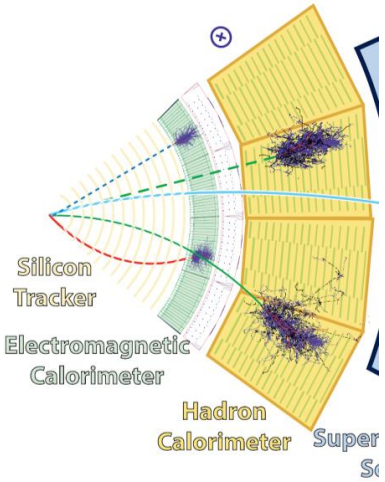
- (**Node Block**) use the interaction to update the state of the receiving node:

$$h_u^{(k)} = MLP_{\phi}^{(k)}\left(h_u^{(k-1)}, \sum_{v \in N(u)} e_{u,v}^{(k)}\right)$$

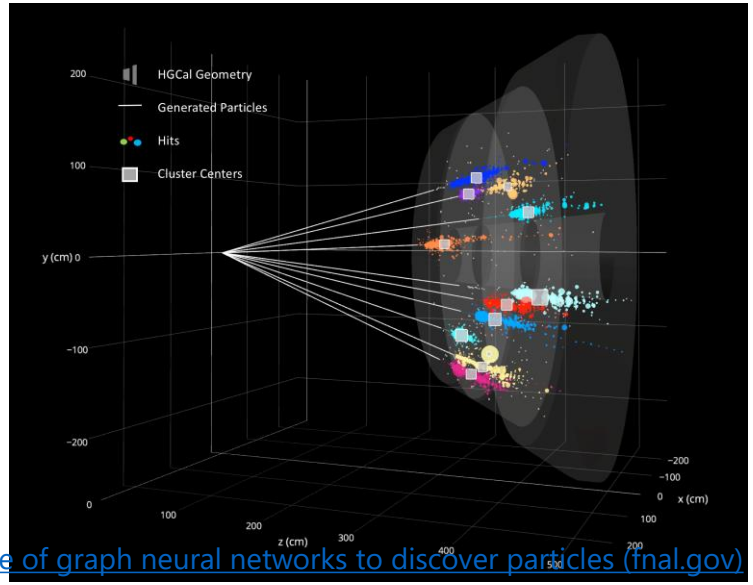
a common form for many GNN tracking architectures
1810.06111.pdf (arxiv.org)

Calorimeters are designed to absorb particles by forcing them to shower

Calorimeter showers allow us to measure *position, ID,* and *energy*



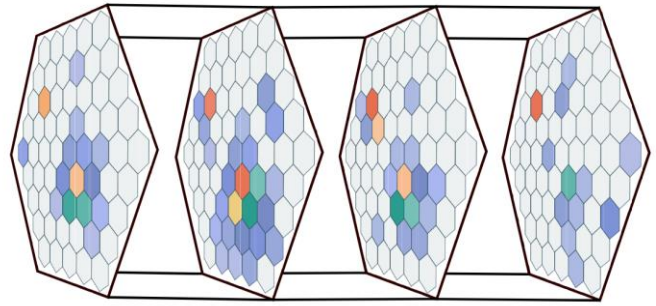
Challenge: disentangling showers, using them to predict energy / particle ID



[The next big thing: the use of graph neural networks to discover particles (fnal.gov)]

Calorimeter Segmentation Tasks

Input data is a set of calorimeter hit cells (features are e.g. energy, position) → embed as graph nodes



- **Edge Classification**: pre-construct a graph and classify edges to form a mesh on the calorimeter hits representing the particle shower
- **Node Classification:** separate two calorimeter showers by predicting the fractional assignment of each hit
- **Object Condensation:** *see next slides*

GNN CALORIMETRY

OVERVIEW / TASKS

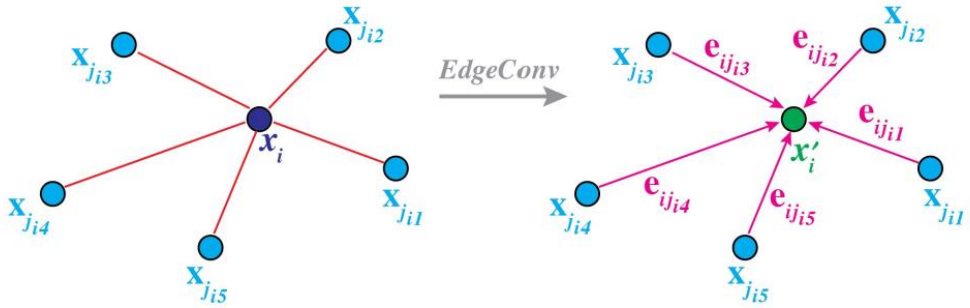
GNNS AT THE LHC
IMPLICIT GRAPH LEARNING

Dynamic Graph Construction
In many cases it is preferable *not* to precompute edges and, instead, form them as part of the learning algorithm

e.g. EdgeConv GNN Layers
During inference, draw edges between nodes clustered by *k-NN;* use these edges for subsequent message passing → local graph creation

$$h_u^{(k)} = \max_{v \in N(u)} MLP_\phi^{(k)}(h_u^{(k-1)}, h_v^{(k-1)} - h_u^{(k-1)})$$

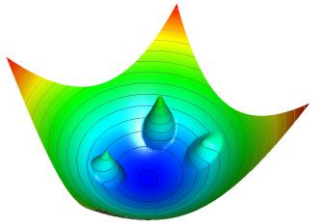
global local



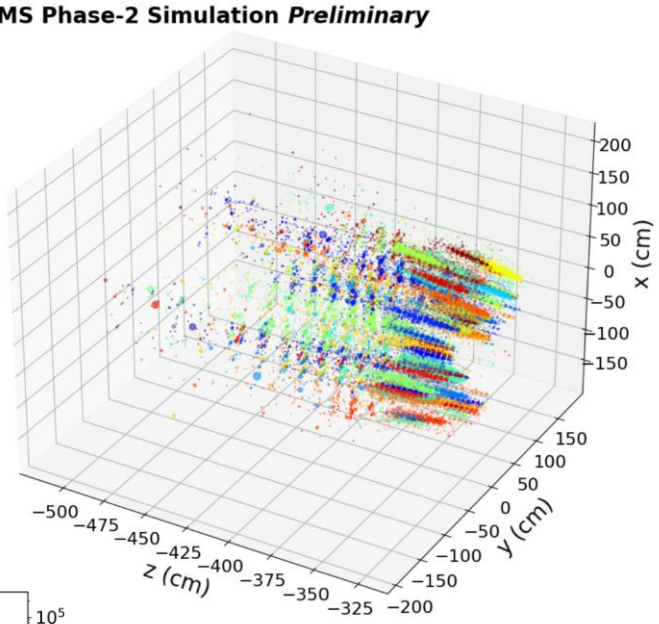
A lightweight version of this operation (GravNet) has been developed at the LHC and applied to calorimeter segmentation!

