Accelerated AI
Algorithms for
Data-Driven
Discovery

Low-latency Calorimetry Clustering at the LHC with SPVCNN

ALEX SCHUY[1], ZHIJIAN LIU[2], JEFF KRUPA[2], PATRICK MCCORMACK[2], PHIL HARRIS[2], SHIH-CHIEH HSU[1], SCOTT HAUCK[1], SONG HAN[2]

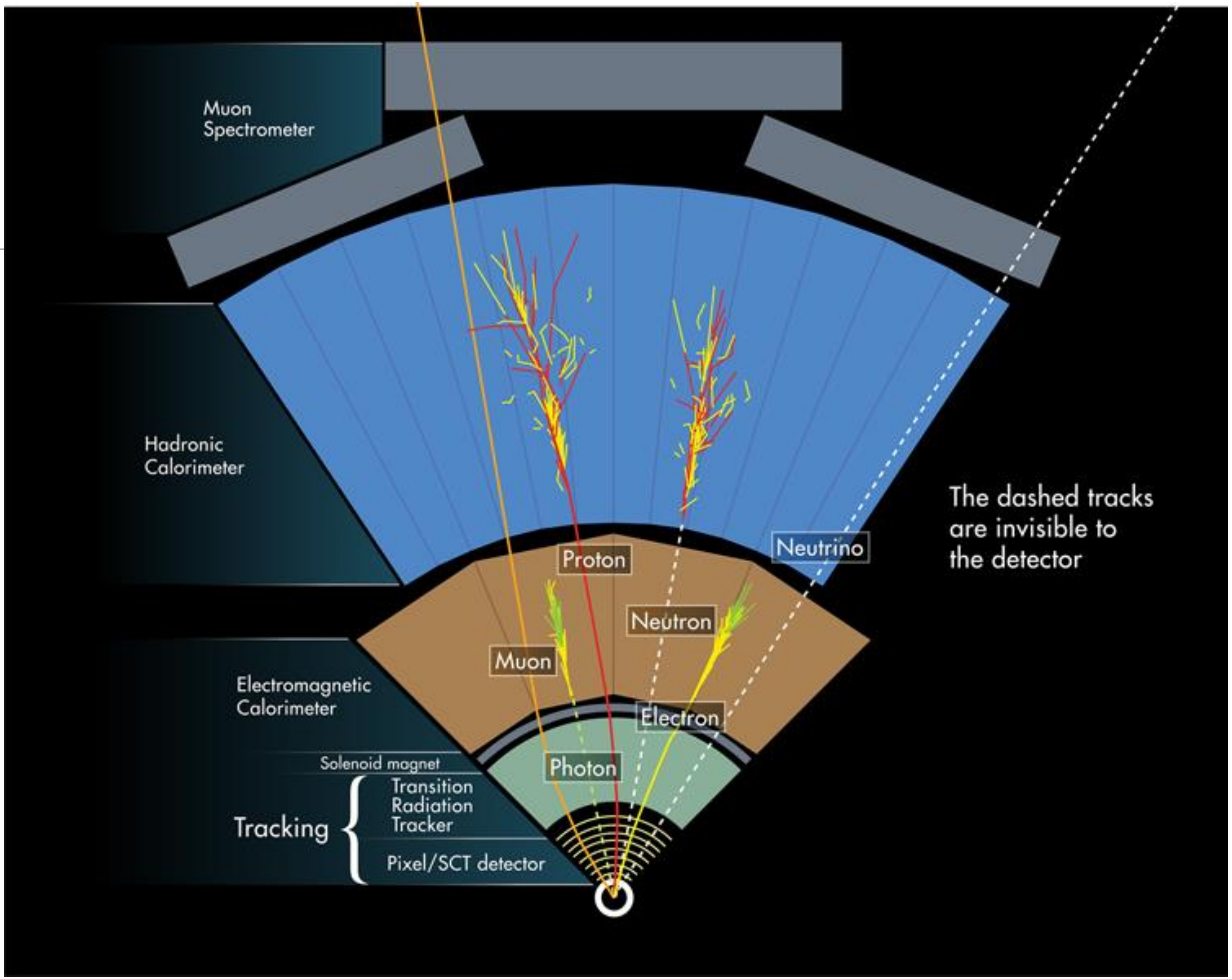
UNIVERSITY OF WASHINGTON[1], MIT[2]

Calorimetry

Physics Event

Goal: record properties of the 'final state' particles produced in a collision:

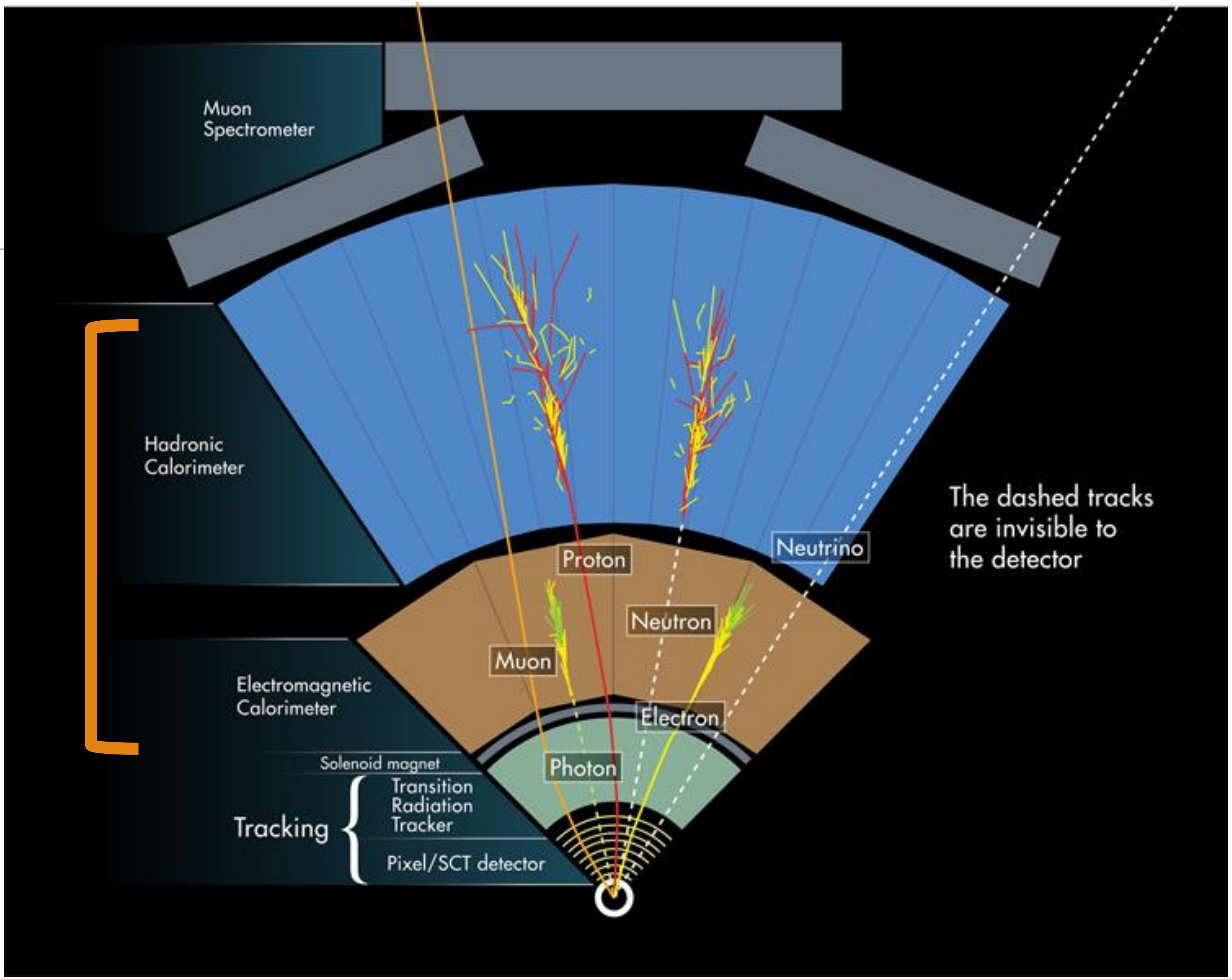
- Type (Proton, Electron, Photon, etc.)
- Energy
- Momenta
- Path through the detector (incl. origin: 'vertex')



Calorimeter

Goal: record properties of the 'final state' particles produced in a collision:

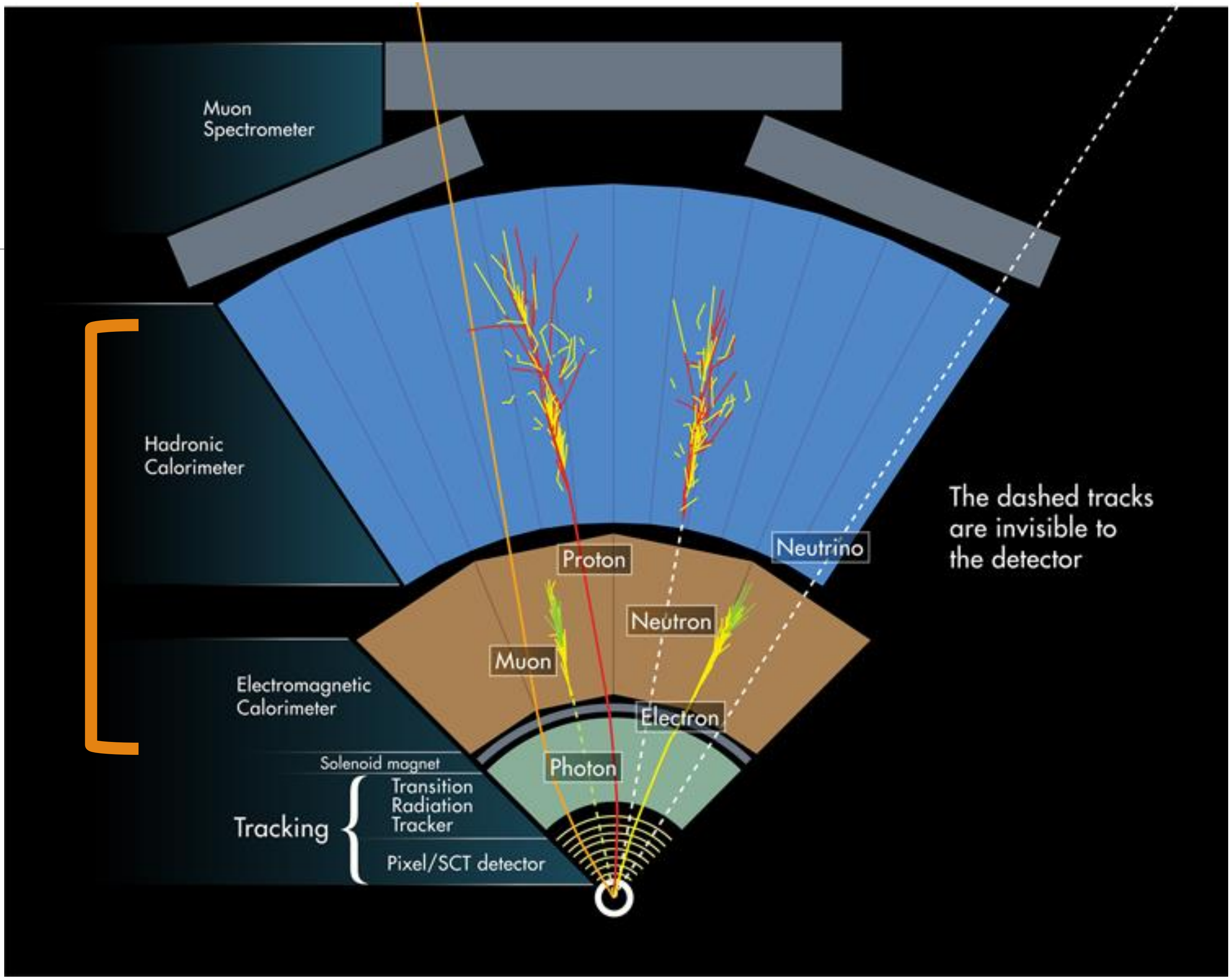
- Type (Proton, Electron, Photon, etc.)
- **Energy**
- Momenta
- Path through the detector (incl. origin: 'vertex')



Calorimeter

Two types of calorimeters:

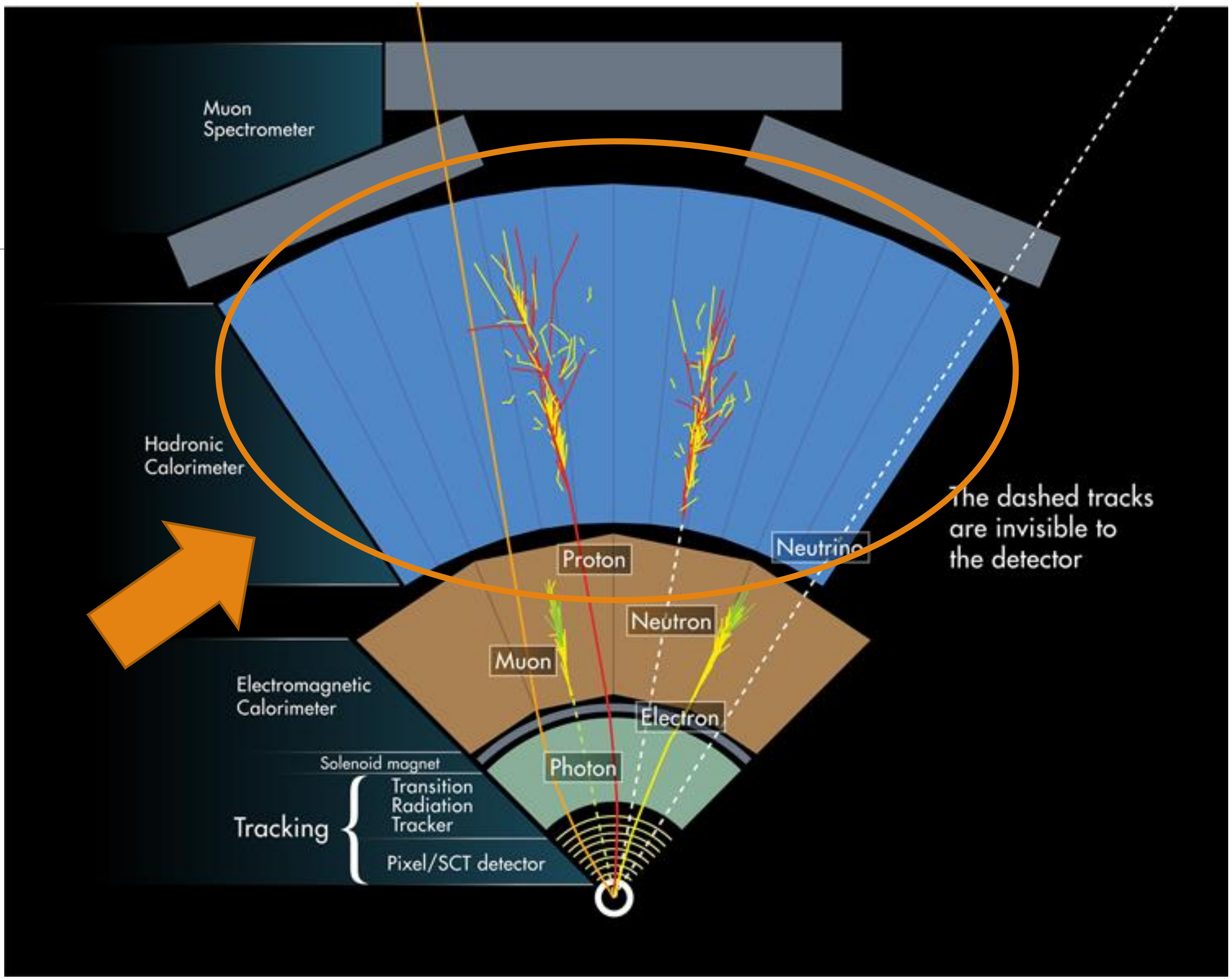
- Electromagnetic
  - Absorbs photons, electrons
- Hadronic
  - Absorbs protons, neutrons, pions, jets, etc.
  - More challenging



