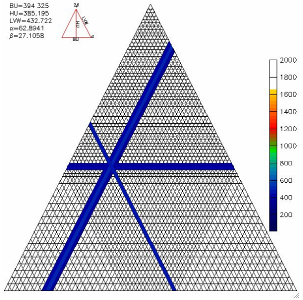
AI-assisted detector clustering, design, and simulation

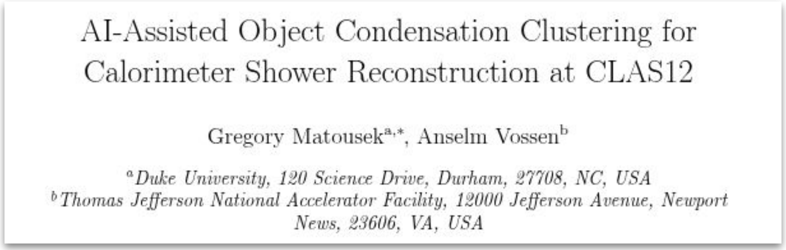
Gregory Matousek

Presentation Overview

1. AI-assisted calorimeter clustering at CLAS12 (Gregory Matousek)

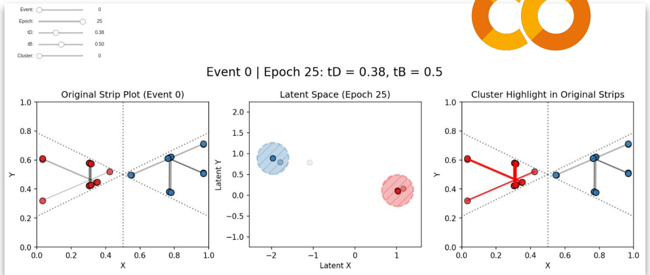
a. Google Colab Notebooks available at [this GitHub repo]
b. Recent preprint available at https://arxiv.org/abs/2503.11277



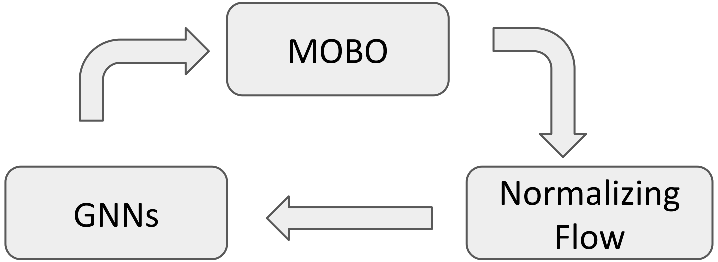

AI-Assisted Object Condensation Clustering for Calorimeter Shower Reconstruction at CLAS12

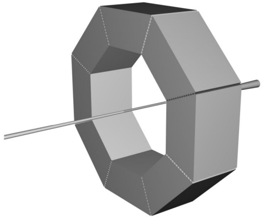
Gregory Matousek[a,*], Anselm Vossen[b]

[a] *Duke University, 120 Science Drive, Durham, 27708, NC, USA*
[b] *Thomas Jefferson National Accelerator Facility, 12000 Jefferson Avenue, Newport News, 23606, VA, USA*



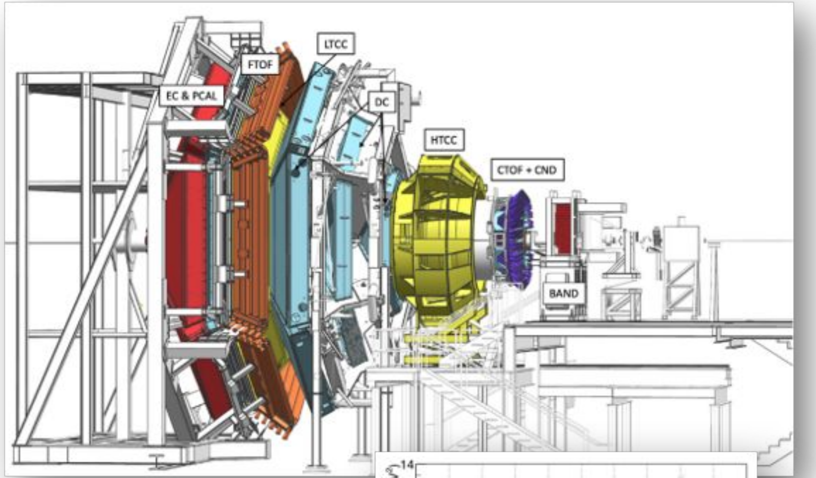
2. Automated detector optimization for a KLM at the EIC (Rowan Kelleher)

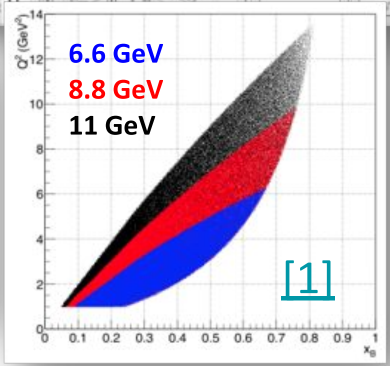




The CLAS12 Detector System

- ➢ Up to 10.6 GeV, longitudinally polarized e^- beams (~85%), fixed target experiment, six azimuthal sectors
  - ○ $2^o < \theta < 5^o$ forward tagger
  - ○ **$5^o < \theta < 35^o$ forward detector system**
  - ○ $35^o < \theta < 125^o$ central detector system
- ➢ **Comprehensive (e, π, K, p, n, γ) id**
  - ○ Several AI methods developed to improve!
- ➢ Majority of CLAS12 kinematic coverage is focused on valence quark (med-high x_B)