Dynamic Graph CNN for Learning on Point Clouds (arxiv.org)
Gravnet: 1902.07987.pdf (arxiv.org)

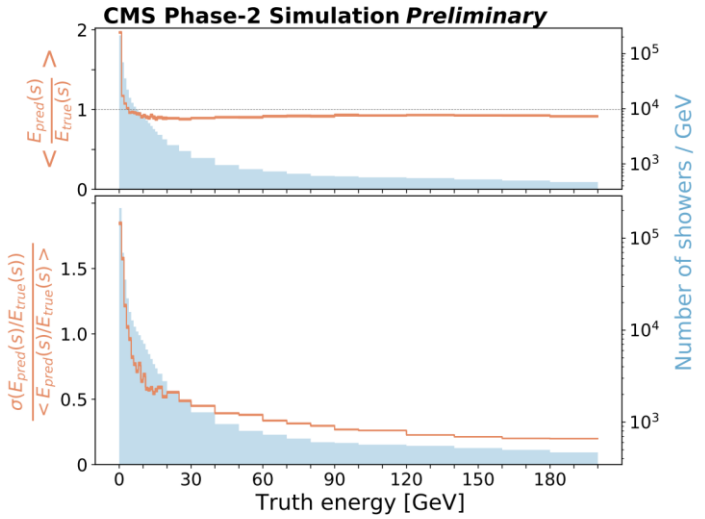
Object Condensation: a GNN learning strategy designed to 1) cluster nodes belonging to the same object (**segmentation**), 2) suppress noise, and 3) predict the properties of objects formed by graph nodes (**regression**)



Showers are well-segmented



- Applied GravNet layers to perform object condensation on calorimeter data and subsequent energy regression
- Up to 400 particle showers (each with up to 1600 hits) are considered per event
- GNN based on GravNet, a lightweight EdgeConv layer performing dynamic graph construction



Predicted shower energies match truth

[Object Condensation 2002.03605.pdf (arxiv.org)]

[Multi-particle reconstruction in the High Granularity Calorimeter using object condensation and graph neural networks]

GNN CALORIMETERY

OBJECT CONDENSATION EXAMPLE

GNN EVENT RECONSTRUCTION

MLPF EXAMPLE

Particle flow (PF) algorithms combine tracks and calorimeter clusters to form physics objects (electrons, photons, muons, etc.)

PF via Node Classification:
A recent GNN-based PF algorithm considered *heterogeneous nodes* representing tracks, electromagnetic calorimeter (ECAL) clusters, and hadronic calorimeter (HCAL) clusters

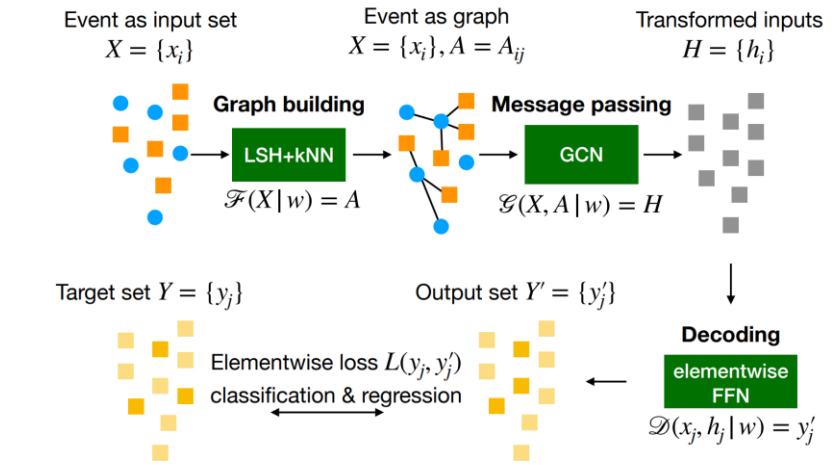
- Targets (per node):

$$y_j = [\text{PID}, p_\text{T}, E, \eta, \phi, q]$$
$$\text{PID} \in \{\text{charged hadron}, \text{neutral hadron}, \gamma, e^\pm, \mu^\pm\}$$

- Nodes:

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

[2101.08578] MLPF: Efficient machine-learned particle-flow reconstruction using graph neural networks (arxiv.org)



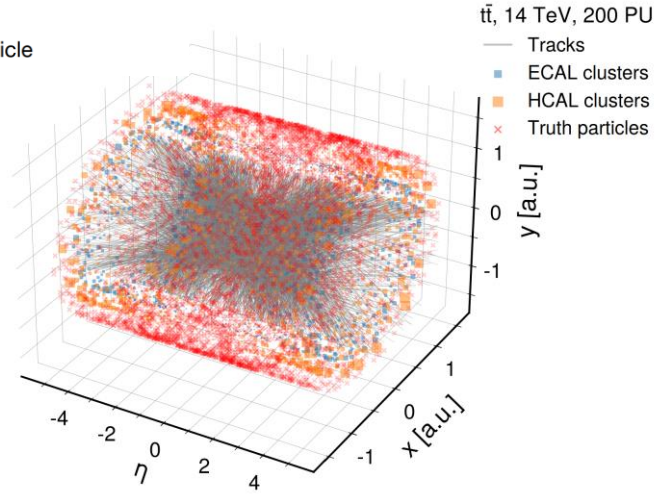
$$x_i = [\text{type}, p_\text{T}, E_\text{ECAL}, E_\text{HCAL}, \eta, \phi, \eta_\text{outer}, \phi_\text{outer}, q, \ldots], \quad \text{type} \in \{\text{track}, \text{cluster}\}$$
$$y_j = [\text{PID}, p_\text{T}, E, \eta, \phi, q, \ldots], \quad \text{PID} \in \{\text{none}, \text{charged hadron}, \text{neutral hadron}, \gamma, e^\pm, \mu^\pm\}$$
$$h_i \in \mathbb{R}^{256}$$

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

● - track, ■ - calorimeter cluster, ■ - encoded element
■ - target (predicted) particle, ■ - no target (predicted) particle

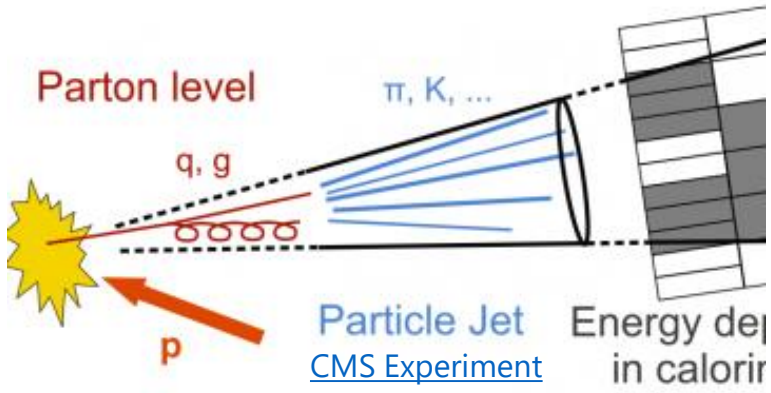
Segmentation, classification, and regression → multi-objective learning (one-shot)

Graph structure computed dynamically, not trained explicitly; ~5000 elements per graph

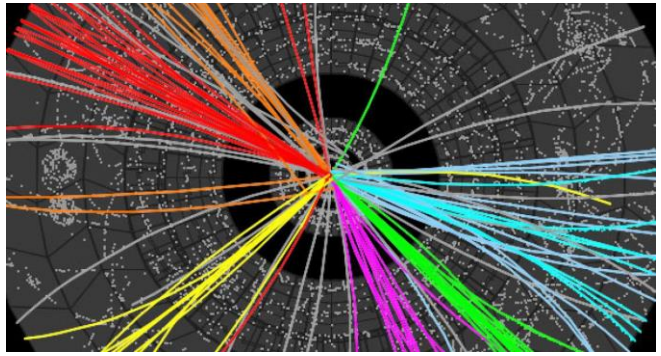


Jets are collimated sprays of particles produced when quarks or gluons are produced in isolation

Correctly reconstructing and classifying jets is critical for many important physics measurements (e.g. Higgs → b quarks)