Calorimeter

Two types of calorimeters:

- Electromagnetic
  - Absorbs photons, electrons
- **Hadronic**
  - Absorbs protons, neutrons, pions, jets, etc.
  - More challenging



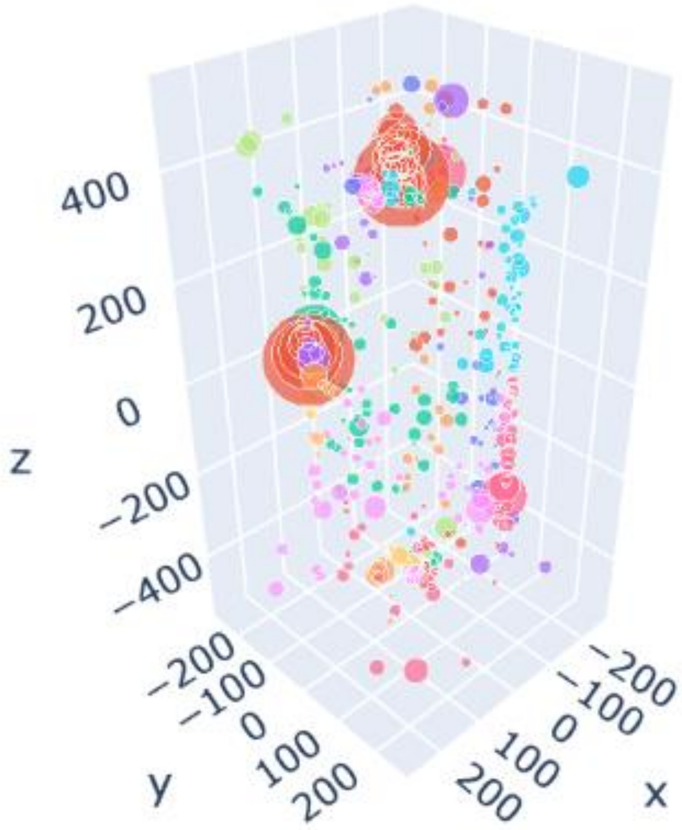
Calorimeter Clustering

Digital readouts are converted into ‘hits’ that look like this…

Multiple hits correspond to a single truth particle

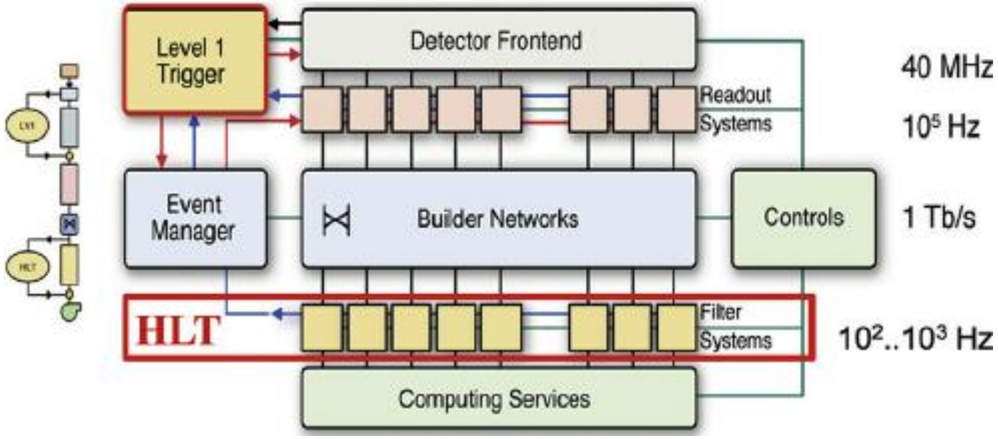
Goal: ‘reconstruct’ energy of original particles by clustering hits

Size of bubble = energy
Color = cluster



Triggering and Data Acquisition

- **~1 billion proton-proton interactions** occur per second inside the CMS detector.
- **Infeasible to store** that much data and much of it is **uninteresting**, anyway.
- The trigger system **identifies interesting events**, while enforcing a **maximum event rate**.
  - L1 trigger – ~100 kHz
  - **High Level Trigger (HLT) – ~1 kHz**



SPVCNN

- MOTIVATION
- IMPLEMENTATION

SPVCNN Motivation

Need models for **3D tasks** with:

- **Low latency**
- **High computational efficiency**
- **High accuracy**

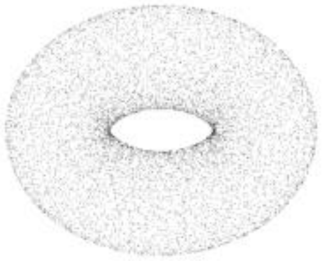
Original motivating problem was driverless cars.

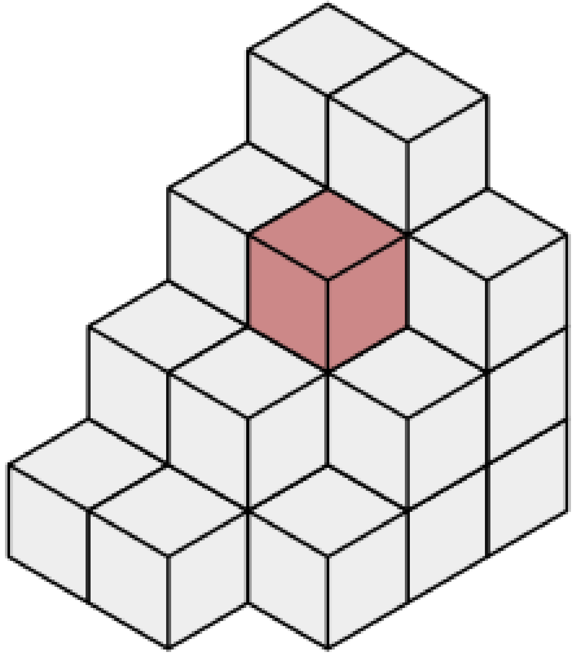
Reconstruction in particle physics shares many of the same requirements.

Previous Approaches

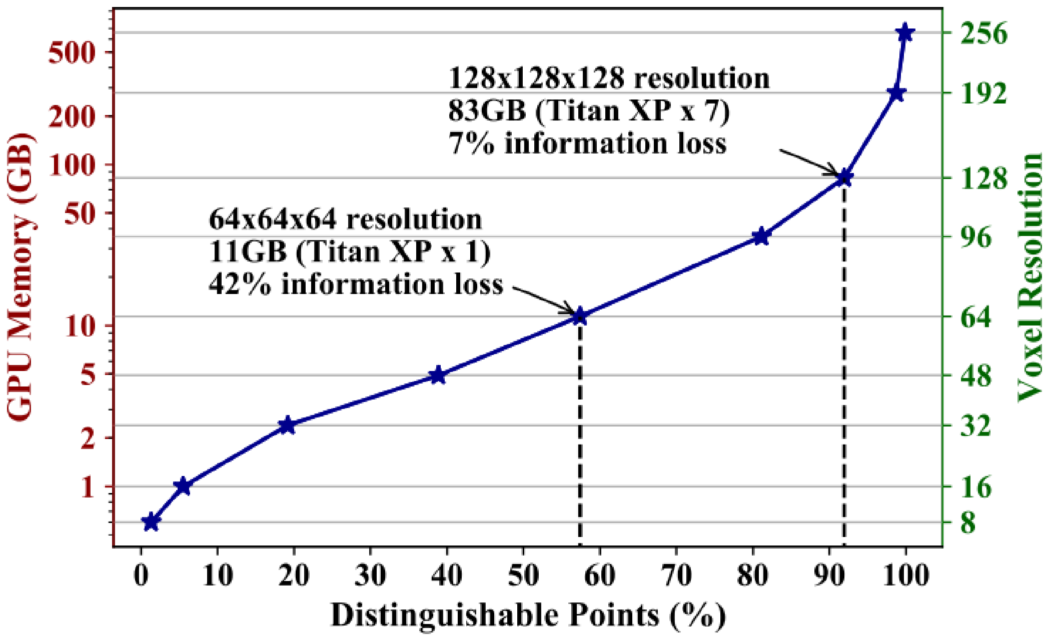
Fall into two categories:

- **Point cloud models**
- **Voxel models**

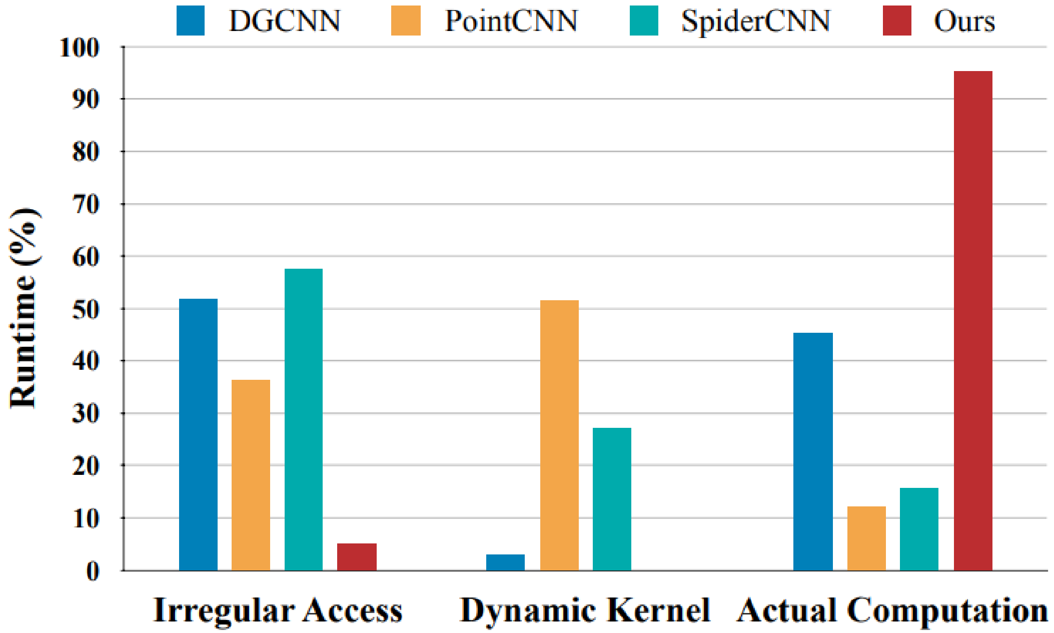




Limitations of Previous Approaches

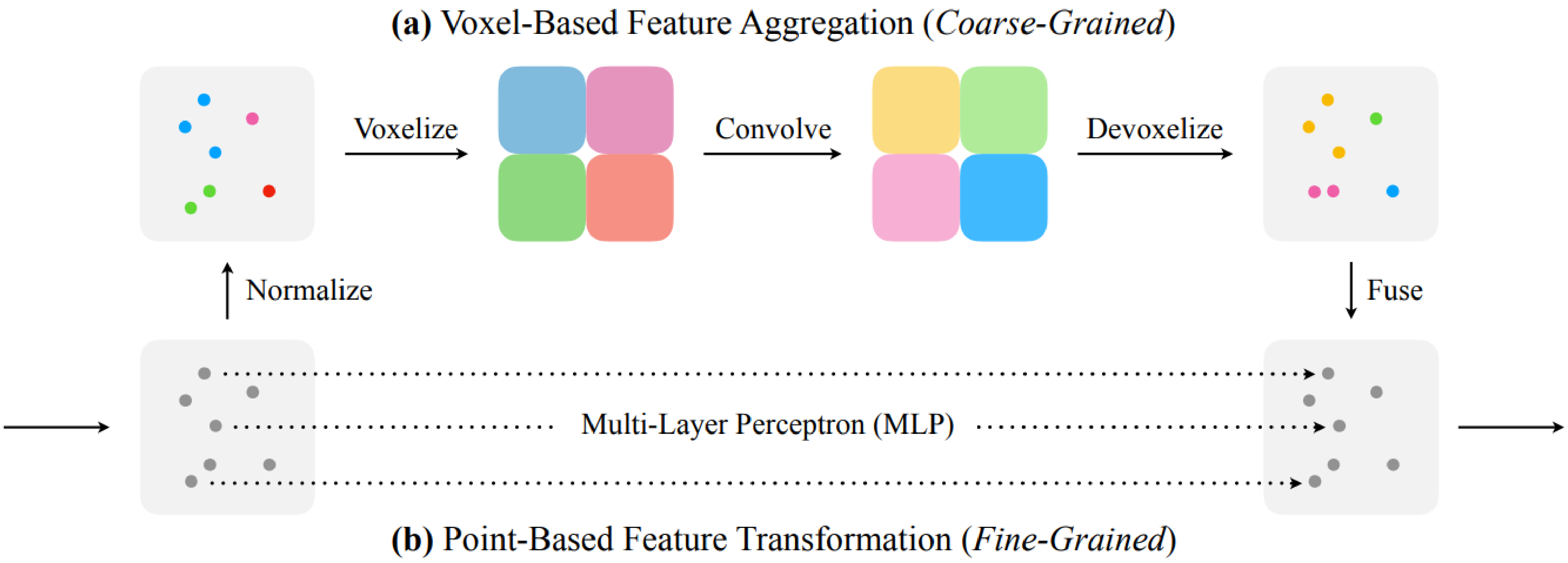


(a) Voxel-based: memory grows cubically

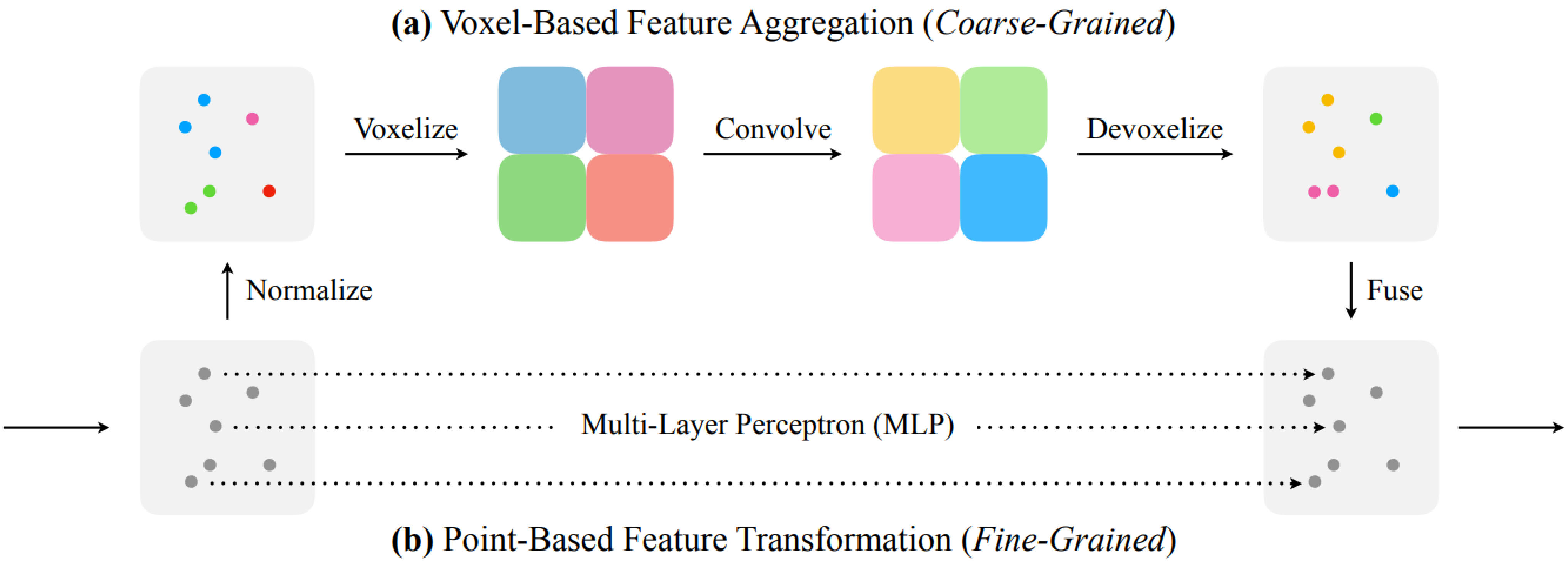


(b) Point-based: large memory/computation overheads

Point-Voxel Convolution (PVConv)

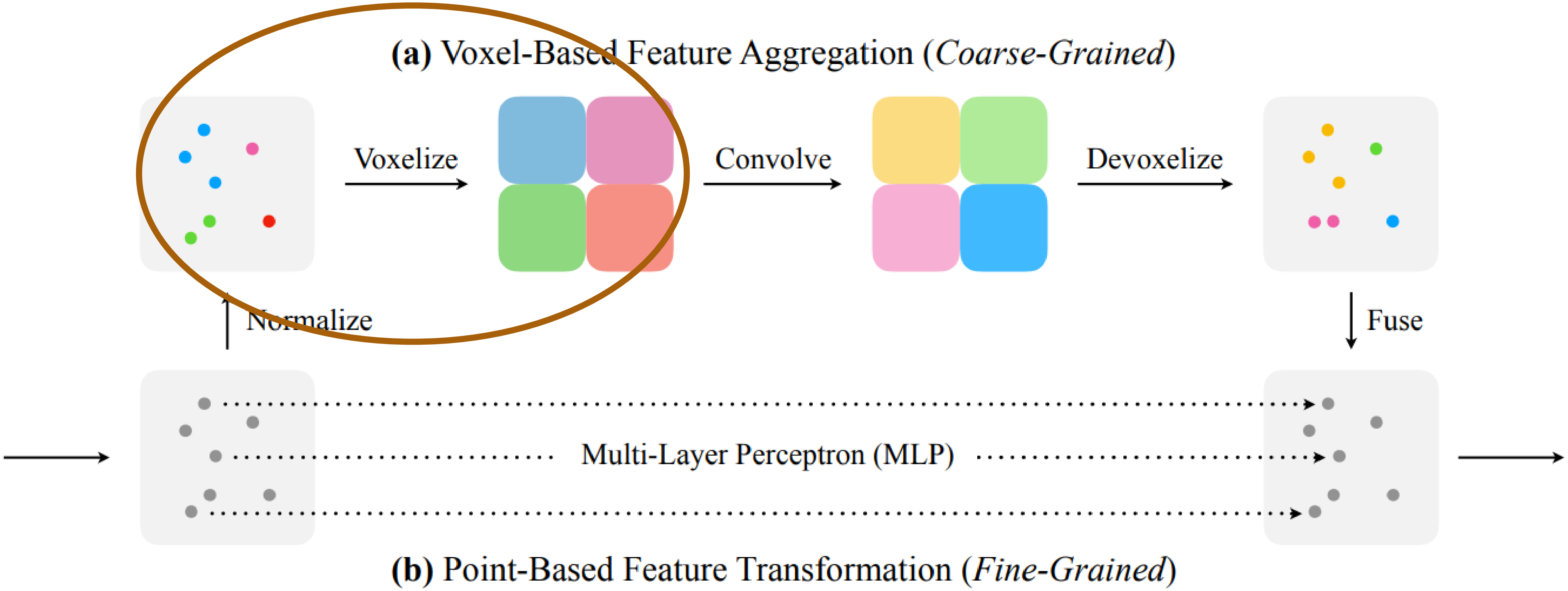

(a) Voxel-Based Feature Aggregation (Coarse-Grained)
Voxelize
Convolve
Devoxelize
Normalize
Fuse
Multi-Layer Perceptron (MLP)
(b) Point-Based Feature Transformation (Fine-Grained)

Sparse Point-Voxel Convolution (SPVConv)


(a) Voxel-Based Feature Aggregation (*Coarse-Grained*)
Voxelize
Convolve
Devoxelize
Normalize
Fuse
Multi-Layer Perceptron (MLP)
(b) Point-Based Feature Transformation (*Fine-Grained*)

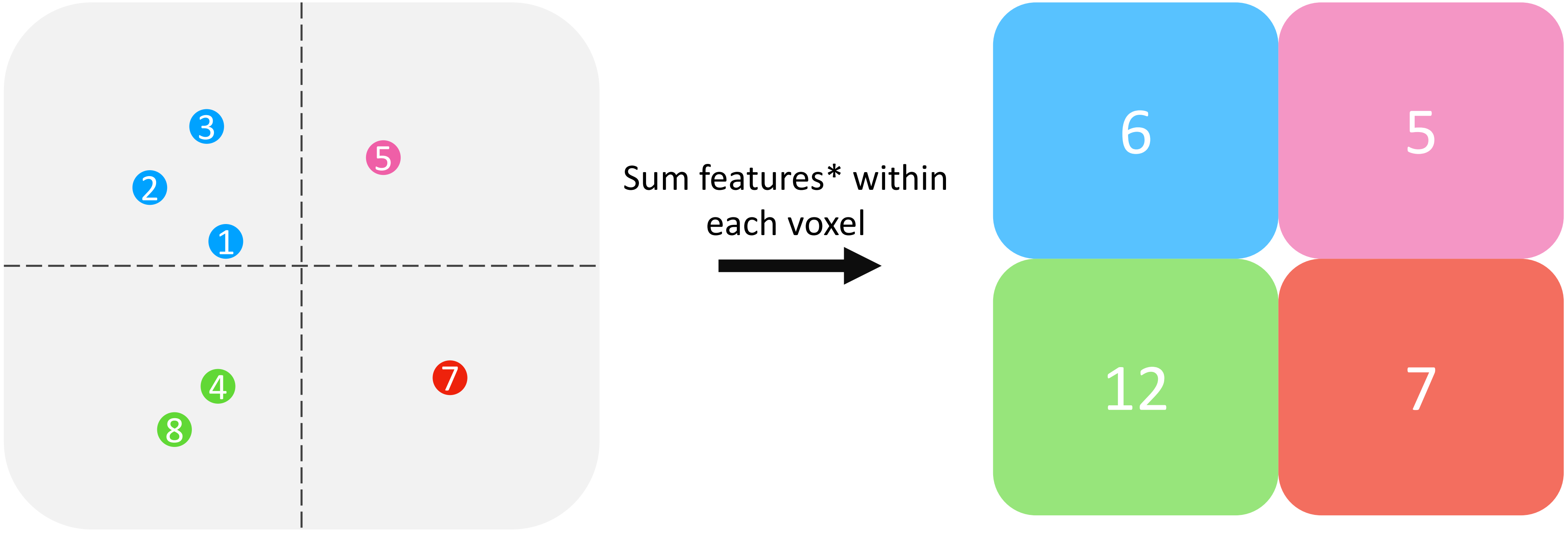
- Simply replaces upper branch with sparse convolution.
- Some details with normalization/voxelization and devoxelization/fusion:
  - Hashing, trilinear interpolation

Sparse Point-Voxel Convolution (SPVConv)


(a) Voxel-Based Feature Aggregation (Coarse-Grained)
Voxelize
Convolve
Devoxelize
Normalize
Fuse
Multi-Layer Perceptron (MLP)
(b) Point-Based Feature Transformation (Fine-Grained)

- Simply replaces upper branch with sparse convolution.
- Some details with normalization/voxelization and devoxelization/fusion:
  - Hashing, trilinear interpolation

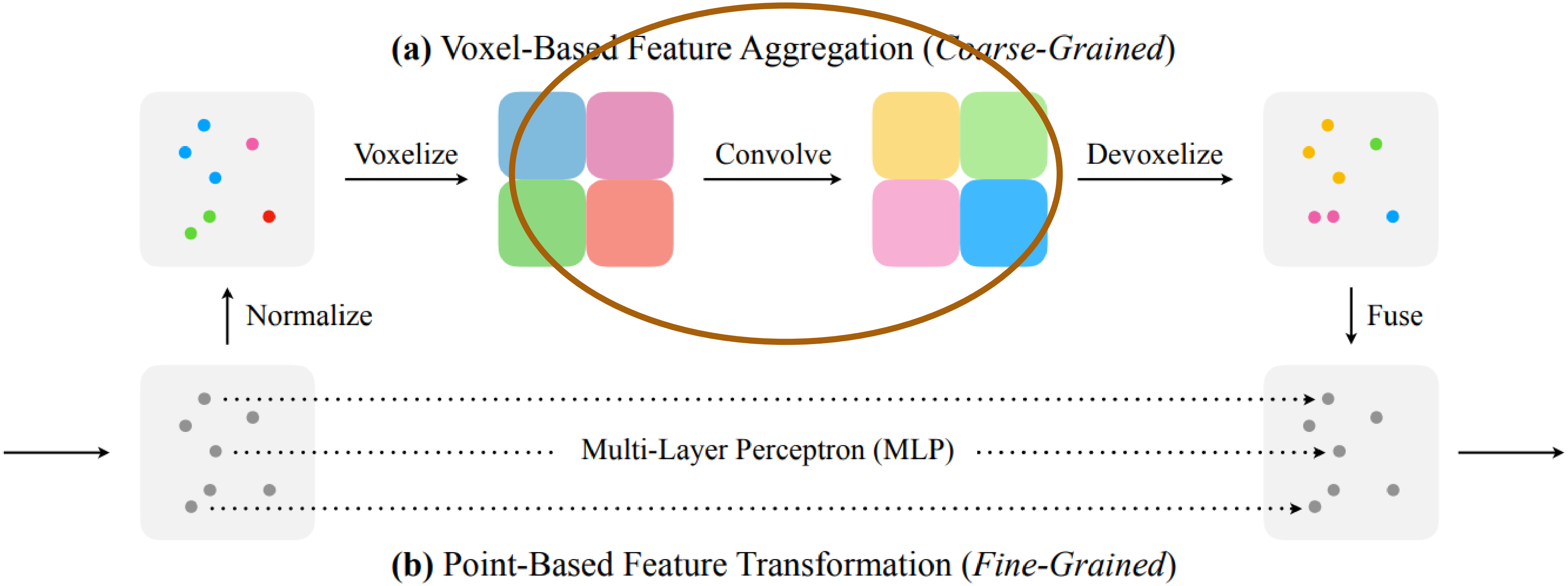
Voxelization



Sum features* within each voxel

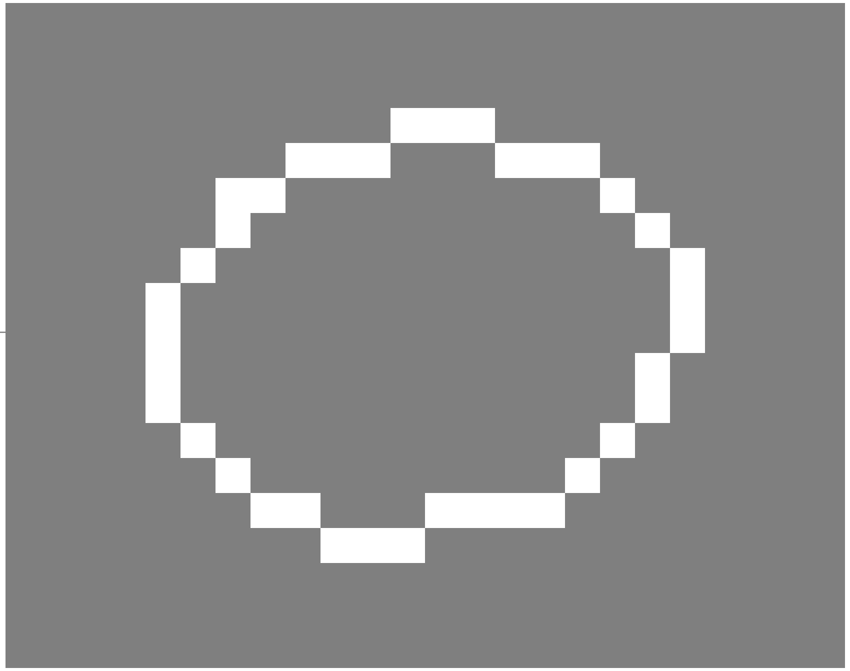
*for illustrative purposes, only a single feature dimension is shown in this example

Sparse Point-Voxel Convolution (SPVConv)


(a) Voxel-Based Feature Aggregation (*Coarse-Grained*)
Voxelize
Convolve
Devoxelize
Normalize
Fuse
Multi-Layer Perceptron (MLP)
(b) Point-Based Feature Transformation (*Fine-Grained*)