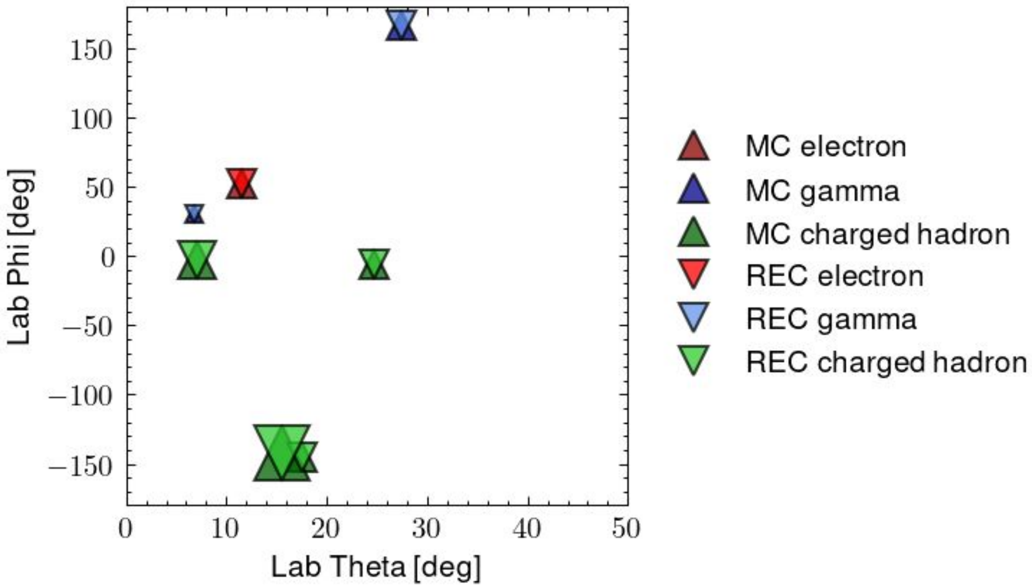


In this study, we consider the **forward ECal** that can…
- Measure photons (π^0, DVCS)
- Measure neutrons (exclusive, deuterium targ.)

Neutral Clustering at CLAS12

- Shown is the (θ,ϕ) distribution of **Monte Carlo** particles from a sample SIDIS event (upwards facing triangles)

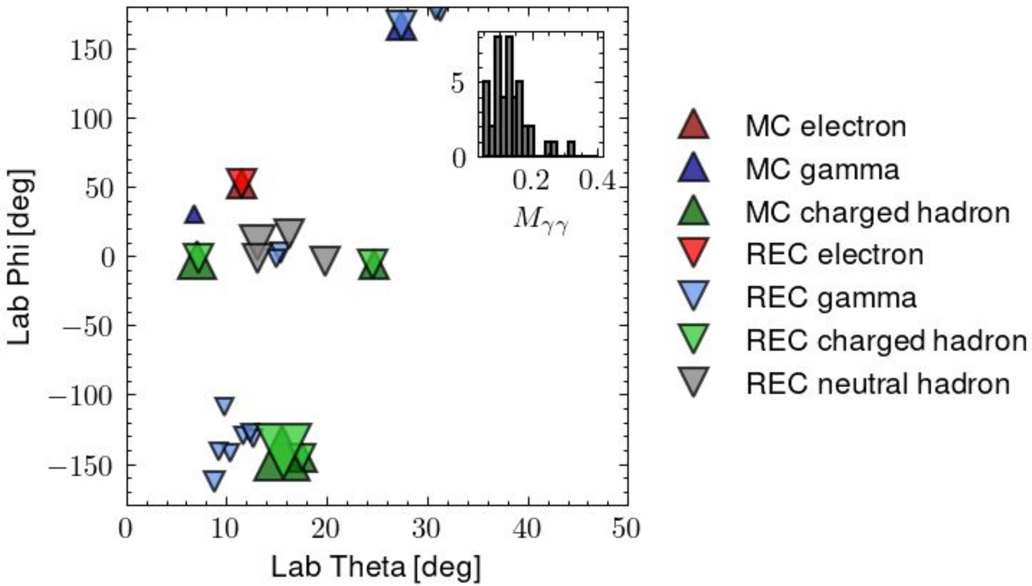
- In an ideal world, the **Reconstructed** particles (downwards facing triangles) would be exactly on top of the thrown MC particles



Neutral Clustering at CLAS12

- ➢ Shown is the (θ,ϕ) distribution of **Monte Carlo** particles from a sample SIDIS event (upwards facing triangles)

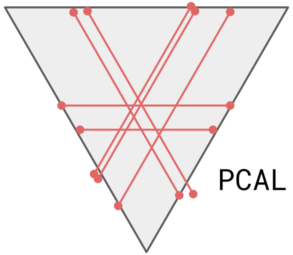
- ➢ However, issues in neutral particle clustering lead to many false neutrals being reconstructed