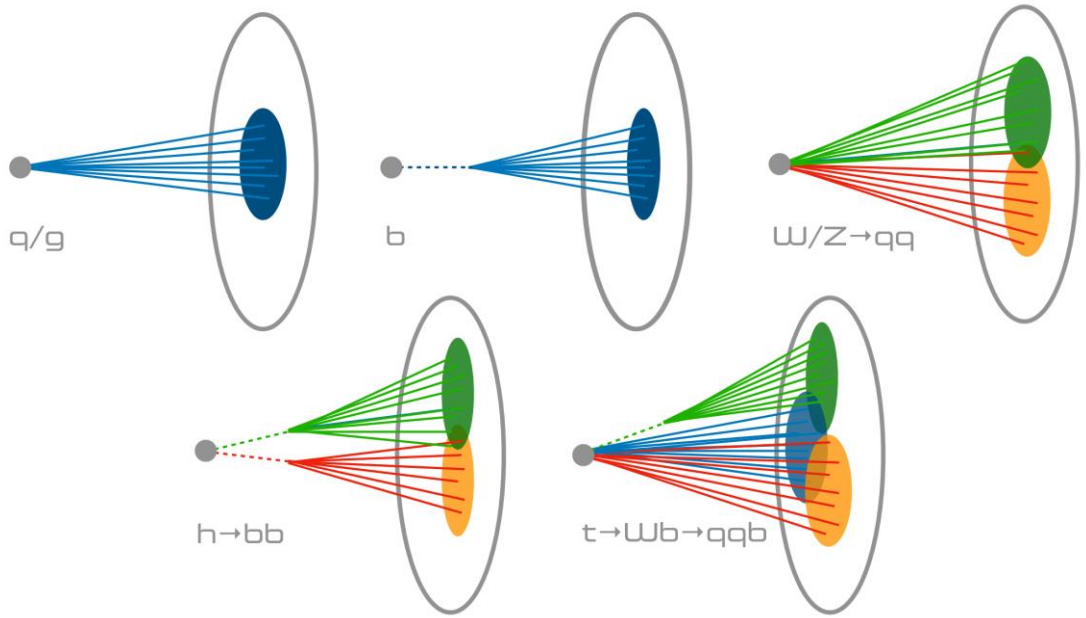


CMS Experiment



O(1) to O(10) jets produced in each particle event

Jet Identification: what particle initiated the jet?



Isolated quarks and gluons form collimated jets; heavier objects decaying to multiple quarks that are reconstructed into larger jets

GNN JET IDENTIFICATION

JETS AND PROBLEM DESCRIPTION

GNN JET IDENTIFICATION
SET-BASED APPROACHES

Set-based Architectures
Treat jets as *sets* of particles with kinematic features (energy, momentum, direction, mass)

- **Particle/Energy Flow Networks** apply the DeepSets result directly:

Observable Decomposition. *An observable $\mathcal{O}$ can be approximated arbitrarily well as:*

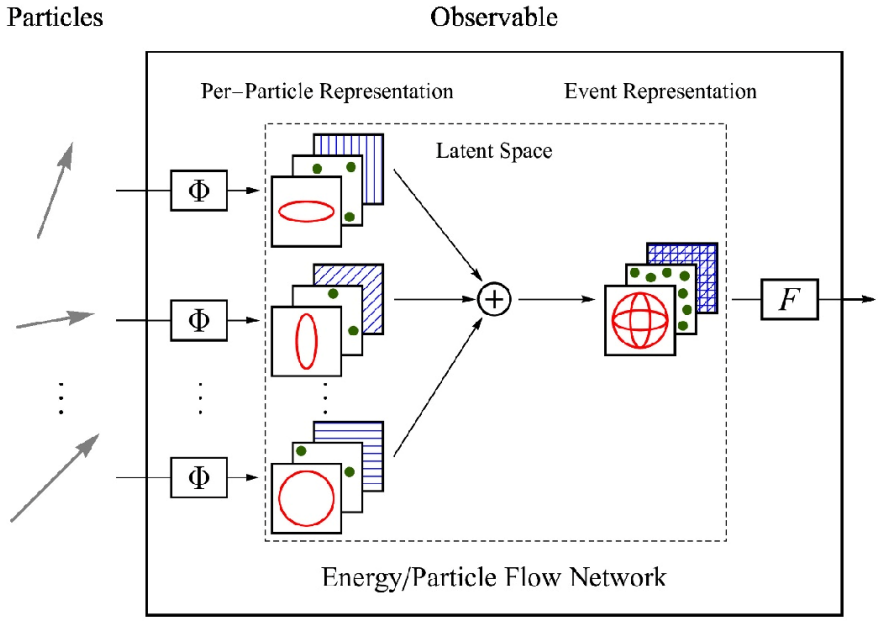
$$\mathcal{O}(\{p_1, \ldots, p_M\}) = F\left(\sum_{i=1}^{M} \Phi(p_i)\right), \tag{1.1}$$

where $\Phi : \mathbb{R}^d \to \mathbb{R}^\ell$ is a per-particle mapping and $F : \mathbb{R}^\ell \to \mathbb{R}$ is a continuous function.

- **Direct Application**:

$$\text{PFN:} \quad F\left(\sum_{i=1}^{M} \Phi(p_i)\right)$$

any corresponding particle information



[1810.05165] Energy Flow Networks: Deep Sets for Particle Jets (arxiv.org)

GNN JET IDENTIFICATION

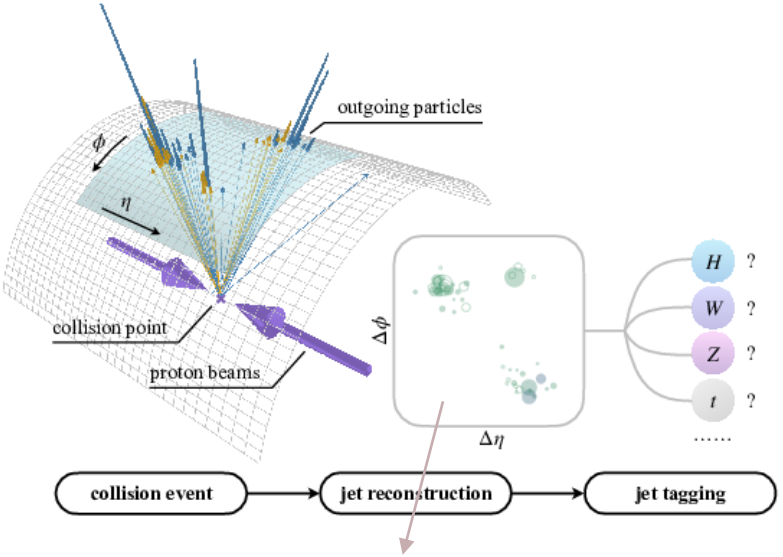
PARTICLE CLOUDS / PARTICLE GRAPHS

Particle Cloud GNNs

Apply GNNs w/ dynamic graph construction (EdgeConv) to make predictions on point clouds

e.g.

[1902.08570] ParticleNet: Jet Tagging via Particle Clouds (arxiv.org)



Particles can be viewed as a “particle cloud” of kinematic features at different spatial locations

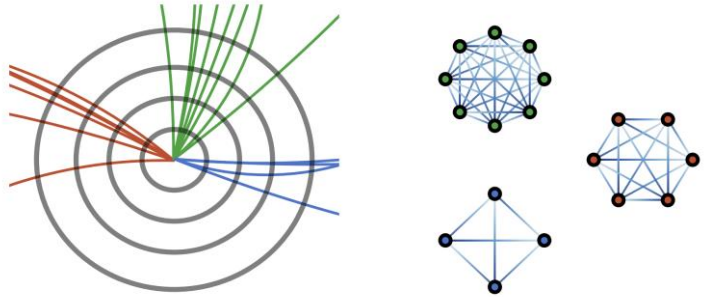
Particle Graph GNNs

Apply a GNN to a pre-constructed graph with particles as nodes

e.g.

[2001.05311] ABCNet: An attention-based method for particle tagging (arxiv.org)

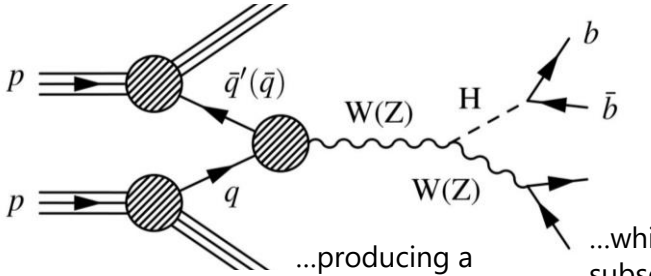
[1908.05318] JEDI-net: a jet identification algorithm based on interaction networks (arxiv.org)



In this case, the particles comprising three jets have been embedded as nodes in fully connected graphs

Signals are collections of particles (topology + kinematics) produced by an interesting physics process

What happened:

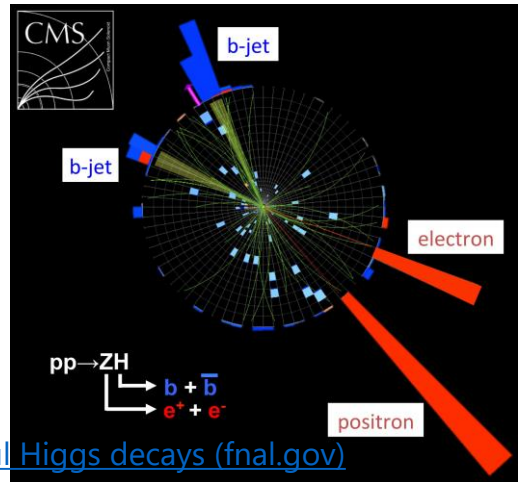


protons collide

...producing a Higgs and associated vector boson (W or Z)

...which subsequently decay into observable particles

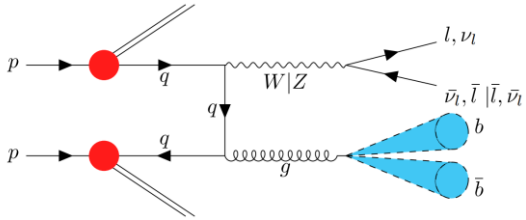
What we see:

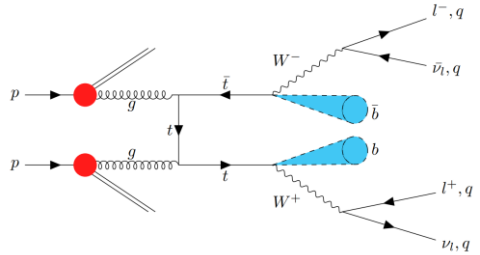


Beautiful Higgs decays (fnal.gov)

Signal Identification: is this event really the signal we're looking for?

Background Rejection: lots of other processes look like the signal, and are often produced at a much higher rate





Associated Production with a vector boson and decay into b-quarks using the ATLAS Run-2 dataset (Dwayne Spiteri) (inspirehep.net)

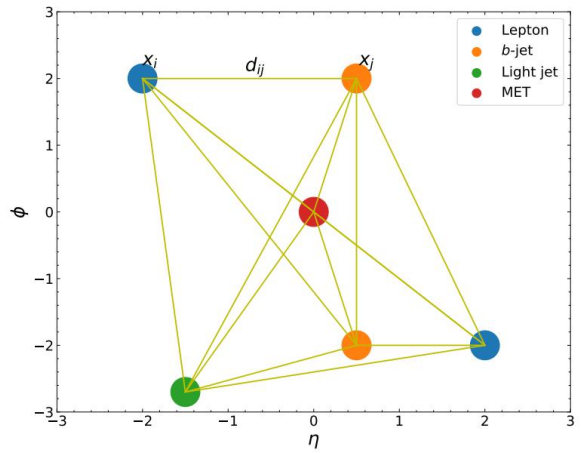
GNN SIGNAL IDENTIFICATION

PHYSICS SIGNALS

GNN JET IDENTIFICATION

EVENT-LEVEL TASKS

Event (Graph-Level) Classification: *heterogeneous* nodes representing different particles / observables



x	photon	lepton charge	b-jet or light jet	MET	p_T (TeV)	E (TeV)	m (TeV)
1	0	1	0	0	0.132	0.135	0.000
2	0	−1	0	0	0.025	0.025	0.000
3	0	0	1	0	0.163	0.227	0.012
4	0	0	1	0	0.052	0.053	0.006
5	0	0	−1	0	0.047	2.485	0.011
6	0	0	0	1	0.078	0.078	0.000

nodes have explicit particle labels
→ heterogeneous structure

Use MPNNs to update node states to “agree” (global average pool them) on a graph-level classification of signal vs. background

[2005.11086.pdf (arxiv.org)]

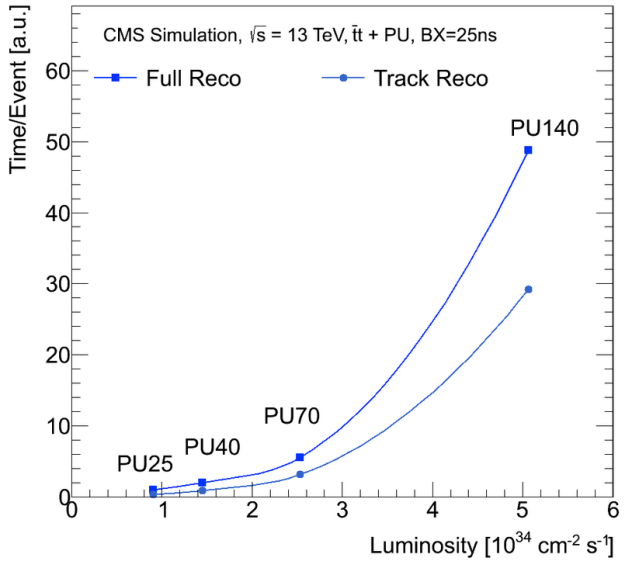
CONCLUSIONS

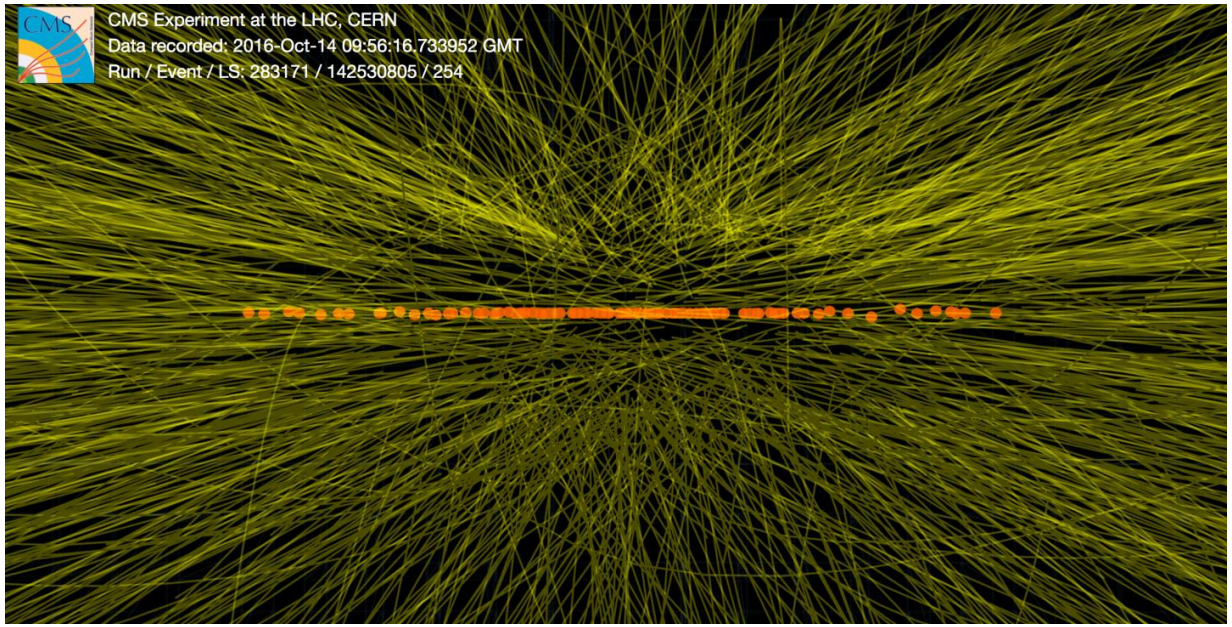
- GNNs are being applied to a wide range of physics tasks at the LHC
  - Track and calorimeter reconstruction, energy regression, particle and signal identification
  - Topics not covered here: generative modeling, anomaly detection, detector calibration, algorithmic acceleration, interpretability
- A variety of graph-based learning approaches:
  - Pre-constructed vs. dynamically computed graphs
  - Node, edge, and graph-level predictions
  - One-shot (multi-objective) learning functions
  - Heterogeneous graph nodes
- GNN layers are quickly becoming one of the "standard building blocks" for LHC architectures

… thanks for listening!

GNNS AT THE LHC

GNN TRACKING AT THE HL-LHC





Many overlapping interactions → many tracks per event!