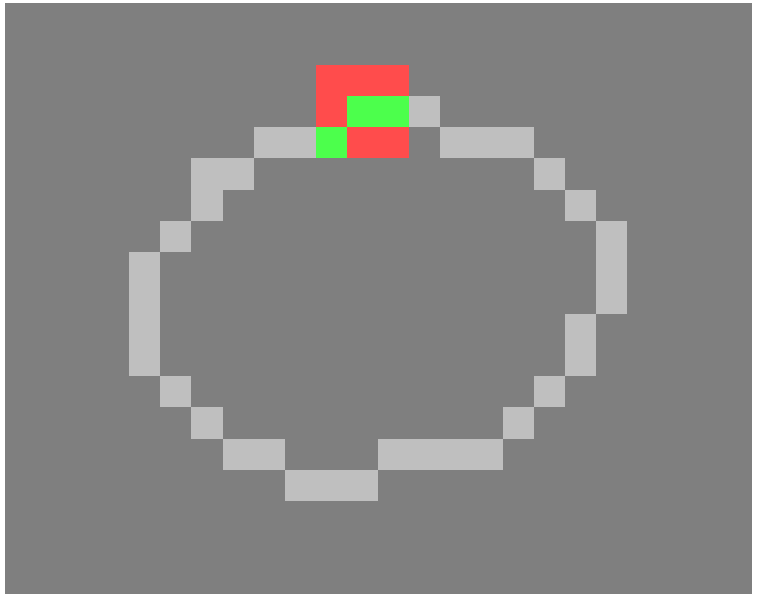
- Simply replaces upper branch with sparse convolution.
- Some details with normalization/voxelization and devoxelization/fusion:
  - Hashing, trilinear interpolation

Generalized Sparse Convolution

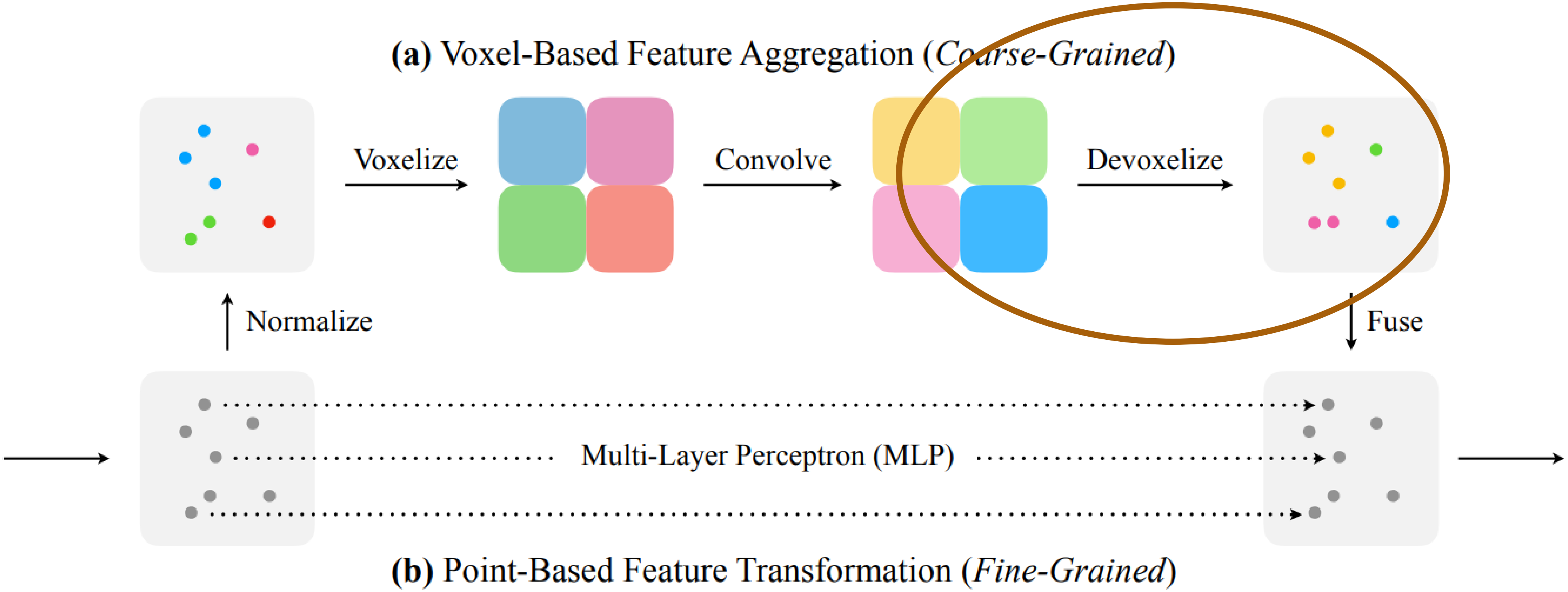


- Sparse convolutions operate **directly on sparse tensors**.
- **Avoids wasted computation** and allows for **higher resolution**.

- Naïve implementations (top) would quickly reduce sparsity.
- Modern implementations (bottom) allow for arbitrary input (c_{in}) and output (c_{out}) coordinates. The example shown is a 'submanifold sparse convolution', which sets $c_{in} = c_{out}$, thus preserving sparsity. This is (almost) used in SPVCNN.



Devoxelization

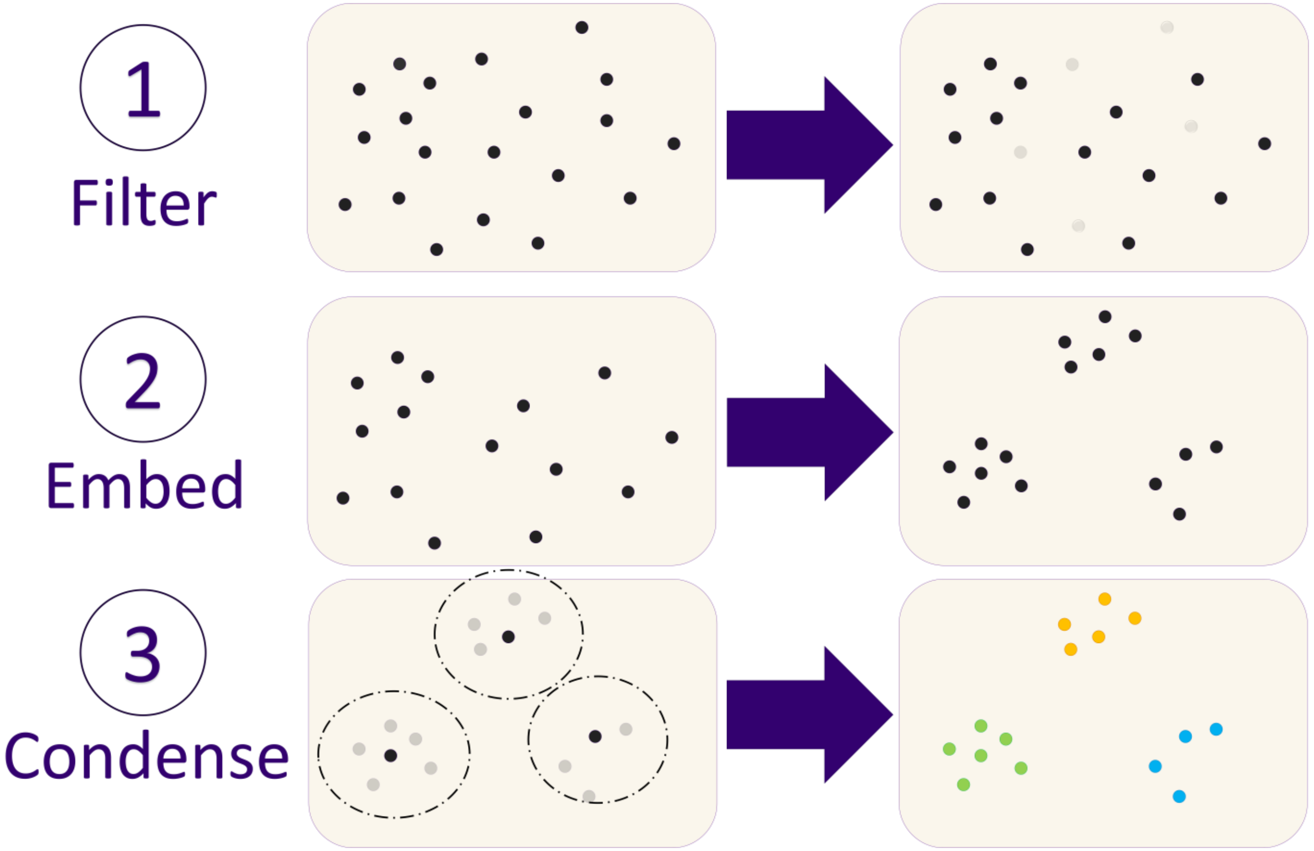

(a) Voxel-Based Feature Aggregation (*Coarse-Grained*)
Voxelize
Convolve
Devoxelize
Normalize
Fuse
Multi-Layer Perceptron (MLP)
(b) Point-Based Feature Transformation (*Fine-Grained*)

- Simply replaces upper branch with sparse convolution.
- Some details with normalization/voxelization and devoxelization/fusion:
  - Hashing, trilinear interpolation

Object Condensation

1. **Filter –** predict semantic labels, discard noise
2. **Embed –** map to embedded space + predict ‘condensation’ score
3. **Condense –** bounded nearest-neighbor search in embedded space around points with high condensation score

See object condensation paper: 2002.03605.pdf (arxiv.org)



1 Filter

2 Embed

3 Condense

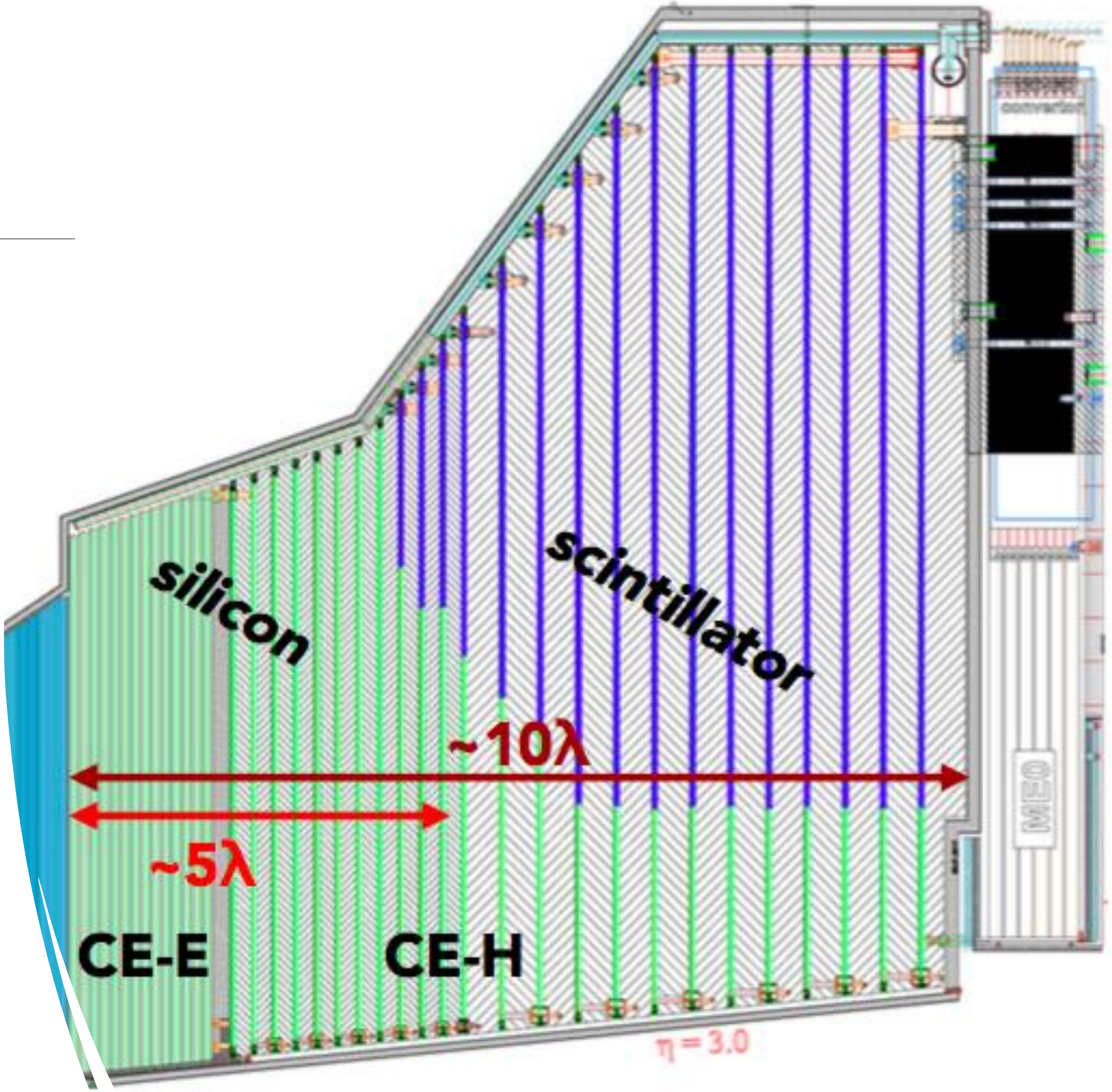
HGCAL

CMS High-Granularity Calorimeter (HGCAL)

Major upgrade for HL-LHC: **6.5M channels**, **50 layers.**

Finer granularity, timing resolution → greater benefit from 3D deep learning.

Despite increased data volume, cannot sacrifice latency.



HGCAL Samples

Zero pileup, double-tau dataset.

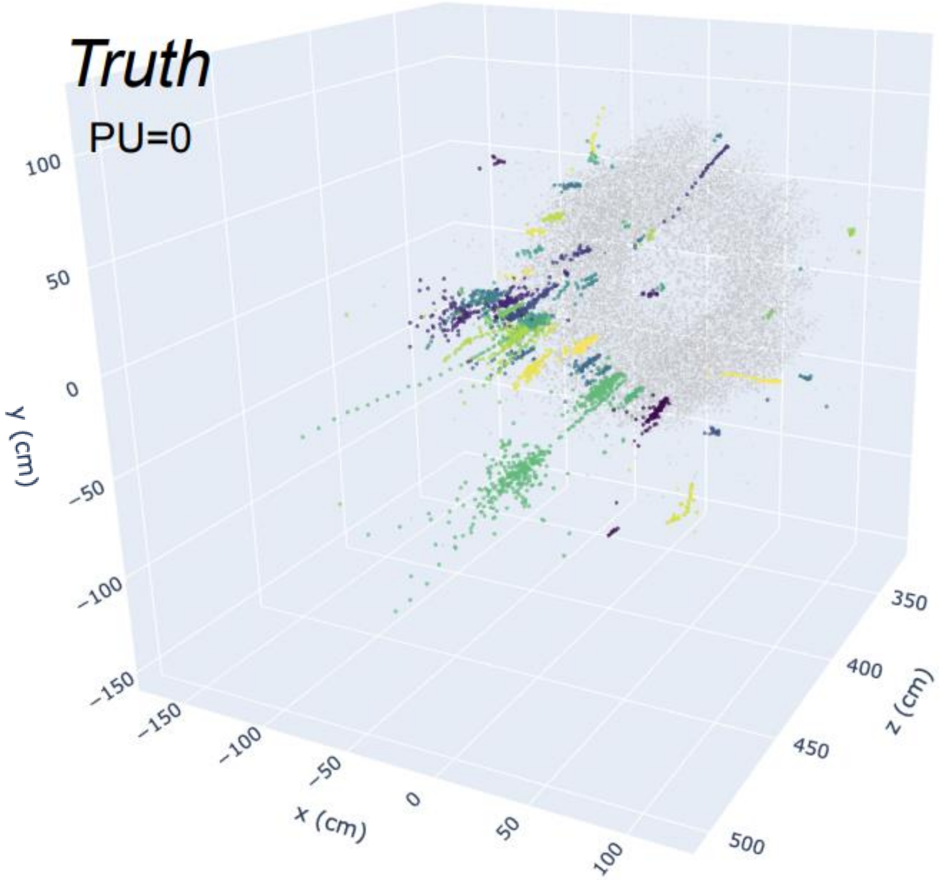
CMS detector simulation with GEANT4.