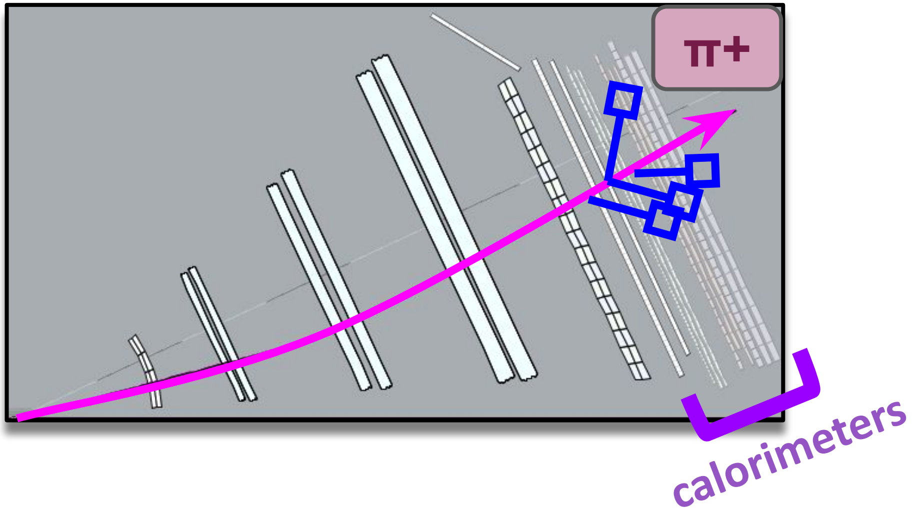


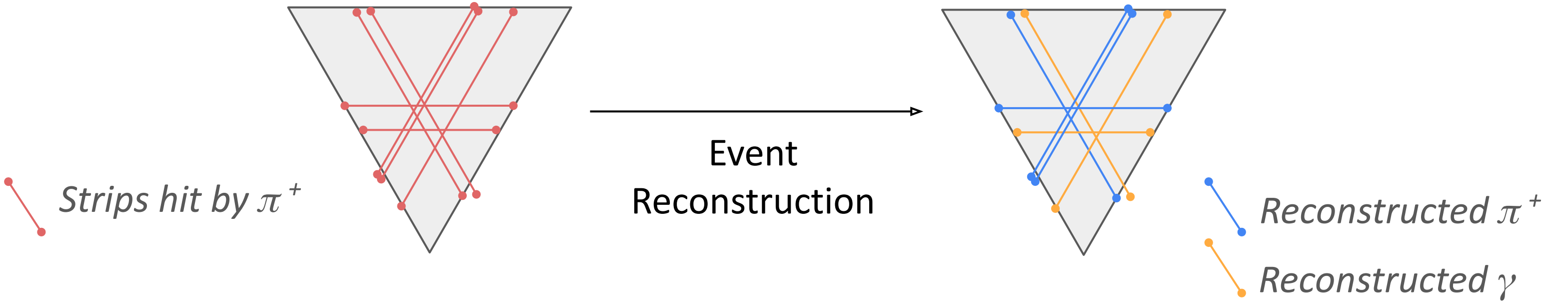
*Non-combinatorial backgrounds emerge for π^0 studies for instance, where one of the photons in the pair is **fake***

What causes False Neutrals?

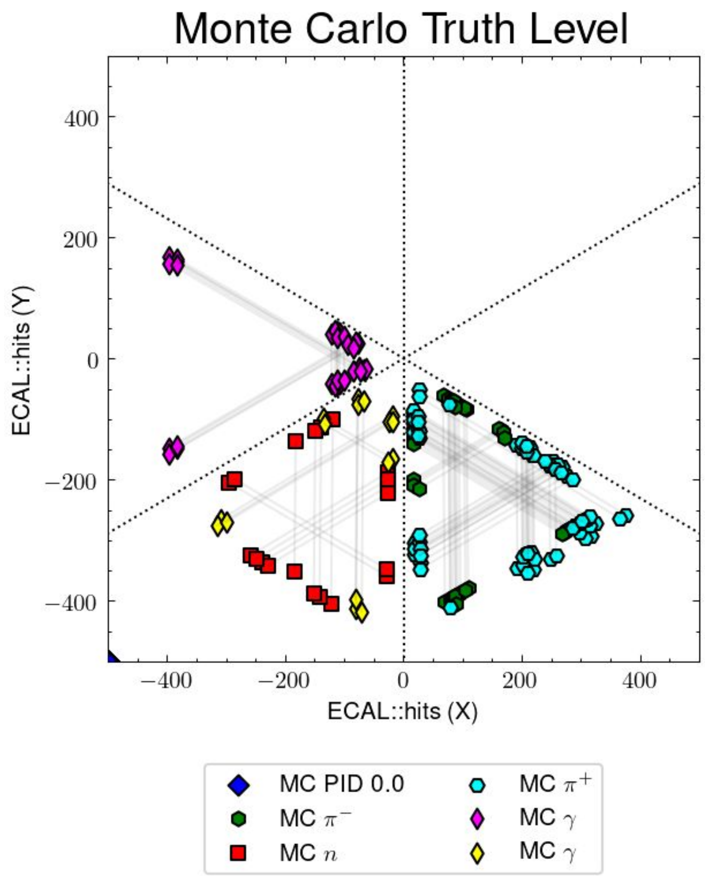
- ➢ Sampling calorimeter with lead sheets + scintillators triangular layout
- ➢ Clusters identified by looking for 3-way intersections
  - ○ Hadronic secondaries can cause disjointed, sizeable clusters to form in later layers, tricking pipeline into thinking they are their own particle