GNNS AT THE LHC
ACCELERATED GNN TRACKING

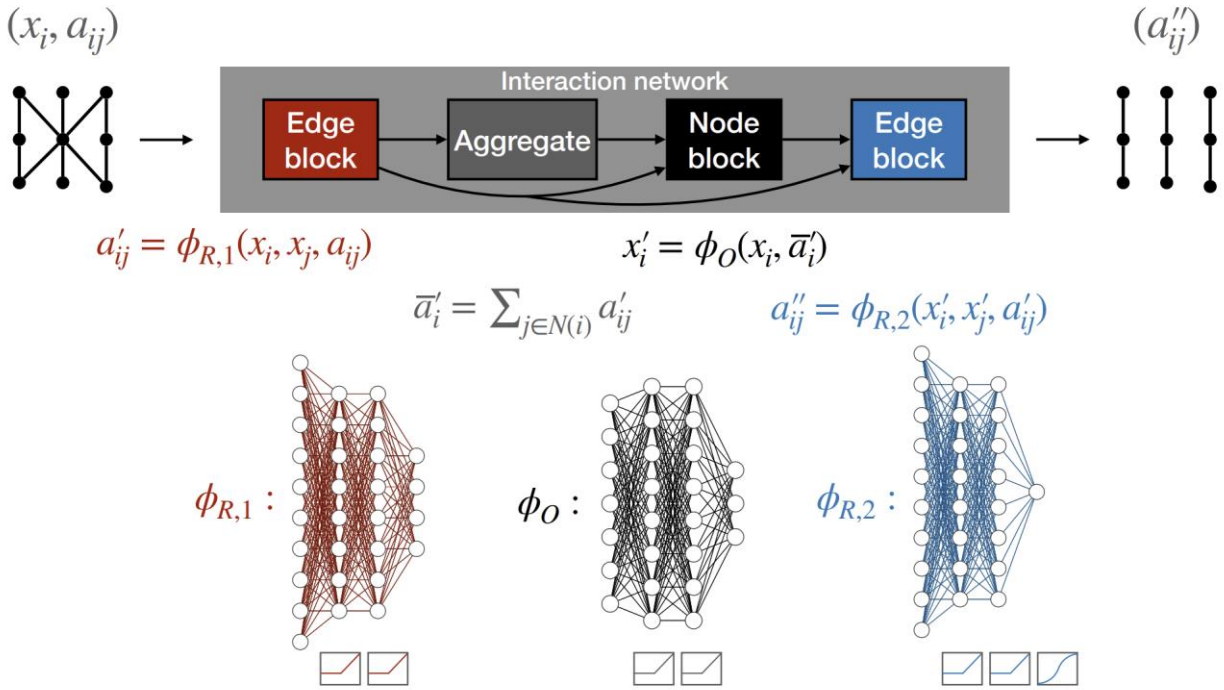
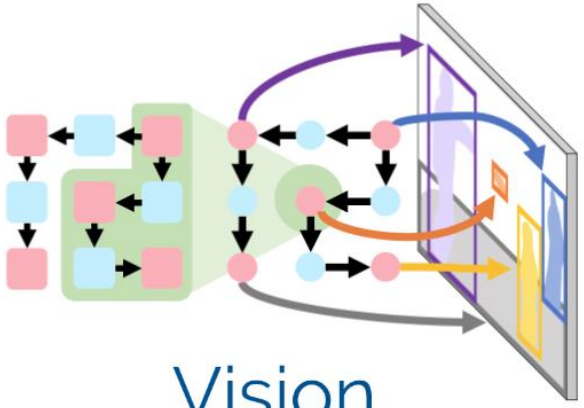


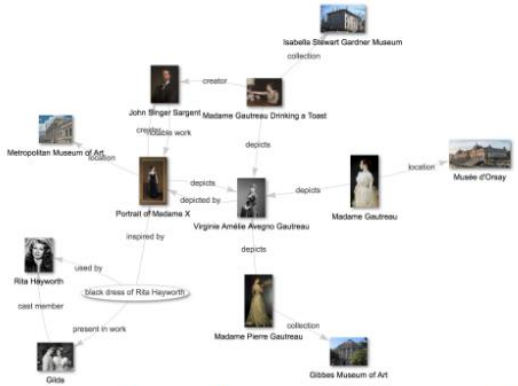
Table 1. Summary of the latency, II, and FPGA resource usage metrics of the synthesized firmware for a variety of design choices. The target FPGA is a Xilinx Virtex UltraScale+ VU9P FPGA (part number `xcvu9p-flga2104-2L-e`), which has 6,840 DSPs, 1,182,240 LUTs, 2,364,480 FFs, and 75.9 Mb of BRAM (Xilinx, Inc., 2021). A 5 ns clock period is used.

Design	$(n_{\text{nodes}}, n_{\text{edges}})$	RF	Precision	Latency [cycles]	II [cycles]	DSP [%]	LUT [%]	FF [%]	BRAM [%]
Throughput-opt.	(28, 56)	1	`ap_fixed<14,7>`	59	1	99.9	66.0	11.7	0.7
Throughput-opt.	(28, 56)	8	`ap_fixed<14,7>`	75	8	21.9	23.8	4.7	0.7
Resource-opt.	(28, 56)	1	`ap_fixed<14,7>`	79	28	56.6	17.6	3.9	13.1
Resource-opt.	(448, 896)	1	`ap_fixed<14,7>`	470	174	56.6	25.0	7.4	16.5
Resource-opt.	(448, 896)	8	`ap_fixed<14,7>`	1590	520	5.6	25.0	7.4	16.3

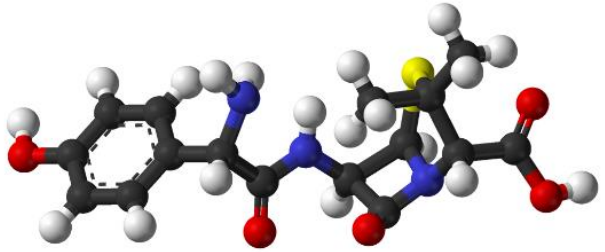
[2112.02048] Graph Neural Networks for Charged Particle Tracking on FPGAs (arxiv.org)