Simulation-level energy deposits are mapped onto reconstructed energy deposits to form the truth definition.

Inseparable showers (due to overlap) are merged.

Each event has ~20K hits.

See [CR2022_033.pdf (cern.ch)]() for detailed description of samples.

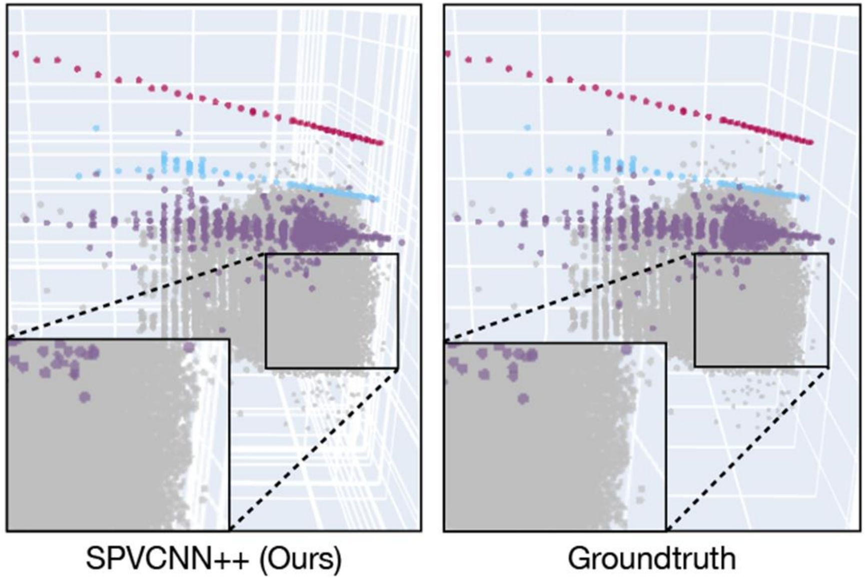


HGCAL Results

First, a **disclaimer**:

- For some of these results**, we compare against a GNN-based method**.
- There have been **further improvements** in GNN-based approaches (as discussed in other talks at this workshop) since this comparison, and there are **pros and cons** of each.
- The primary purpose is to show that **both methods are competitive**.

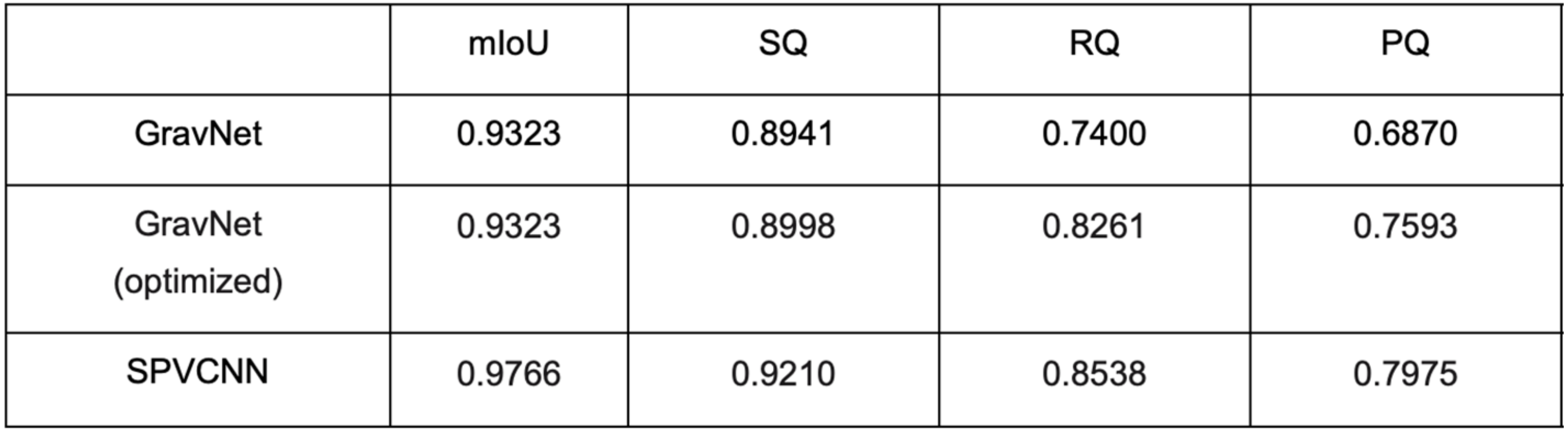
HGCAL Results



Left – predicted clusters from SPVCNN.
Right – event display from HGCAL.

Each point represents an energy deposit in the calorimeter. Each color corresponds to a cluster.

HGCAL Results



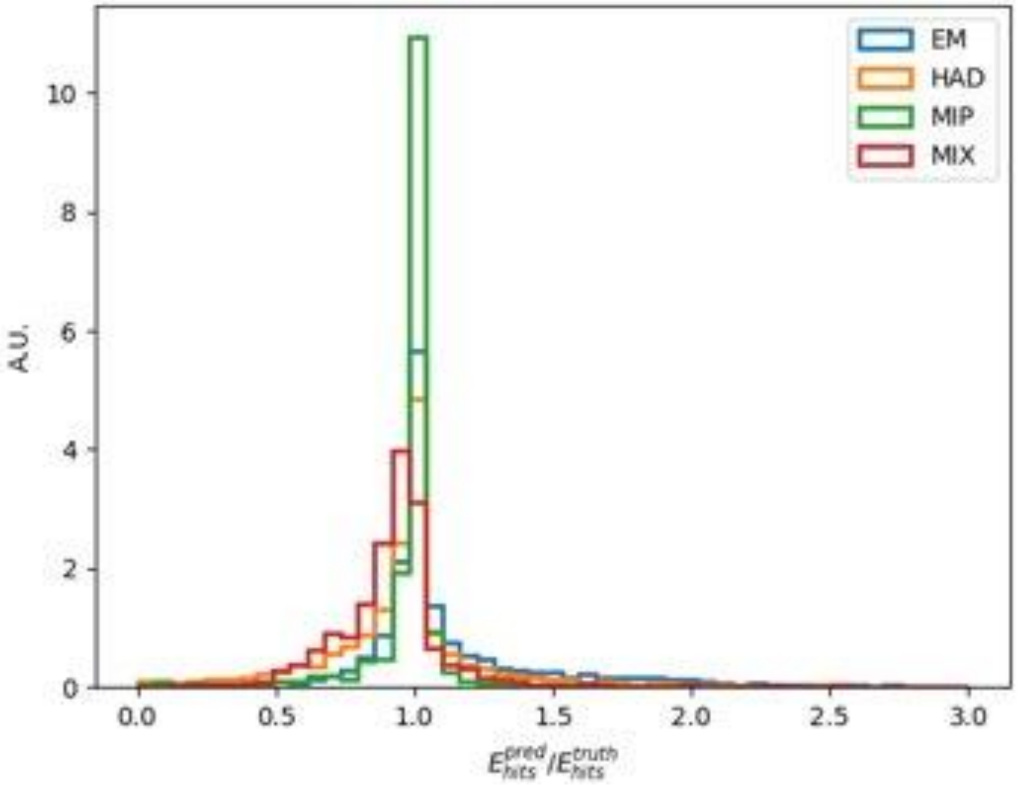
	mIoU	SQ	RQ	PQ
GravNet	0.9323	0.8941	0.7400	0.6870
GravNet (optimized)	0.9323	0.8998	0.8261	0.7593
SPVCNN	0.9766	0.9210	0.8538	0.7975

IoU – measure of overlap between predicted and true classes (signal and noise).
SQ – average overlap between predicted and true clusters for each semantic class.
RQ – fraction of clusters for each semantic class that were matched.
PQ – product of SQ and RQ.

HGCAL

Right: **ratio of predicted to true energy** for each predicted cluster, split into four types:

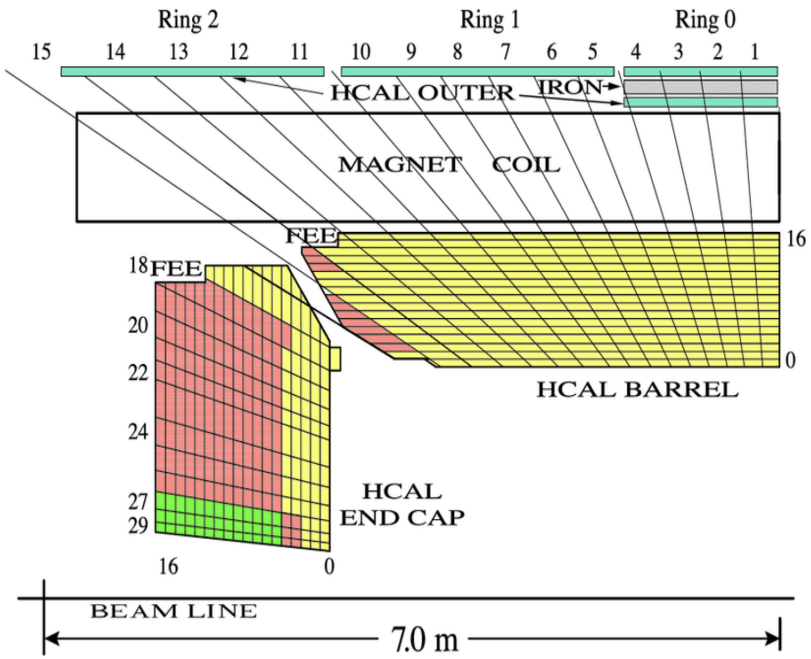
- Electromagnetic (EM) particles
- Hadronic (HAD) particles
- Minimum-ionizing particles (MIP)
- A mixture of the above (MIX)

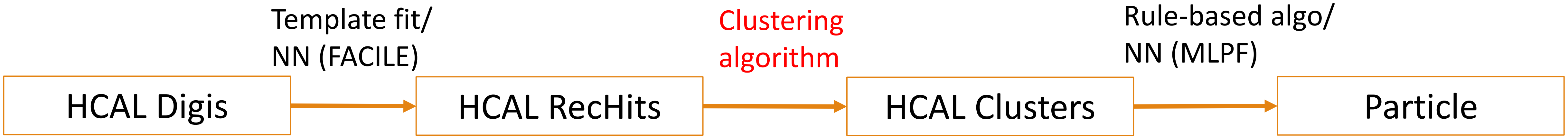


HCAL

CMS Hadronic Calorimeter (HCAL)

- ~16k channels, ~18 layers
- More challenging for ML-based method, due to reduced information




HCAL Digis

Template fit/ NN (FACILE)

HCAL RecHits

Clustering algorithm

HCAL Clusters

Rule-based algo/ NN (MLPF)

Particle

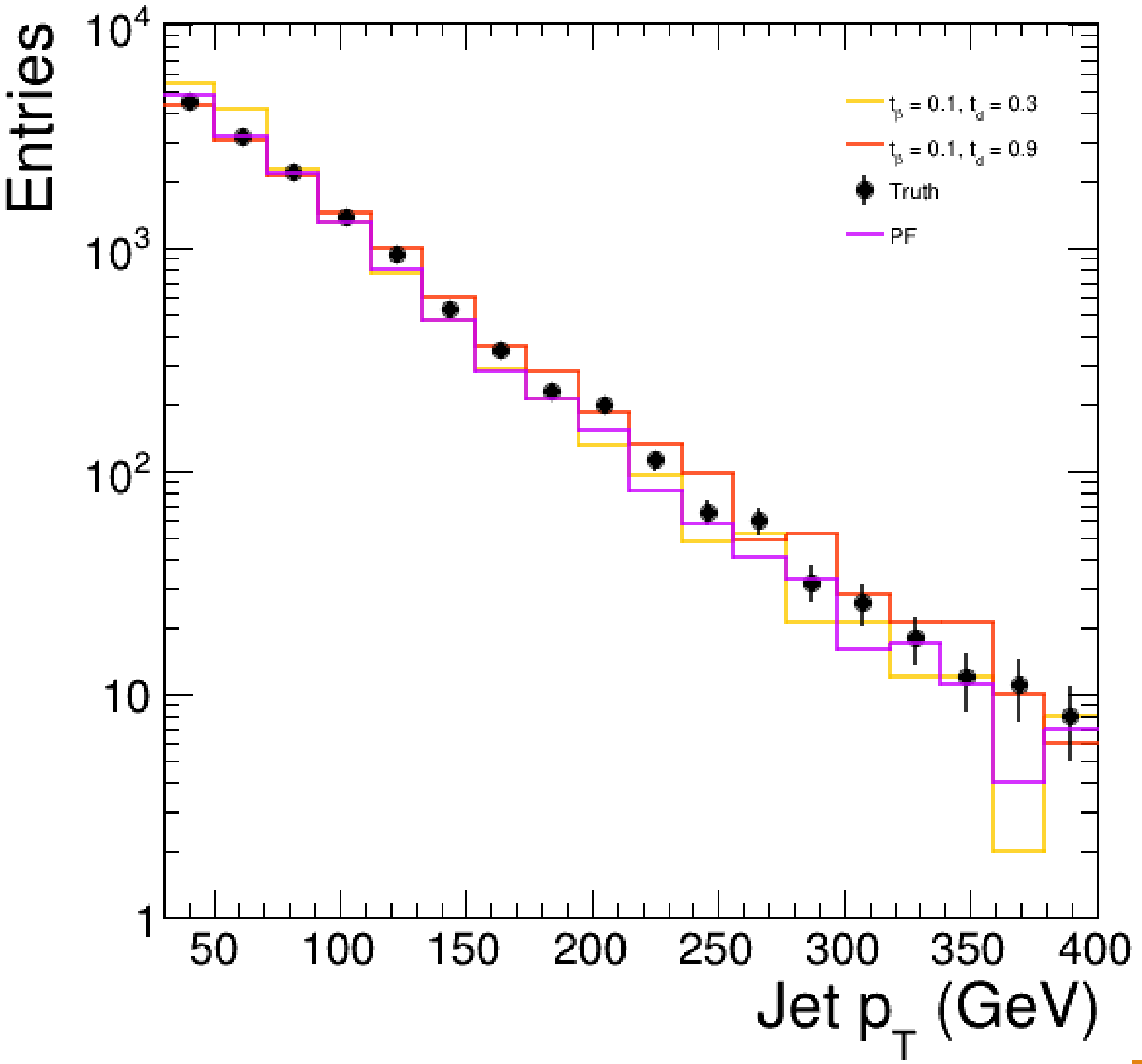
Ultimate goal is to build clusters from low-level charge collected in 25 ns windows in a single step

HCAL Samples

- Zero pileup, ttbar dataset.
- CMS detector simulation with GEANT4.
- Simulation-level energy deposits are mapped onto reconstructed energy deposits to form the truth definition.
- The details of truth matching are a bit different than for HGCAL – see backup slides for more details.