Sample Monte Carlo Event

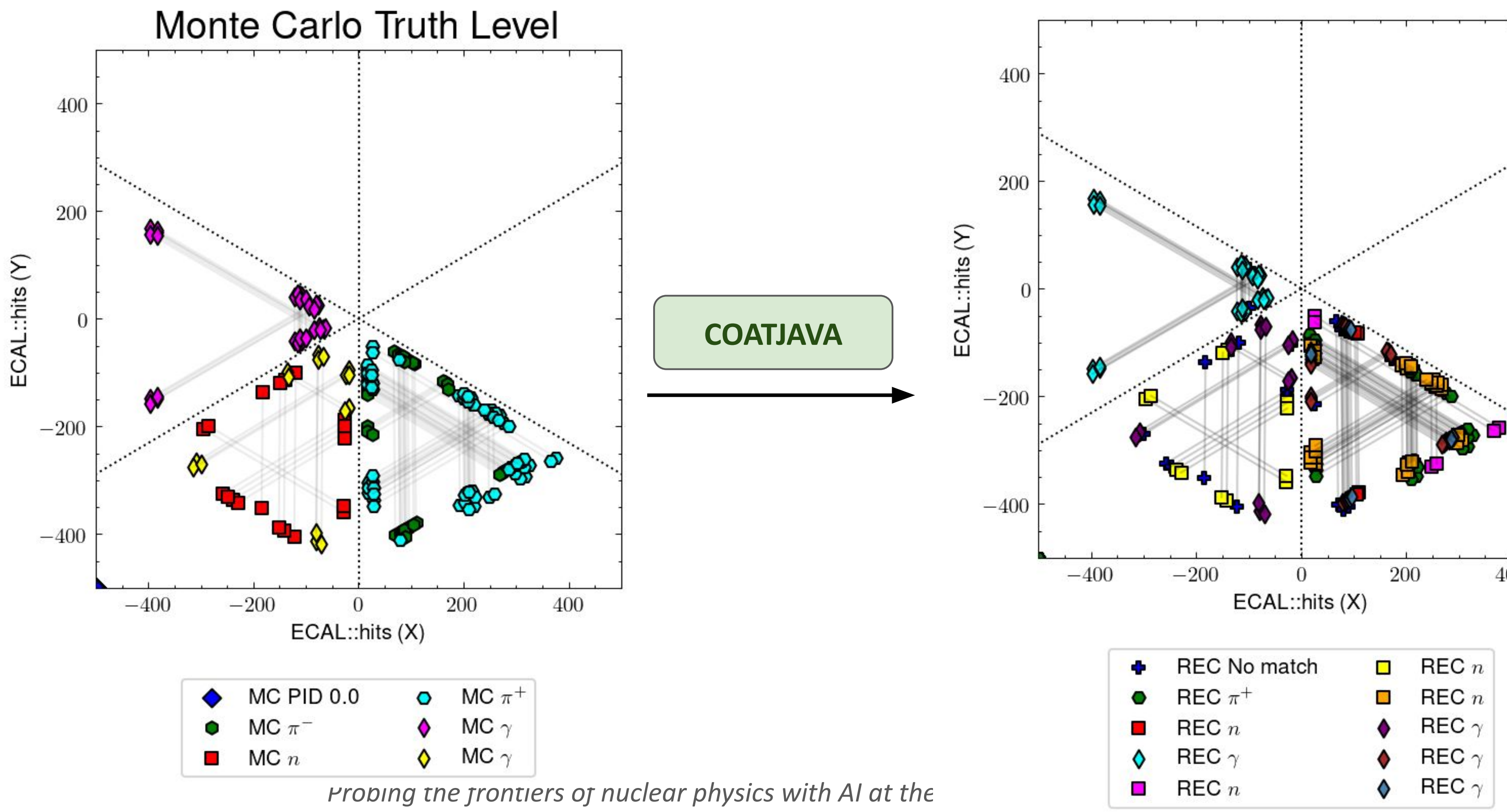


- (Left) CLAS12 ECal hits and their Monte Carlo particles
  - **Colors** → Different particles
  - **Shapes** → Different MC PIDs
- PCAL, ECin, and ECout are overlaid

Features per strip

- (x_1,y_1,z_1) and (x_2,y_2,z_2)
- Energy deposited
- Time
- Layer Number

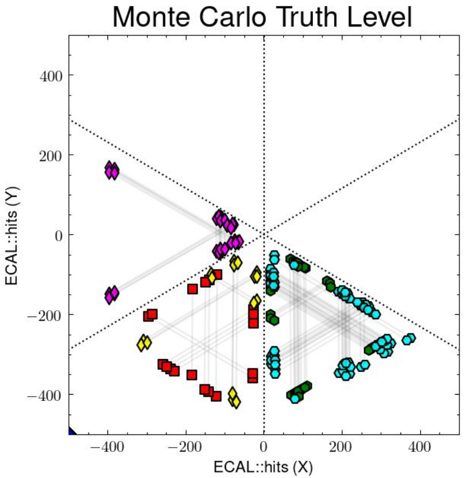
Sample Monte Carlo Event

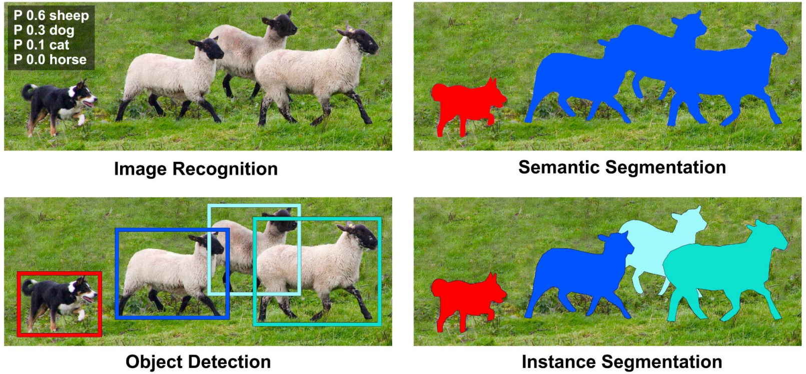


Defining the Problem

- **Input:** ***Point Cloud*** of ECAL strips with several features (layer, E, t, x, y, z)
  - From Geant4 we are aware of the Monte Carlo particle responsible for the strip hit
- **Output:** Distinct groups/clusters of strips that *belong to the same particle*