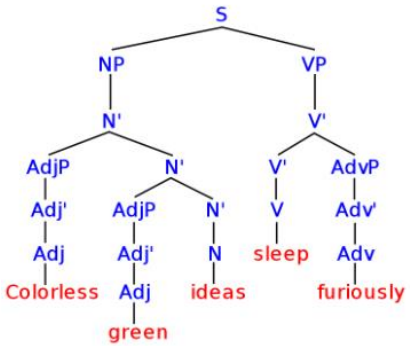
Vision



Knowledge graphs

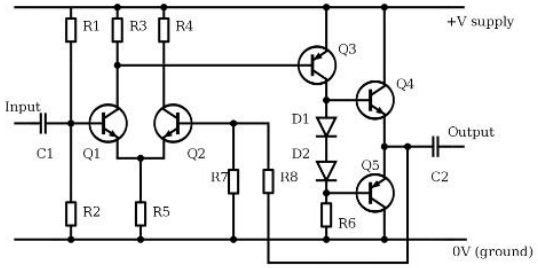


Chemistry



Language

Traffic networks



Circuits

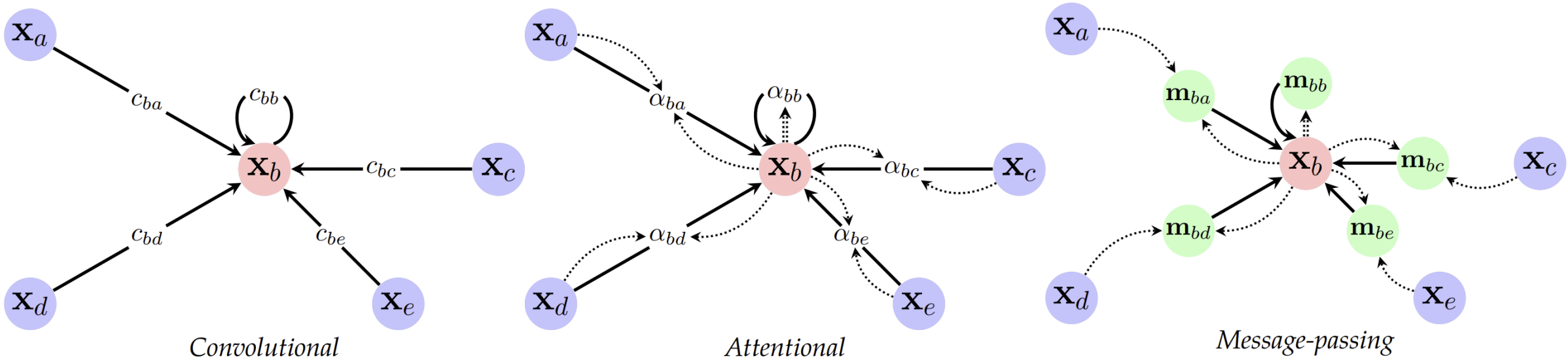
Social networks

GRAPH NEURAL NETWORKS

GRAPH EXAMPLES

Generic MPNN Layers: $$\boldsymbol{h}_u^{(k)} = \phi^{(k)}\left[\boldsymbol{h}_u^{(k-1)}, \bigoplus_{v \in N(u)} \psi^{(k)}\left(\boldsymbol{h}_u^{(k-1)}, \boldsymbol{h}_v^{(k-1)}, \boldsymbol{e}_{uv}^{(k-1)}\right)\right]$$

Different Equivariant Node Updates:



2104.13478.pdf (arxiv.org)

GRAPH NEURAL NETWORKS

DIFFERENT FLAVORS

EDGE CLASSIFICATION
BINARY CROSS ENTROPY

Edge Classifier
(updates edge features)

Node Features: $x_i = (r_i, \phi_i, z_i)$

Edge Features:
$\tilde{a}_{ij} = (w_{ij}, \Delta r_{ij}, \Delta\phi_{ij}, \Delta\eta_{ij}, \Delta R_{ij})$

Opacity ~ Edge Score (w_{ij})

- Optimize BCE as usual:

$$\mathcal{L}_w(y_j, w_j) = -\sum_{j=1}^{|\mathcal{E}|} \left(y_j \log w_j + (1 - y_j) \log(1 - w_j)\right)$$

```
# re-embed the graph twice with add aggregation
x1, edge_attr_1 = self.in_w1(x, edge_index, edge_attr)
x2, edge_attr_2 = self.in_w2(x1, edge_index, edge_attr_1)

# combine all edge features, use to predict edge weights
initial_edge_attr = torch.cat([edge_attr, edge_attr_1,
                               edge_attr_2], dim=1)
edge_weights = torch.sigmoid(self.w(initial_edge_attr))
```

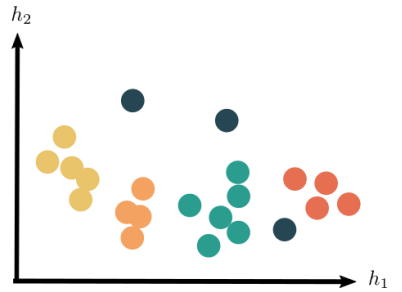
OBJECT CONDENSATION

POTENTIAL LOSS + BACKGROUND SUPPRESSION

Object Condensation
(coordinates in learned clustering space)

New Coordinates: $h_i \in \mathbb{R}^{d_{\text{out}}}$

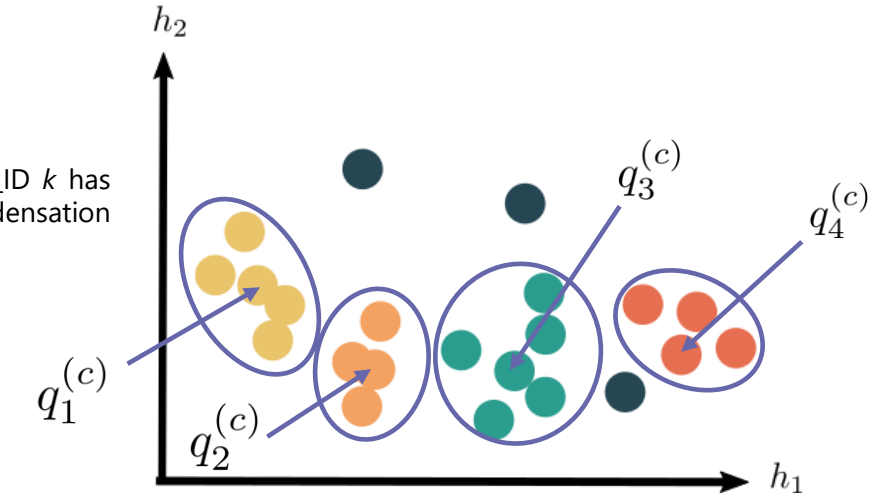
Condensation Strength: $\beta_i \in (0, 1)$



- **Predict** condensation "likelihood" ($\beta_i \in (0, 1)$) and learned clustering coordinates ($h_i \in \mathbb{R}^{d_h}$)
- Define a charge per node: $q_i = \text{arctanh}^2 \beta_i + q_{\min}$
  - Condensation points: $q_k^{(c)} = \max_i q_i \, \mathbb{1}_{\{l_i = k\}}$

Indicator function: the i^{th} node's particle_ID (l_i) must be k

Each particle_ID k has a single condensation point



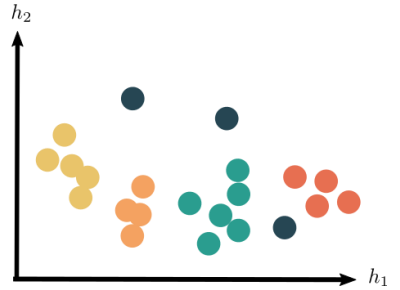
OBJECT CONDENSATION

POTENTIAL LOSS + BACKGROUND SUPPRESSION

Object Condensation
(coordinates in learned clustering space)

New Coordinates: $h_i \in \mathbb{R}^{d_{\text{out}}}$

Condensation Strength: $\beta_i \in (0, 1)$

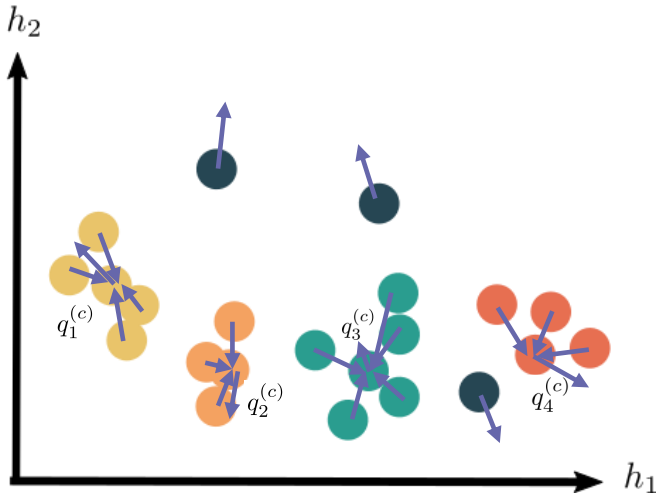


- Condensation points: $q_k^{(c)} = \max_i q_i \, \mathbb{1}_{\{l_i=k\}}$
- Optimize two terms:
  - Potential Loss:

$$\mathcal{L}_V = \frac{1}{|\mathcal{V}|} \sum_{i=1}^{|\mathcal{V}|} q_i \sum_{k=1}^{K} \left(\mathbb{1}_{(l_i=k)} V_k^{\text{attract}}(h_i) + (1 - \mathbb{1}_{(l_i=k)}) V_k^{\text{repulse}}(h_i) \right)$$

$$V_k^{\text{attract}}(h) = ||h - h_\alpha||_2^2 \, q_k^{(c)} \quad V_k^{\text{repulse}}(h) = \max(0, 1 - ||h - h_\alpha||) q_k^{(c)}$$

Nodes are attracted to their particle's condensation point and repulsed from other particles'