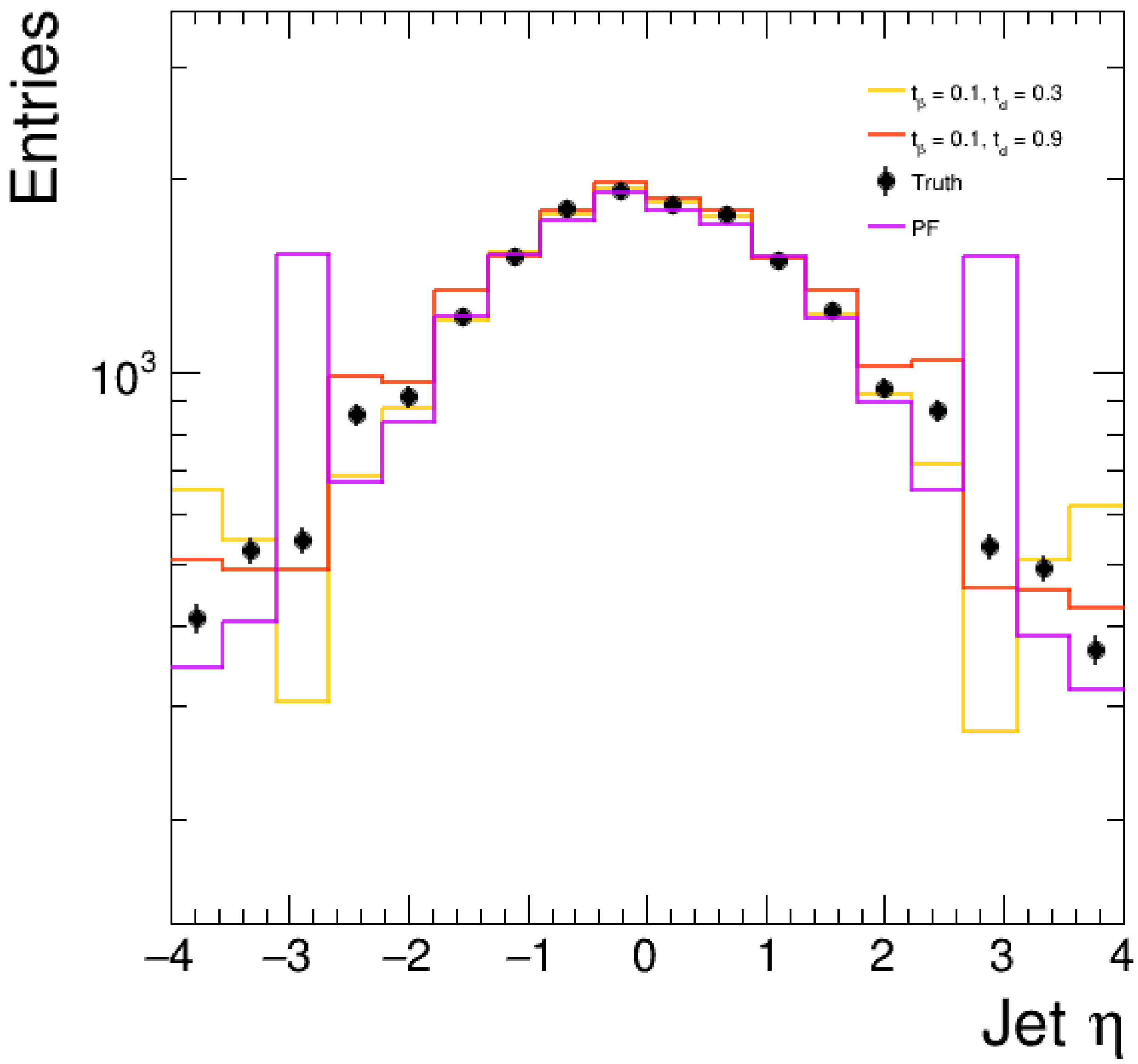
HCAL Results

- Right: jet p_T reconstructed w/ CMS toolchain, using various HCAL clusters as input:
    - Yellow and red lines use SPVCNN clusters with **different clustering hyperparameters**.
    - Purple lines use the existing **particle flow** (PF) method.
    - Finally, the **black points** use the **true clusters**.
- Similar performance between the methods.
- Could be useful to have dynamic clustering parameters.



HCAL Results

- Right: jet η reconstructed w/ CMS toolchain, using various HCAL clusters as input:
  - Yellow and red lines use SPVCNN clusters with **different clustering hyperparameters**.
  - Purple lines use the existing **particle flow** (PF) method.
  - Finally, the **black points** use the **true clusters**.
- Note the spurious peak at $|\eta| \approx 3$ for PF.



Main Takeaways

Modern convolutional approaches that exploit tricks for **efficient computation** are **competitive** with **current clustering methods** and other proposed **ML methods** at **calorimeters**.

Latency at the level needed for the HLT **(~ms)** is **currently achievable with GPU accelerators**. Beyond this level, further innovations are probably required, e.g., exploiting FPGAs and ASICs.

For now, only looked at CMS detectors – could expand in the future (e.g., to ATLAS).

Links & References

PVCNN – [1907.03739] Point-Voxel CNN for Efficient 3D Deep Learning (arxiv.org)

SPVNAS – [2007.16100] Searching Efficient 3D Architectures with Sparse Point-Voxel Convolution (arxiv.org)

CMS GNN – https://cds.cern.ch/record/2803236/files/CR2022_033.pdf

Object condensation – 2002.03605.pdf (arxiv.org)

Backup

HCAL Truth Definition

Needed objects in CMSSW

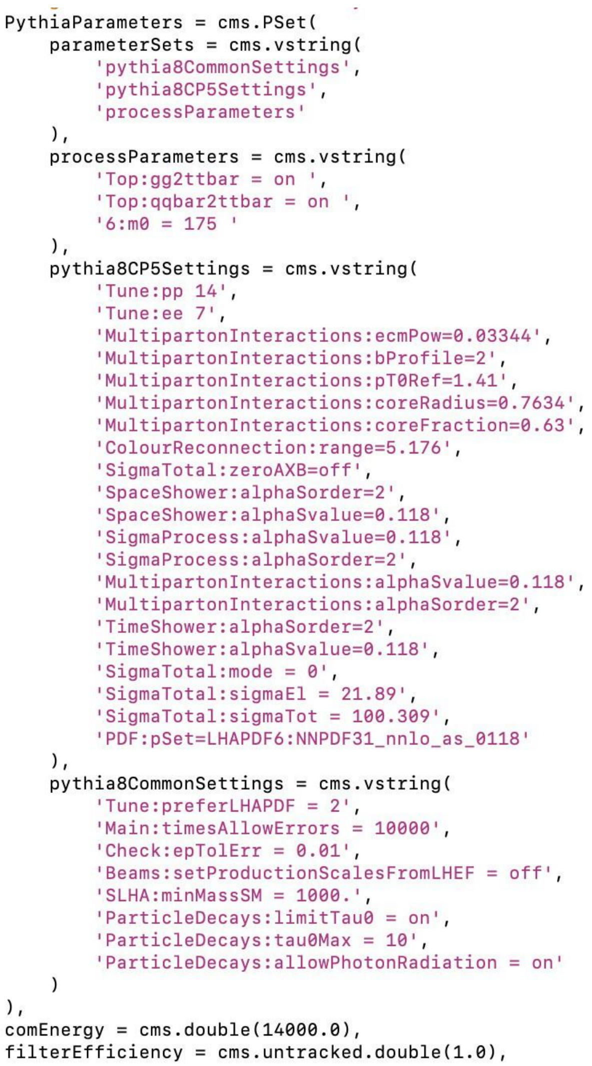
- Collection of **Simulated Hits** (simHits). A simHit has the following info:
  - An associated **detector ID** (takes a bit of massaging to get this info in the right format)
  - A **GEANT Track ID**. Every time a particle branches or interacts, its daughter particles are assigned unique IDs, and the parent information is preserved via its association with a *GEANT vertex*. The simHit *does not* have info about the parent though, just its own associated track ID
  - simHit energy, depth, and time information
- Collection of "**simulated tracks**" (simTracks). These are for the GEANT tracks discussed above, which are basically simulated particles. A simTrack has:
  - A **GEANT track ID**
  - A **GEANT *vertex* ID**, which is where the simulated particle was created
- Collection of "**simulated vertices**" (simVertices). These are the points at which GEANT decays happen. simVertices have:
  - An associated **parent particle ID** (which is the GEANT track ID of the particle that was in the initial state of the vertex)
  - *Vertex position information* (This is not currently used in my code, but it could be)
- Collection of **reconstructed hits** (recHits). This is *reconstructed* information, i.e. an emulation of the information that you would have for a data event. recHits have:
  - Associated **detector ID** information
  - Energy, time, position, etc. information. When I get the truth information for training purposes, the recHits have already gone through default PF HCAL clustering, so each recHit also has an associated PF cluster

Truth-based clustering concept

- The truth-based clustering concept is relatively straightforward, but possibly too naïve:
  - For a given recHit, find the simHit that **contributed the most energy** to that recHit
  - Trace back along the GEANT daughter->parent relationships until we get to parents with ID = -1. These are the “stable” particles that resulted from simulated process
- Notes:
  - We are considering only tracing back until we get to the GEANT vertex closest to the inner edge of the HCAL. Some discussions with Jan have led us to think that this might be a better truth-clustering definition
  - The scheme as described above was motivated by the desire to associate a stable particle (which would give a reco track for charged particles) with a single cluster containing all daughter particles (and thus as much of the stable particle’s energy as possible)

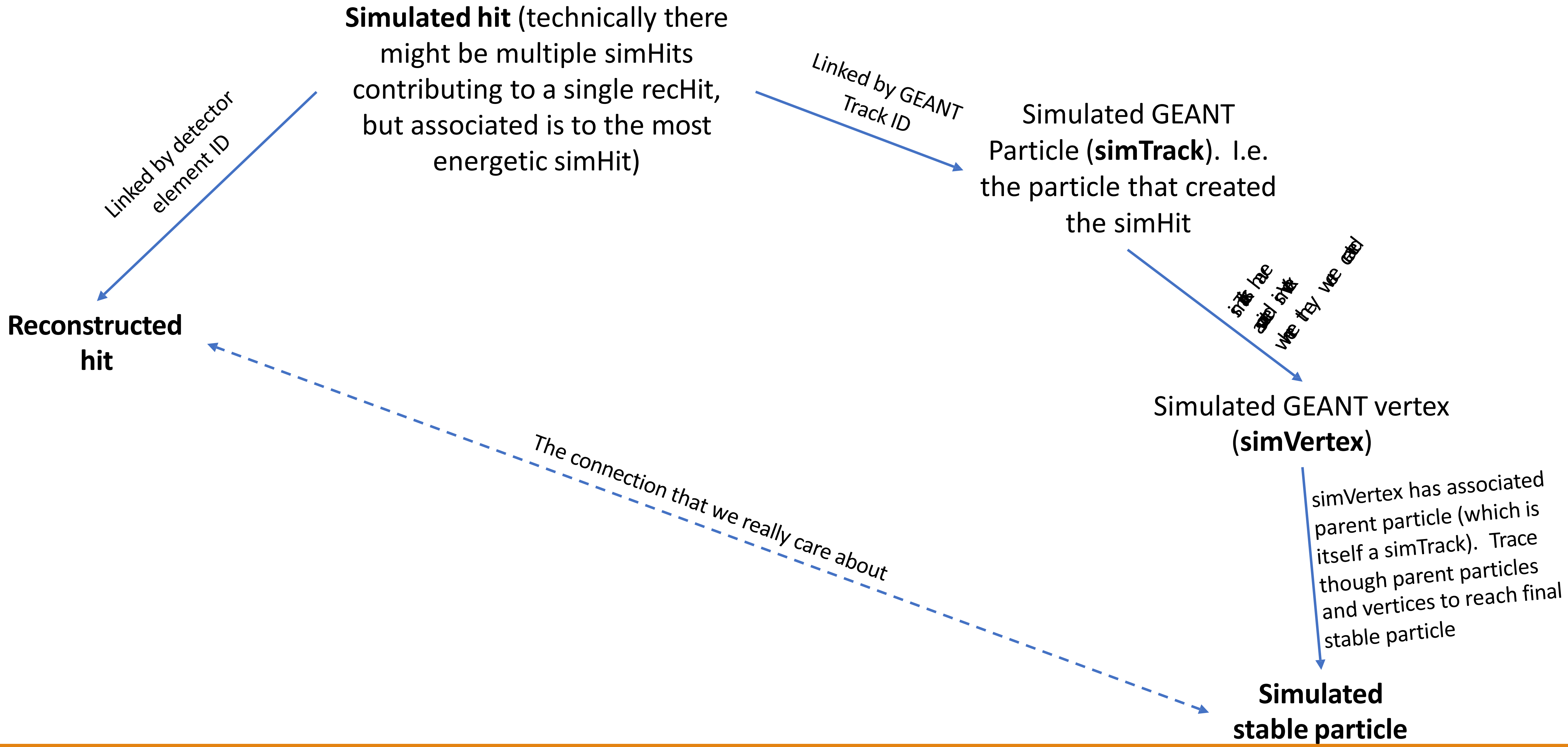
Simulation

- The following studies were performed using pythia 8 ttbar samples with the settings shown on the right
- Showering through the CMS simulation is performed with GEANT



```
PythiaParameters = cms.PSet(
    parameterSets = cms.vstring(
        'pythia8CommonSettings',
        'pythia8CP5Settings',
        'processParameters'
    ),
    processParameters = cms.vstring(
        'Top:gg2ttbar = on ',
        'Top:qqbar2ttbar = on ',
        '6:m0 = 175 '
    ),
    pythia8CP5Settings = cms.vstring(
        'Tune:pp 14',
        'Tune:ee 7',
        'MultipartonInteractions:ecmPow=0.03344',
        'MultipartonInteractions:bProfile=2',
        'MultipartonInteractions:pT0Ref=1.41',
        'MultipartonInteractions:coreRadius=0.7634',
        'MultipartonInteractions:coreFraction=0.63',
        'ColourReconnection:range=5.176',
        'SigmaTotal:zeroAXB=off',
        'SpaceShower:alphaSorder=2',
        'SpaceShower:alphaSvalue=0.118',
        'SigmaProcess:alphaSvalue=0.118',
        'SigmaProcess:alphaSorder=2',
        'MultipartonInteractions:alphaSvalue=0.118',
        'MultipartonInteractions:alphaSorder=2',
        'TimeShower:alphaSorder=2',
        'TimeShower:alphaSvalue=0.118',
        'SigmaTotal:mode = 0',
        'SigmaTotal:sigmaEl = 21.89',
        'SigmaTotal:sigmaTot = 100.309',
        'PDF:pSet=LHAPDF6:NNPDF31_nnlo_as_0118'
    ),
    pythia8CommonSettings = cms.vstring(
        'Tune:preferLHAPDF = 2',
        'Main:timesAllowErrors = 10000',
        'Check:epTolErr = 0.01',
        'Beams:setProductionScalesFromLHEF = off',
        'SLHA:minMassSM = 1000.',
        'ParticleDecays:limitTau0 = on',
        'ParticleDecays:tau0Max = 10',
        'ParticleDecays:allowPhotonRadiation = on'
    )
),
comEnergy = cms.double(14000.0),
filterEfficiency = cms.untracked.double(1.0),
```