*Similar to **Image-within-Image** classification*

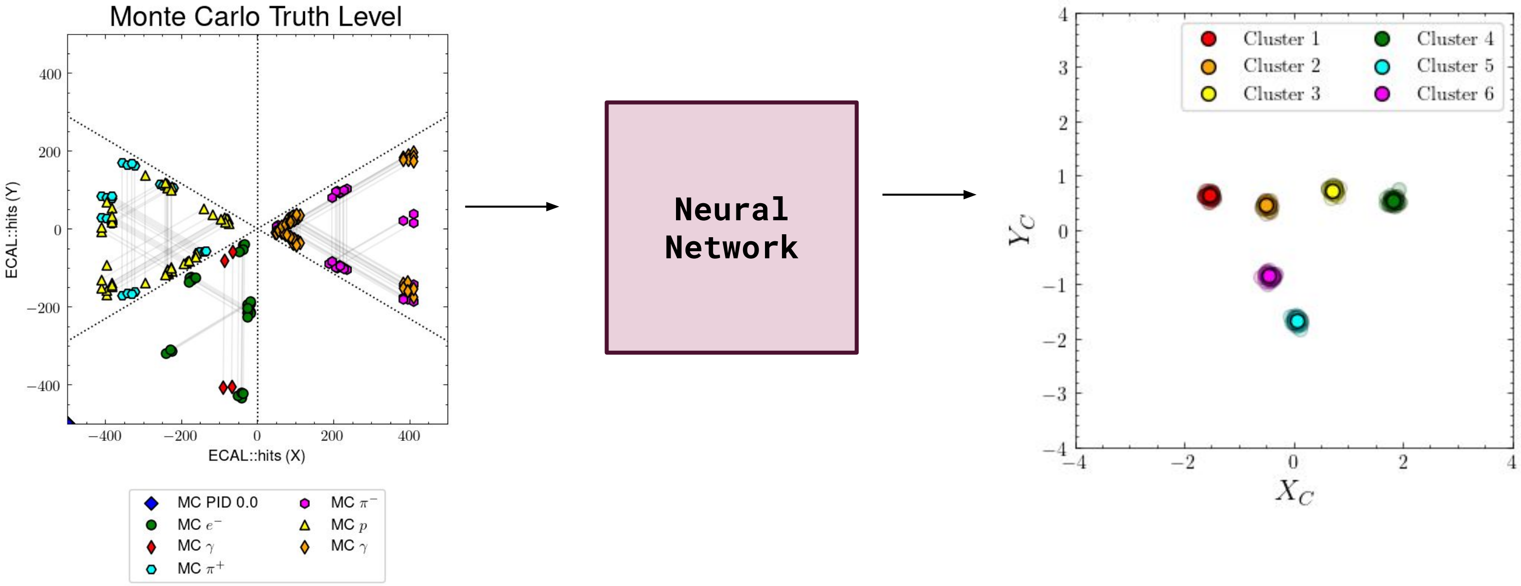




Computer Vision: MaskFormer, Mask R-CNN, YOLACT, etc.

What is ★Object Condensation★?

- Object Condensation **defines a loss function** that a neural network will try to minimize
- If this loss function is minimized, the point cloud is mapped to a clustered latent space
- *Each ECAL strip* learns its own point in the latent space **(x_c , y_c)** as well as a brightness **(0<□<1)**
- Utilized in several other nuclear physics clustering tasks ([1], [2], [3])



Object Condensation Loss

$$L_V = \frac{1}{N}\sum_{j=1}^{N} q_j \sum_{k=1}^{K}\left(M_{jk}\breve{V}_k(x_j) + (1 - M_{jk})\hat{V}_k(x_j)\right).$$

$$q_i = \operatorname{arctanh}^2 \beta_i + q_{\min}$$

Attractive Loss

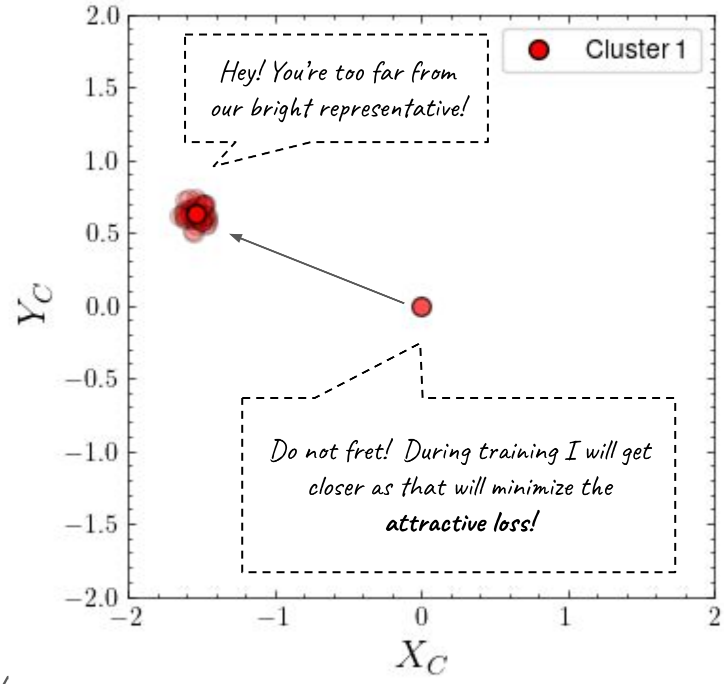
Each individual strip calculates one piece of the **attractive loss**