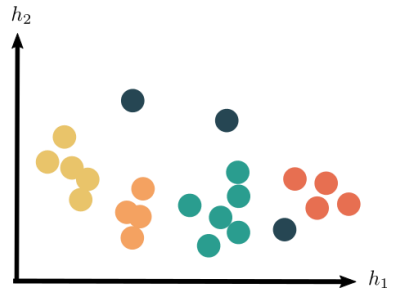
OBJECT CONDENSATION

POTENTIAL LOSS + BACKGROUND SUPPRESSION

Object Condensation
(coordinates in learned clustering space)

New Coordinates: $h_i \in \mathbb{R}^{d_{\text{out}}}$

Condensation Strength: $\beta_i \in (0, 1)$



- **Predict** condensation "likelihood" ($\beta_i \in (0, 1)$) and learned clustering coordinates ($h_i \in \mathbb{R}^{d_h}$)
- Define a charge per node: $q_i = \text{arctanh}^2 \beta_i + q_{\min}$
  - Condensation points: $q_k^{(c)} = \max_i q_i \, \mathbb{1}_{\{l_i = k\}}$
- Optimize two terms:
  - Potential Loss:

$$\mathcal{L}_V = \frac{1}{|\mathcal{V}|} \sum_{i=1}^{|\mathcal{V}|} q_i \sum_{k=1}^{K} \left(\mathbb{1}_{(l_i=k)} V_k^{\text{attract}}(h_i) + \left(1 - \mathbb{1}_{(l_i=k)}\right) V_k^{\text{repulse}}(h_i) \right)$$

$$V_k^{\text{attract}}(h) = ||h - h_\alpha||_2^2 \, q_k^{(c)} \qquad V_k^{\text{repulse}}(h) = \max(0, 1 - ||h - h_\alpha||) q_k^{(c)}$$

  - Background Loss:

$$\mathcal{L}_\beta = \frac{1}{K} \sum_k (1 - \beta_k^{(c)}) + s_B \frac{\sum_{i=1}^{|\mathcal{V}|} \beta_i \mathbb{1}_{\{l_i=0\}}}{\sum_{i=1}^{|\mathcal{V}|} \mathbb{1}_{\{l_i=0\}}}$$

Sum over noise hits (here, any particle hitting only one detector layer – O(20-50) hits)

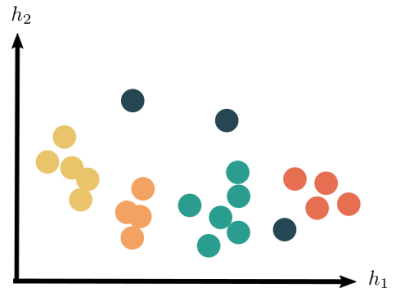
OBJECT CONDENSATION

POTENTIAL LOSS + BACKGROUND SUPPRESSION

Object Condensation
(coordinates in learned clustering space)

New Coordinates: $h_i \in \mathbb{R}^{d_{\text{out}}}$

Condensation Strength: $\beta_i \in (0, 1)$



- Predict condensation likelihood ($\beta_i \in (0, 1)$) and learned clustering coordinates ($h_i \in \mathbb{R}^{d_h}$)
- Define a charge per node: $q_i = \text{arctanh}^2 \beta_i + q_{\min}$
  - Condensation points: $q_k^{(c)} = \max_i q_i \, \mathbb{1}_{\{l_i = k\}}$
- Optimize two terms:
  - Potential Loss:

$$\mathcal{L}_V = \frac{1}{|\mathcal{V}|} \sum_{i=1}^{|\mathcal{V}|} q_i \sum_{k=1}^{K} \left(\mathbb{1}_{(l_i=k)} V_k^{\text{attract}}(h_i) + 10\times (1 - \mathbb{1}_{(l_i=k)}) V_k^{\text{repulse}}(h_i) \right)$$

$$V_k^{\text{attract}}(h) = ||h - h_\alpha||_2^2 \, q_k^{(c)} \qquad V_k^{\text{repulse}}(h) = \max(0, 1 - ||h - h_\alpha||) q_k^{(c)}$$

  - Background Loss:

$$\mathcal{L}_\beta = \frac{1}{K} \sum_k (1 - \beta_k^{(c)}) + s_B \frac{\sum_{i=1}^{|\mathcal{V}|} \beta_i \mathbb{1}_{\{l_i=0\}}}{\sum_{i=1}^{|\mathcal{V}|} \mathbb{1}_{\{l_i=0\}}}$$

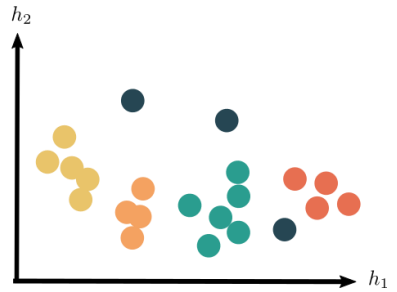
OBJECT CONDENSATION

POTENTIAL LOSS + BACKGROUND SUPPRESSION

Object Condensation
(coordinates in learned clustering space)

New Coordinates: $h_i \in \mathbb{R}^{d_{\text{out}}}$

Condensation Strength: $\beta_i \in (0, 1)$



- Optimize two terms:

$$\mathcal{L}_V = \frac{1}{|\mathcal{V}|} \sum_{i=1}^{|\mathcal{V}|} q_i \sum_{k=1}^{K} \left(\mathbb{1}_{(l_i=k)} V_k^{\text{attract}}(h_i) + \left(1 - \mathbb{1}_{(l_i=k)}\right) V_k^{\text{repulse}}(h_i) \right)$$

$$\mathcal{L}_\beta = \frac{1}{K} \sum_k (1 - \beta_k^{(c)}) + s_B \frac{\sum_{i=1}^{|\mathcal{V}|} \beta_i \mathbb{1}_{\{l_i=0\}}}{\sum_{i=1}^{|\mathcal{V}|} \mathbb{1}_{\{l_i=0\}}}$$

- Architecture:

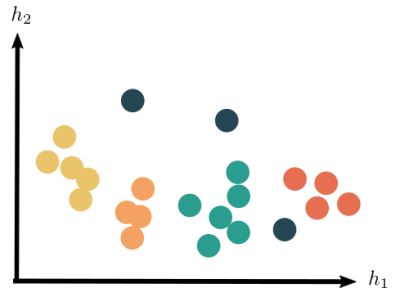
```
# combine edge weights with original edge features
edge_attr_w = torch.cat([edge_weights,
                         initial_edge_attr], dim=1)
xc1, edge_attr_c1 = self.in_c1(x, edge_index, edge_attr_w)
xc2, edge_attr_c2 = self.in_c2(xc1, edge_index, edge_attr_c1)
xc3, edge_attr_c3 = self.in_c3(xc2, edge_index, edge_attr_c2)
all_xc = torch.cat([x,xc1,xc2,xc3], dim=1)
beta = torch.sigmoid(self.B(all_xc))
xc = self.X(all_xc)
return edge_weights, xc, beta
```

TOTAL LOSS
SUM OF PARTS

Object Condensation
(coordinates in learned clustering space)

New Coordinates: $h_i \in \mathbb{R}^{d_{\text{out}}}$

Condensation Strength: $\beta_i \in (0, 1)$



- Three terms:
  - Edge Classification

$$\mathcal{L}_w(y_j, w_j) = -\sum_{j=1}^{|\mathcal{E}|} \left(y_j \log w_j + (1 - y_j) \log(1 - w_j)\right)$$

  - Attraction/Repulsion → scale by 10^4

$$\mathcal{L}_V = \frac{1}{|\mathcal{V}|} \sum_{i=1}^{|\mathcal{V}|} q_i \sum_{k=1}^{K} \left(\mathbb{1}_{(l_i=k)} V_k^{\text{attract}}(h_i) + \left(1 - \mathbb{1}_{(l_i=k)}\right) V_k^{\text{repulse}}(h_i) \right)$$

  - Background Suppression → scale by 2.5×10^{-3}

$$\mathcal{L}_\beta = \frac{1}{K} \sum_k (1 - \beta_k^{(c)}) + s_B \frac{\sum_{i=1}^{|\mathcal{V}|} \beta_i \mathbb{1}_{\{l_i=0\}}}{\sum_{i=1}^{|\mathcal{V}|} \mathbb{1}_{\{l_i=0\}}}$$