Truth-based clustering in practice



Simulated hit (technically there might be multiple simHits contributing to a single recHit, but associated is to the most energetic simHit)

Linked by GEANT Track ID

Simulated GEANT Particle (**simTrack**). I.e. the particle that created the simHit

[illegible] they [illegible]

Simulated GEANT vertex (**simVertex**)

simVertex has associated parent particle (which is itself a simTrack). Trace though parent particles and vertices to reach final stable particle

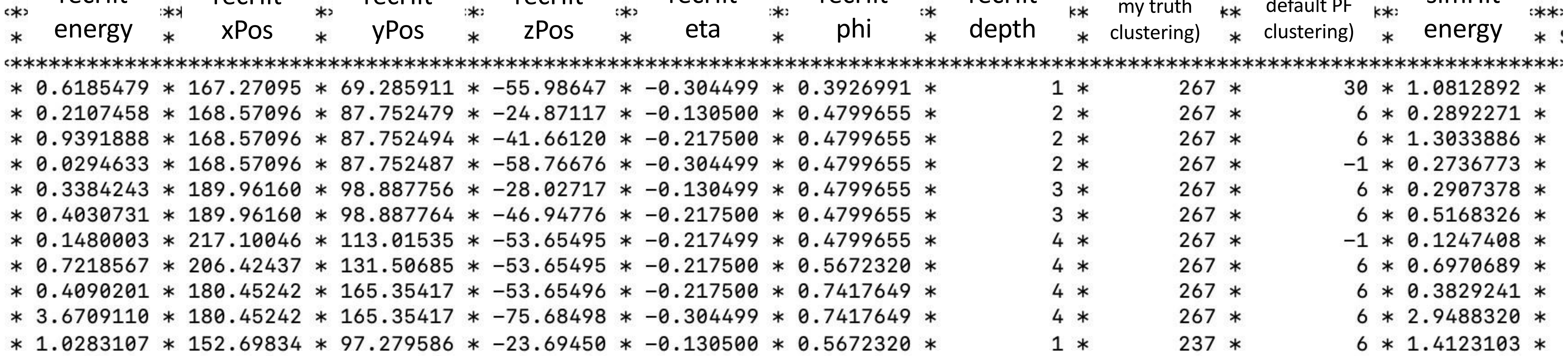
Simulated stable particle

The connection that we really care about

Reconstructed hit

Linked by detector element ID

Truth-based clustering vs PF clustering



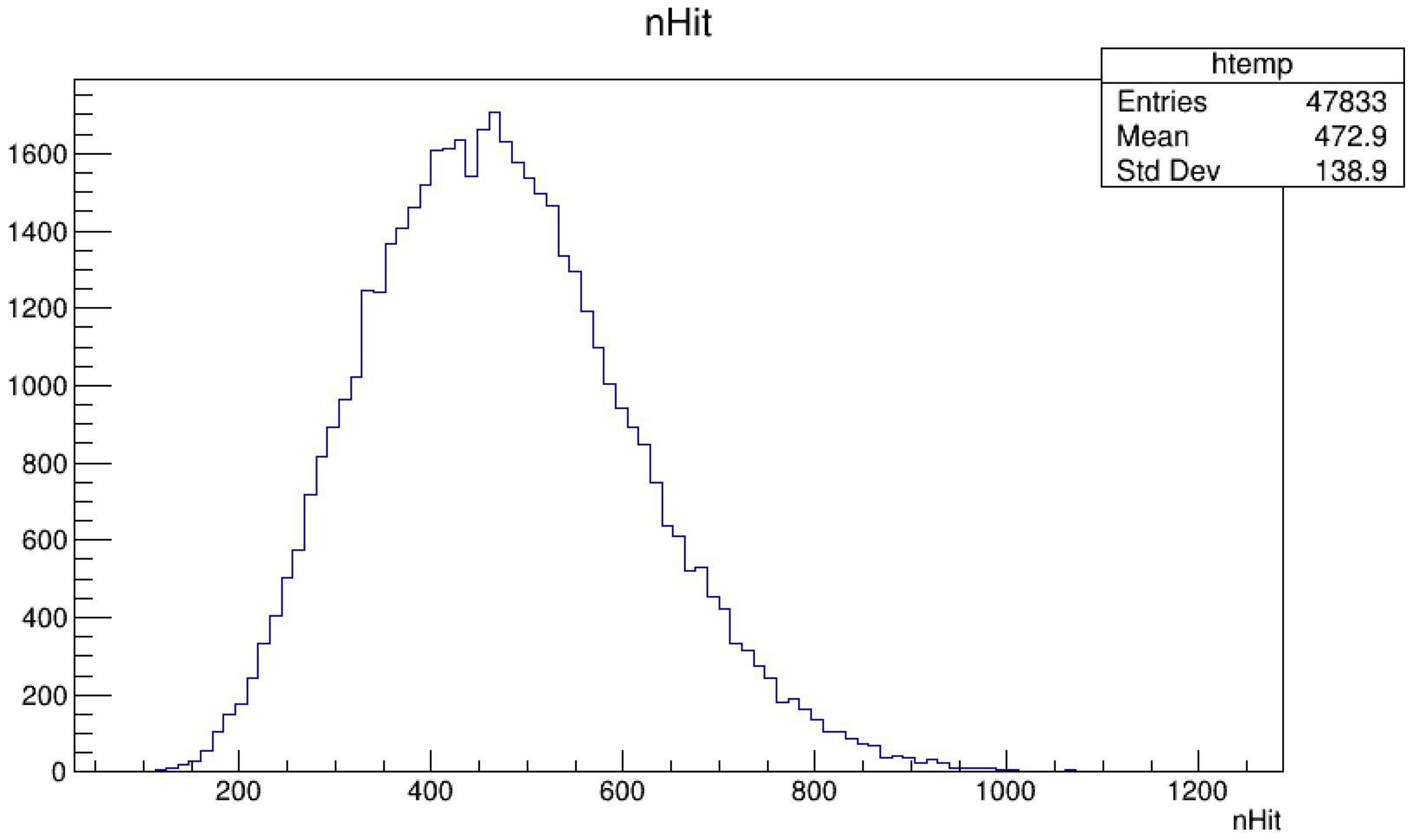
recHit energy	recHit xPos	recHit yPos	recHit zPos	recHit eta	recHit phi	recHit depth	recHit parent (based on my truth clustering)	recHit cluster (based on default PF clustering)	simHit energy
0.6185479	167.27095	69.285911	-55.98647	-0.304499	0.3926991	1	267	30	1.0812892
0.2107458	168.57096	87.752479	-24.87117	-0.130500	0.4799655	2	267	6	0.2892271
0.9391888	168.57096	87.752494	-41.66120	-0.217500	0.4799655	2	267	6	1.3033886
0.0294633	168.57096	87.752487	-58.76676	-0.304499	0.4799655	2	267	-1	0.2736773
0.3384243	189.96160	98.887756	-28.02717	-0.130499	0.4799655	3	267	6	0.2907378
0.4030731	189.96160	98.887764	-46.94776	-0.217500	0.4799655	3	267	6	0.5168326
0.1480003	217.10046	113.01535	-53.65495	-0.217499	0.4799655	4	267	-1	0.1247408
0.7218567	206.42437	131.50685	-53.65495	-0.217500	0.5672320	4	267	6	0.6970689
0.4090201	180.45242	165.35417	-53.65496	-0.217500	0.7417649	4	267	6	0.3829241
3.6709110	180.45242	165.35417	-75.68498	-0.304499	0.7417649	4	267	6	2.9488320
1.0283107	152.69834	97.279586	-23.69450	-0.130500	0.5672320	1	237	6	1.4123103

- This parent particle (ID 267) contributed the leading energy deposition to 10 recHit. In my truth-based clustering, these would all be clustered together. In default PF clustering, 7 of the recHits are put into a cluster with ID 6, one is put into a cluster with ID 30, and two are unassigned to a cluster (ID -1)
  - You can inspect the hits' x, y, z, eta, phi, and depth information
- Conversely, one additional recHit, which is linked to the parent particle with ID 237, is grouped into cluster 6 based on default PF clustering
  - PF Cluster 30 has only the 1 hit

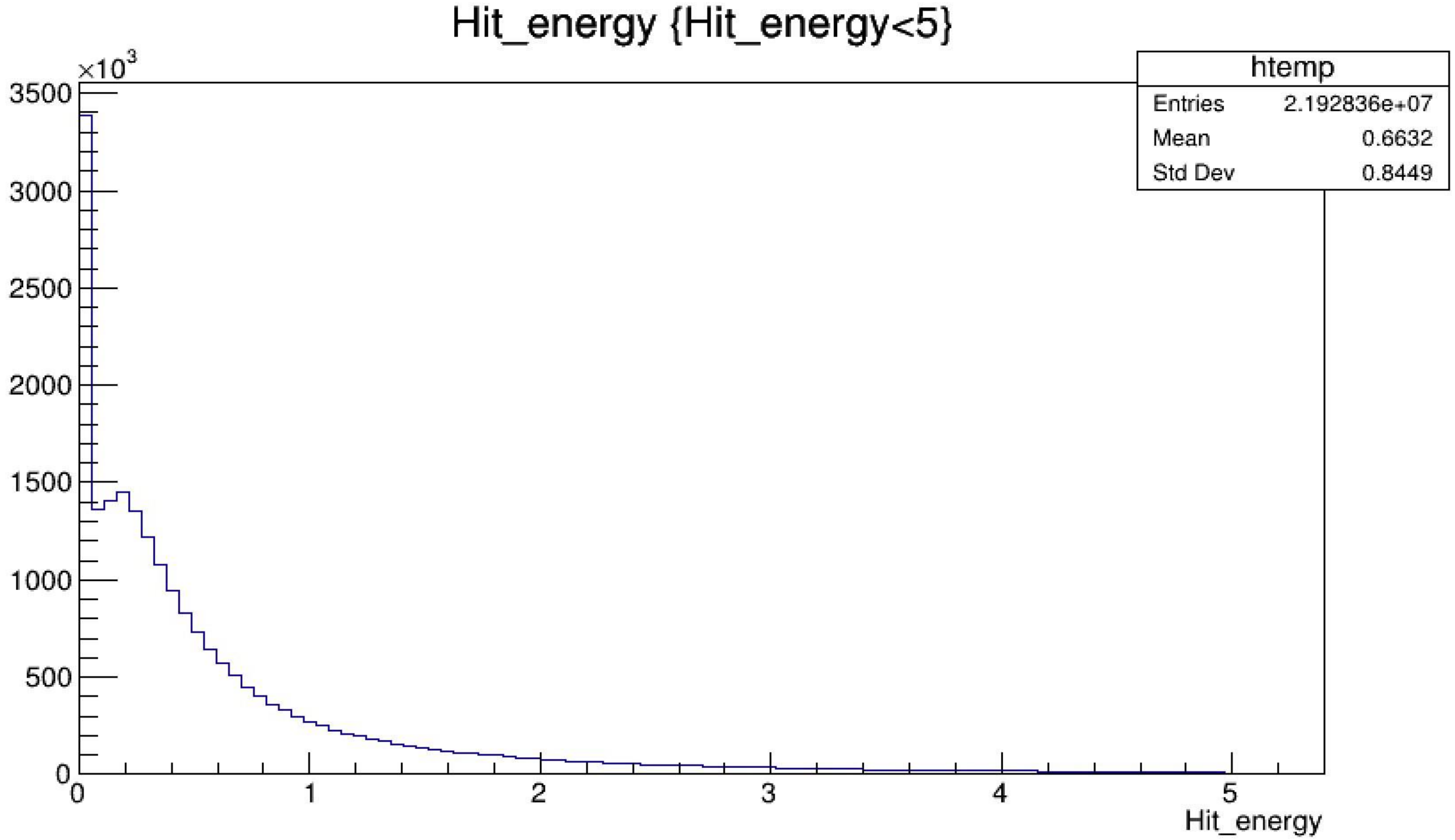
Another comparison

- Next slide has a table from a different event
  - You can see that there are 16 recHits that I've assigned to parent particle 267
  - These end up getting grouped into clusters 92, 13, 12, 8, and 14 based on default PF clustering (and 4 are unassigned)
  - I've included the other recHits that contribute to clusters 92, 13, 12, 8, and 14
- I'm showing this event since it's got about twice as many hits as the previous example and has a bit more of a non-trivial truth-based to pf-clustering mapping