Very similar to E&M **U = qV**

For each strip (j), punish the loss function the further it is from the *brightest beta* for its particle (k)

The *brightest* strip for particle (*k*) is αk

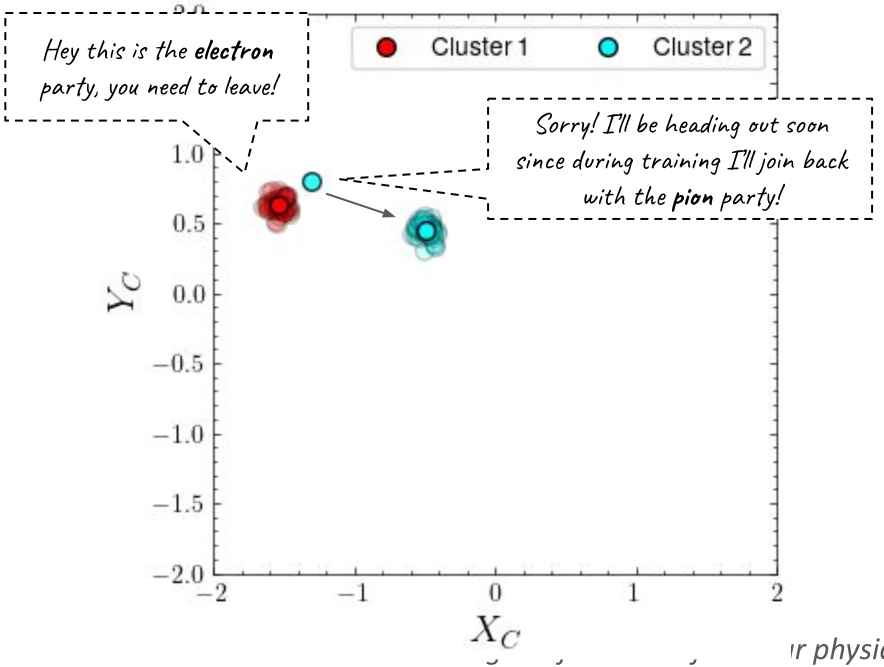
$$\breve{V}_k(x) = ||x - x_\alpha||^2 q_{\alpha k}$$



Object Condensation Loss

$$L_V = \frac{1}{N}\sum_{j=1}^{N} q_j \sum_{k=1}^{K}\left(M_{jk}\breve{V}_k(x_j) + (1 - M_{jk})\hat{V}_k(x_j)\right).$$

Repulsive Loss



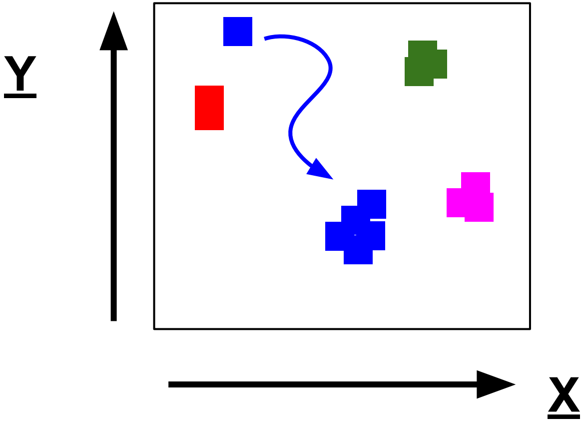
Each individual strip calculates **K-1** pieces of the **repulsive loss**