GNNS AT THE LHC

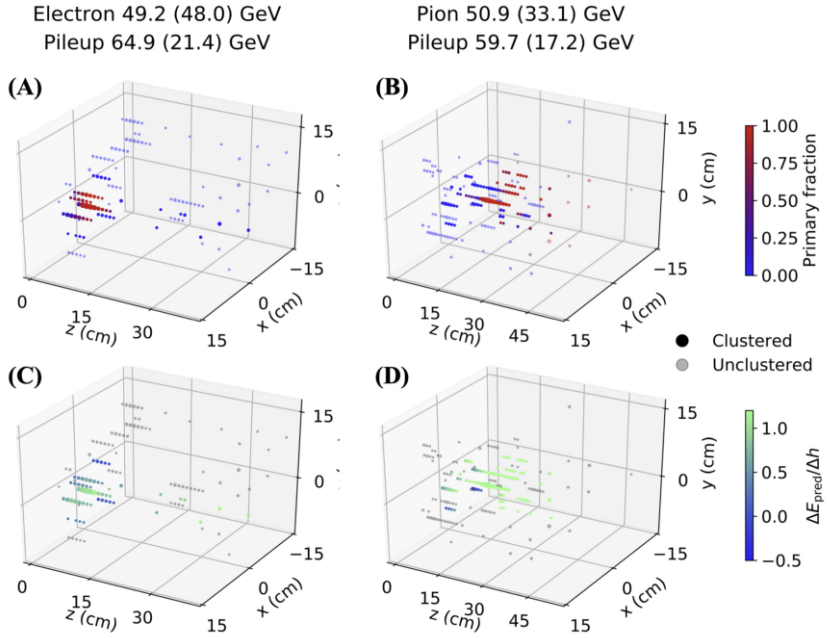
GRAVNET

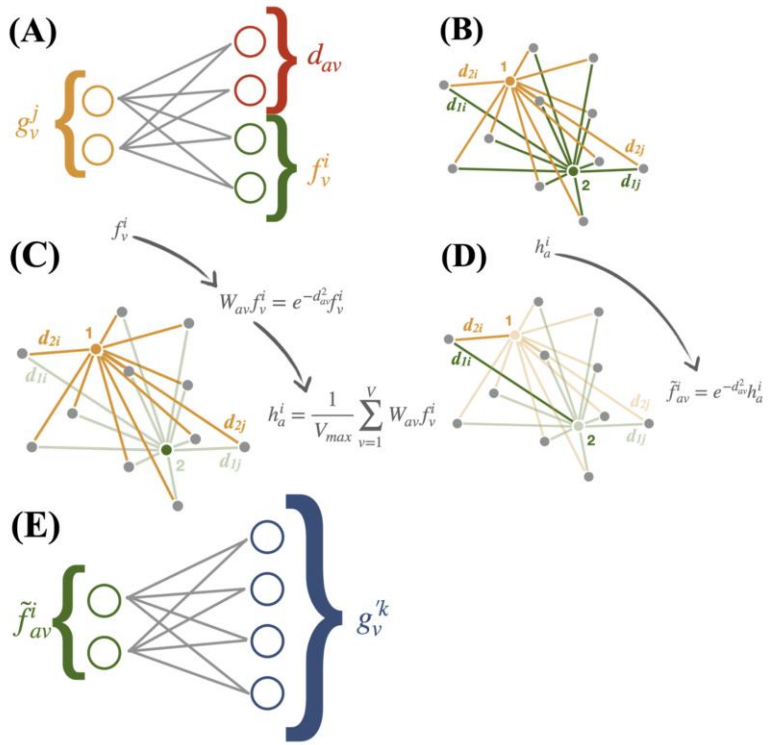
Dynamic Graph Construction

In many cases it is preferable *not* to precompute edges and, instead, form them as part of the learning algorithm

e.g. GravNet/GarNet

Dynamic graph construction in a learned feature space, extract new node features in learned coordinates





[2008.03601.pdf (arxiv.org)]

GNN JET IDENTIFICATION

SET-BASED APPROACHES

Set-based Architectures

Treat jets as *sets* of particles with kinematic features (energy, momentum, direction, mass)

- **Particle/Energy Flow Networks** apply the DeepSets result directly (learnable permutation-invariant approximators with MLPs F, Φ):

Observable Decomposition. *An observable $\mathcal{O}$ can be approximated arbitrarily well as:*

$$\mathcal{O}(\{p_1, \ldots, p_M\}) = F\left(\sum_{i=1}^{M} \Phi(p_i)\right), \tag{1.1}$$

where $\Phi : \mathbb{R}^d \to \mathbb{R}^\ell$ is a per-particle mapping and $F : \mathbb{R}^\ell \to \mathbb{R}$ is a continuous function.

- **Direct Application**:

PFN: $F\left(\sum_{i=1}^{M} \Phi(p_i)\right)$

any corresponding particle information

Specific to Jets:

directional information

EFN: $F\left(\sum_{i=1}^{M} z_i \Phi(\hat{p}_i)\right)$

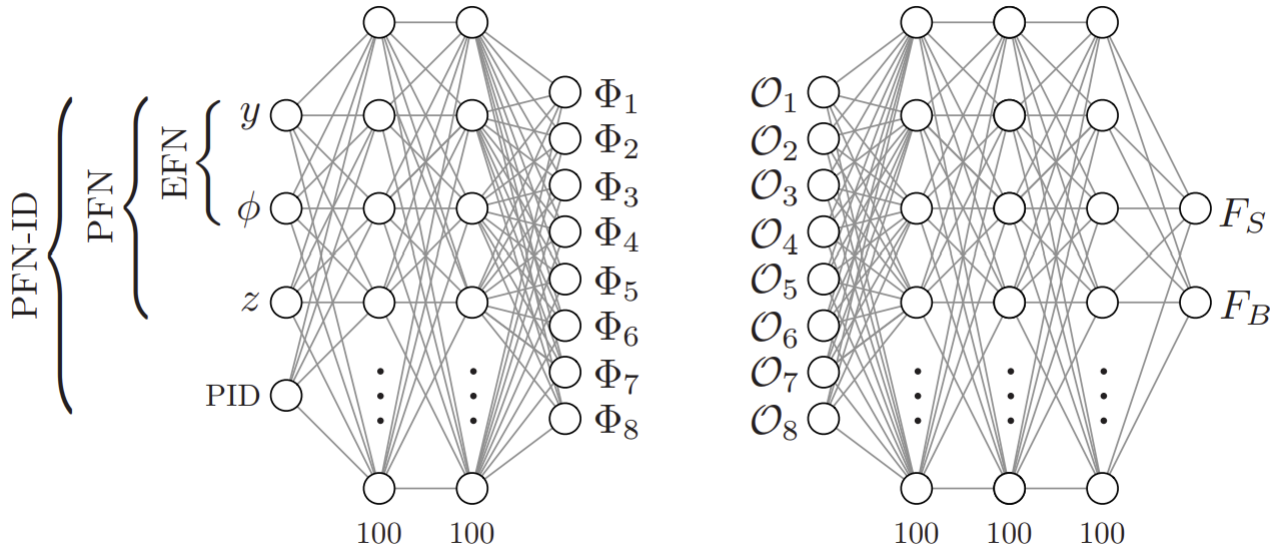
energy weight

[1810.05165] Energy Flow Networks: Deep Sets for Particle Jets (arxiv.org)

GNN JET IDENTIFICATION
SET-BASED APPROACHES

Set-based Architectures
Treat jets as *sets* of particles with kinematic features (energy, momentum, direction, mass)

- **Particle/Energy Flow Networks** apply the DeepSets result directly (learnable permutation-invariant approximators with MLPs F, Φ):



Produce a latent representation of each particle in the set (Φ)

Form an observable from each latent feature by summing over particles… produce classification results

e.g. EFN: $\mathcal{O}_a = \sum_i z_i \, \Phi_a(y_i, \phi_i)$