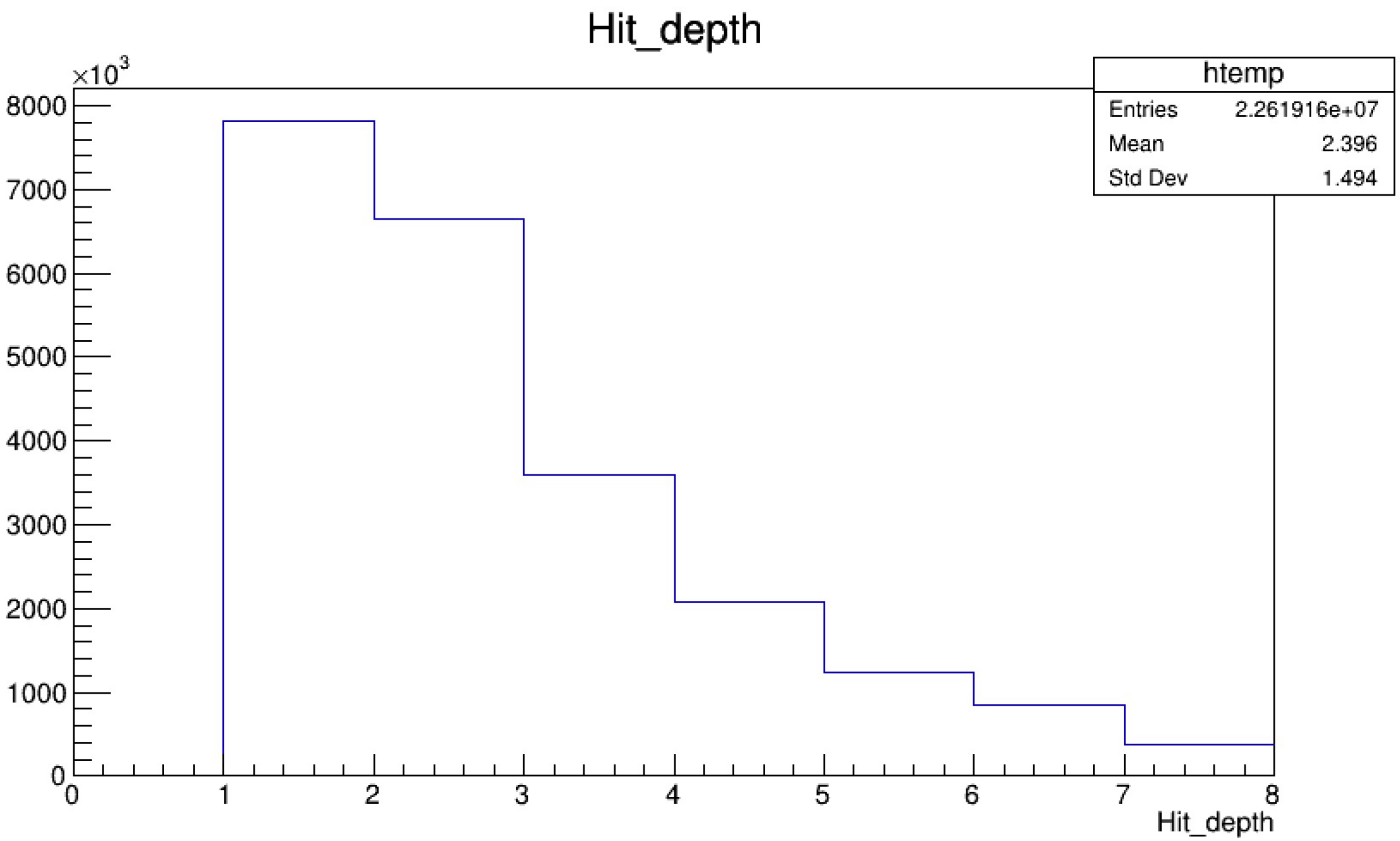
Number of recHits per event



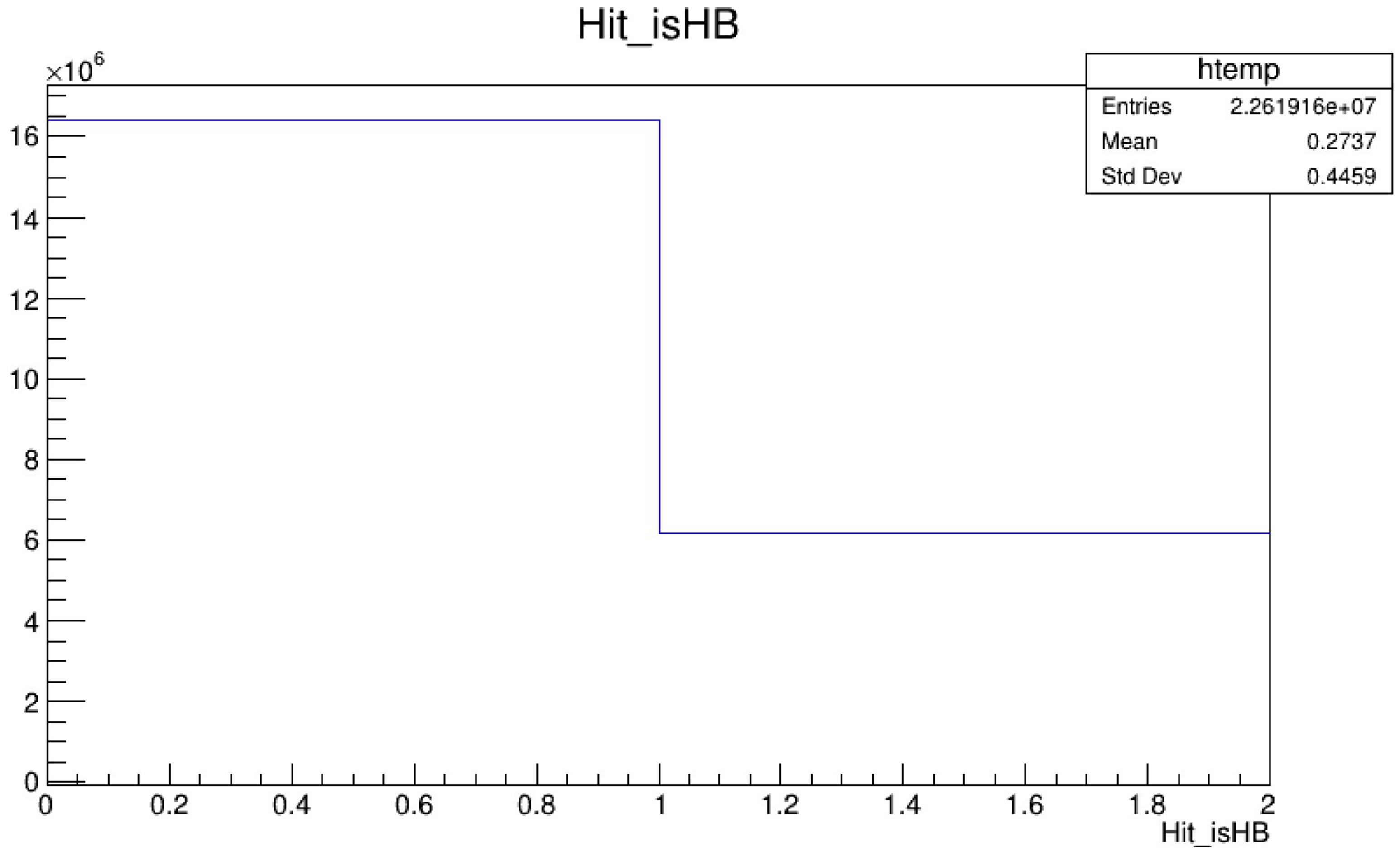
recHit energy



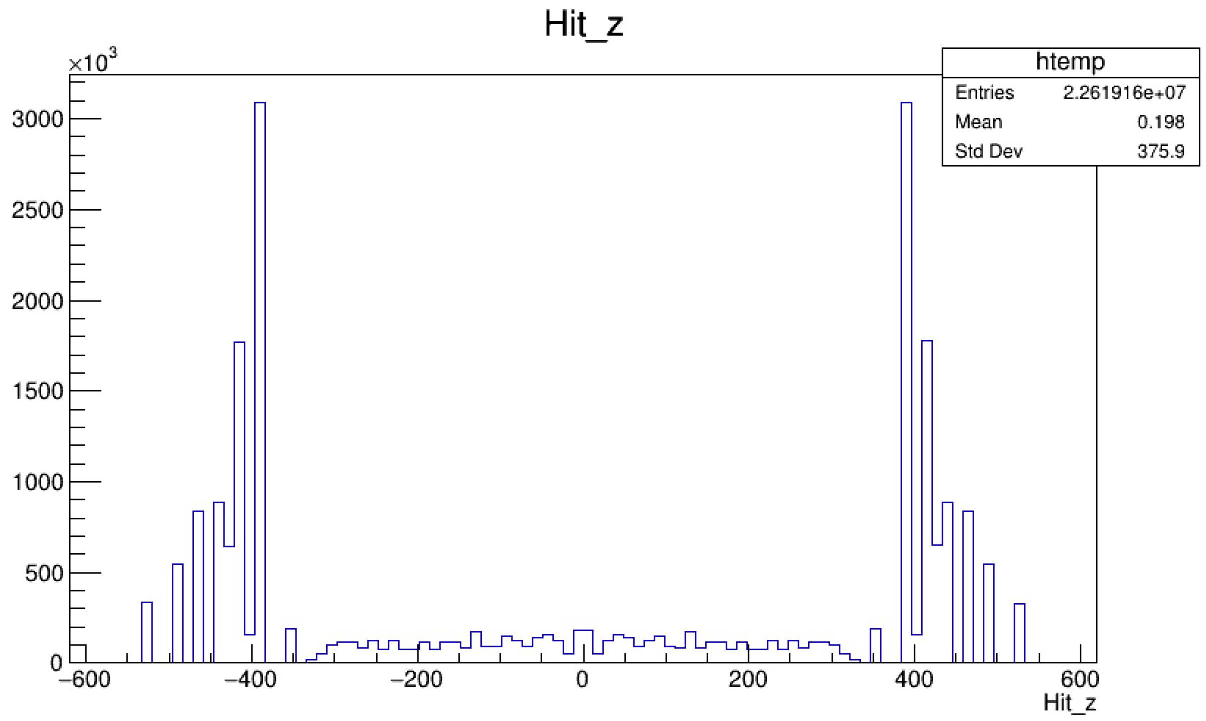
recHit depth

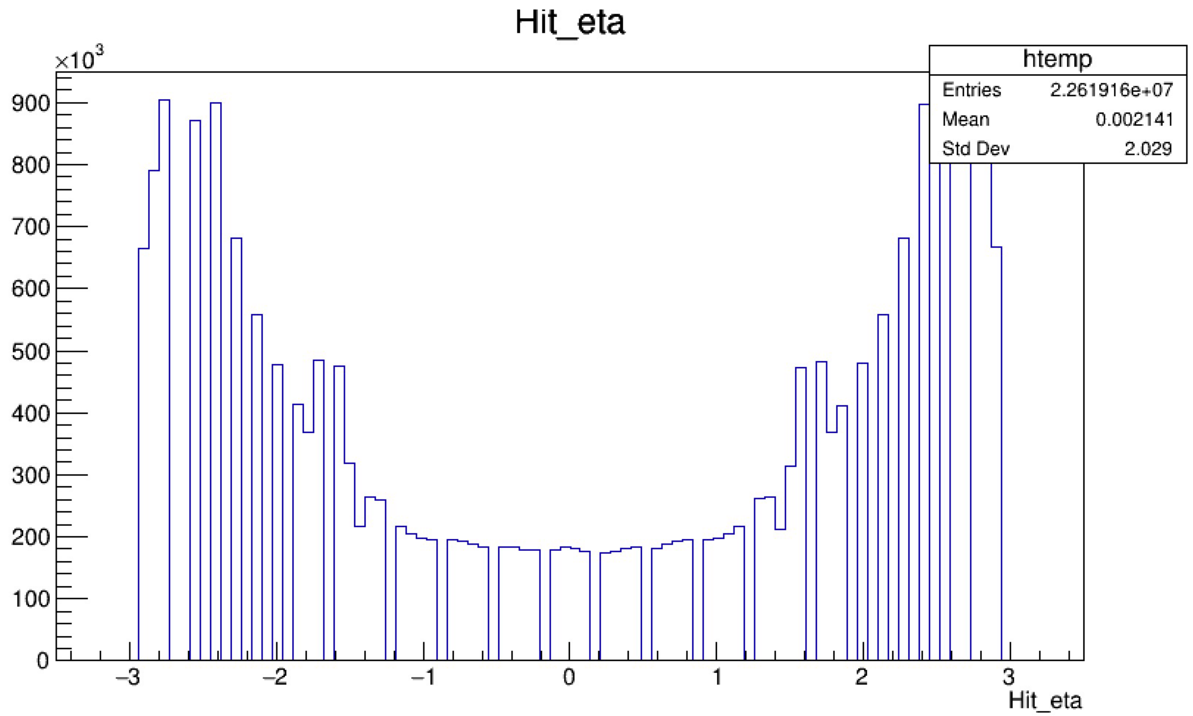


recHit in barrel? (0 means endcap, 1 means barrel)

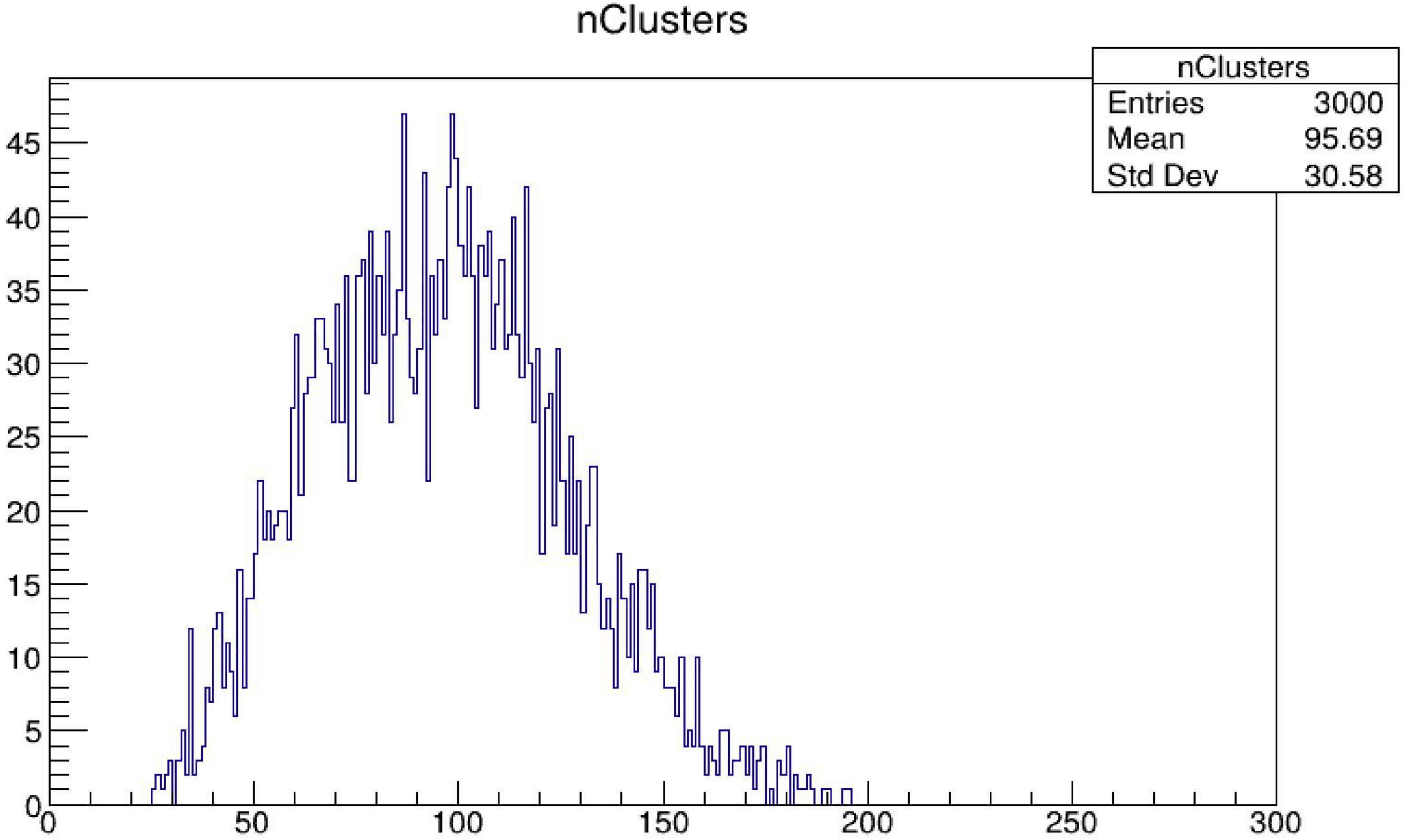


recHit z and eta





Number of truth-based clusters



NOTE: noise hits (those with parentParticle = -1) are *not* included in this distribution