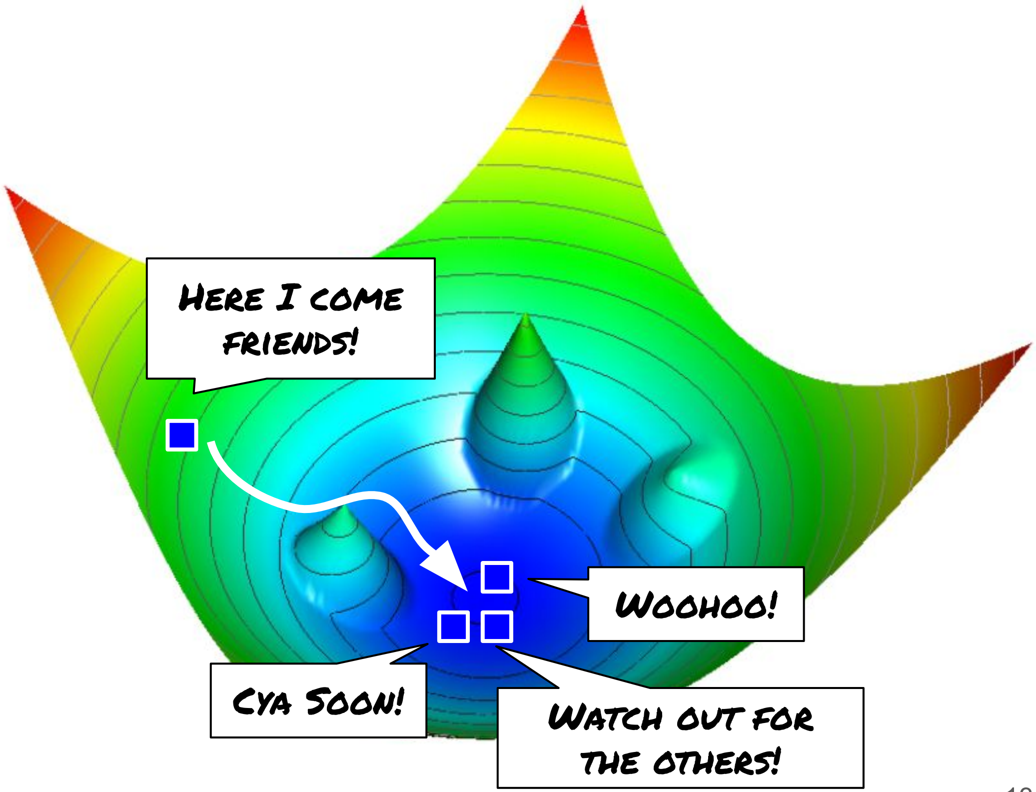
$$\hat{V}_k(x) = \max(0, 1 - ||x - x_\alpha||)q_{\alpha k}.$$

For each strip (j), punish the loss function the closer it is to the *brightest beta* of <u>any other particle</u> (k)

The *brightest* strip for particle (*k*) is αk

Object Condensation Attractive & Repulsive





(Right) The total potential V experienced by the blue square as it navigates past 3 unaffiliated objects (peaked condensation points) towards its clustering home (the bottom of the well, another condensation point)

Object Condensation Loss

$$L_\beta = \frac{1}{K}\sum_k (1-\beta_{\alpha k}) + s_B \frac{1}{N_B}\sum_i^N n_i \beta_i,$$

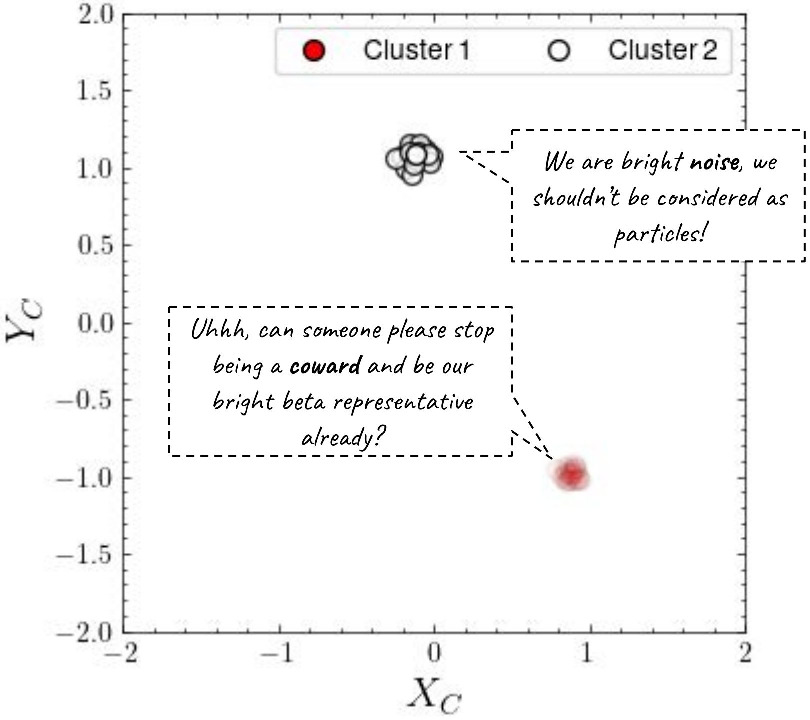
Coward Loss

For each particle (k) , punish the **coward loss** if the object's *brightest beta* is dim (near 0)

Noise Loss

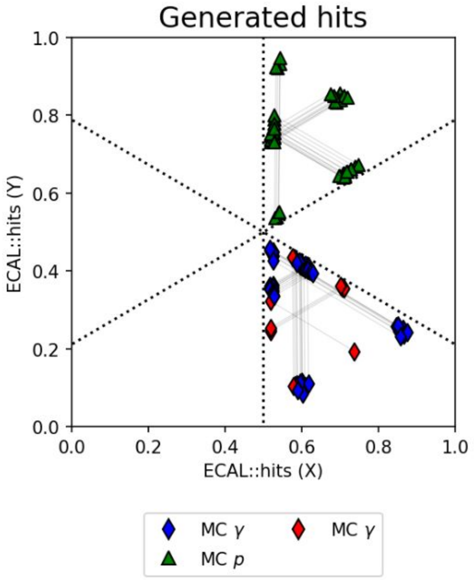
For each strip (i) punish the **noise loss** if the strip is noise (ex: 0-padded) and has a high *brightness beta*

Here n_i is a bit that is 1 if strip (i) is noise



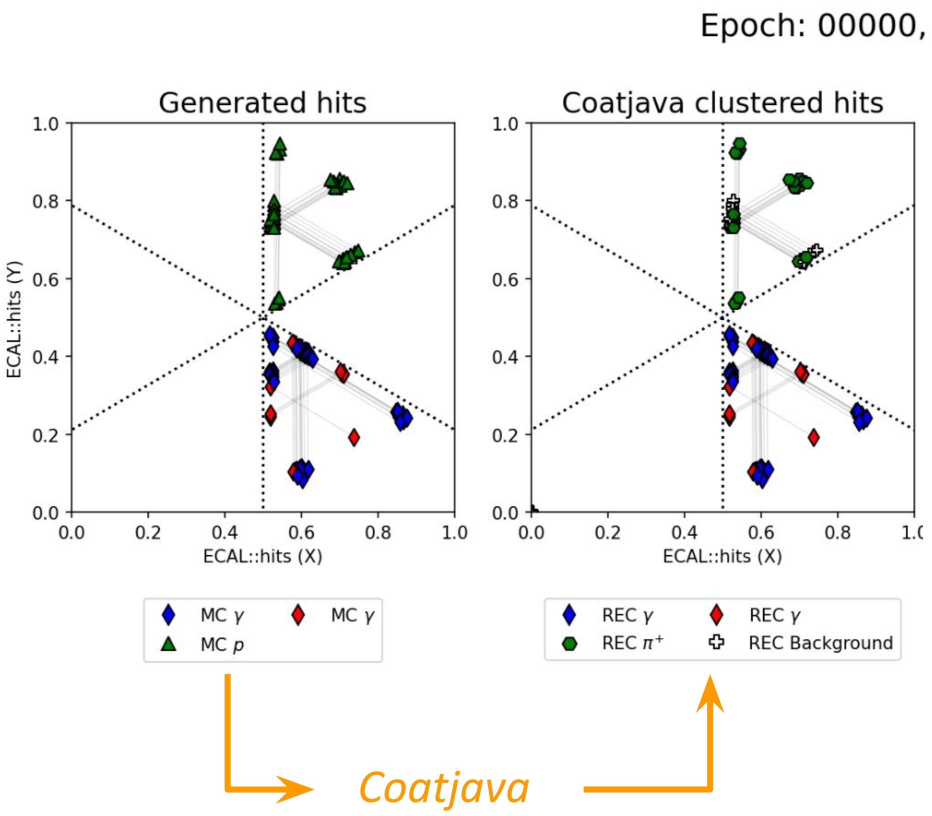
Minimizing the Object Condensation Loss

Epoch: 00000, Batch: 00010



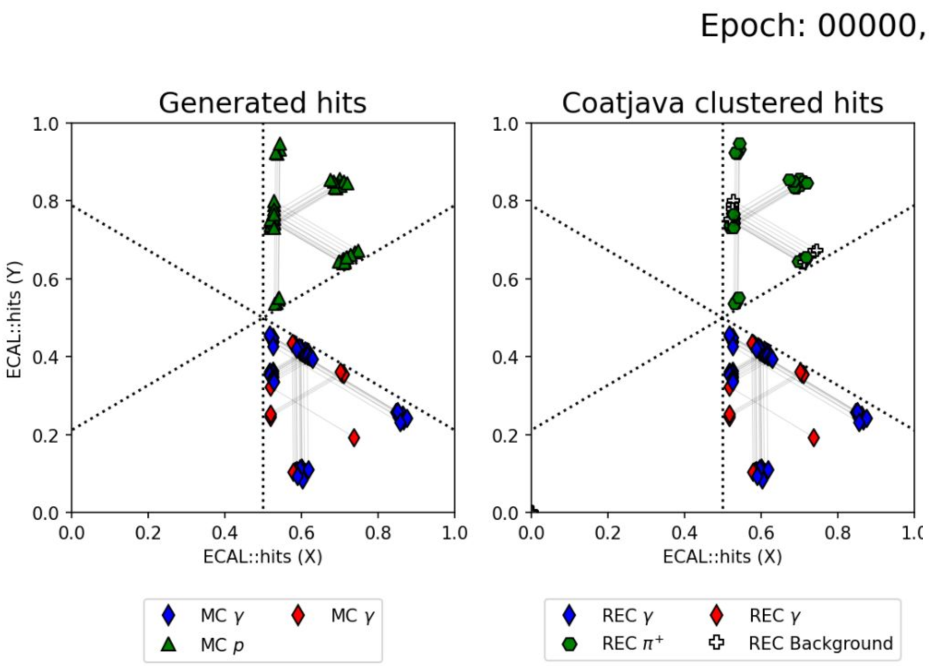
Sample event with 2 photons, 1 proton

Minimizing the Object Condensation Loss



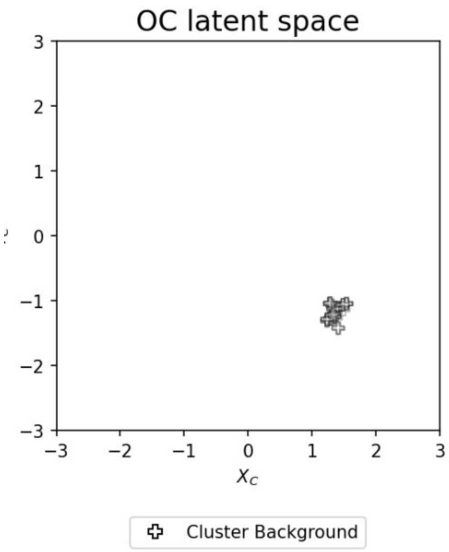
Using the pre-built CLAS12 reconstruction software, we see a near 1-to-1 reconstruction

Minimizing the Object Condensation Loss



Each hit is mapped to a point in the latent space.

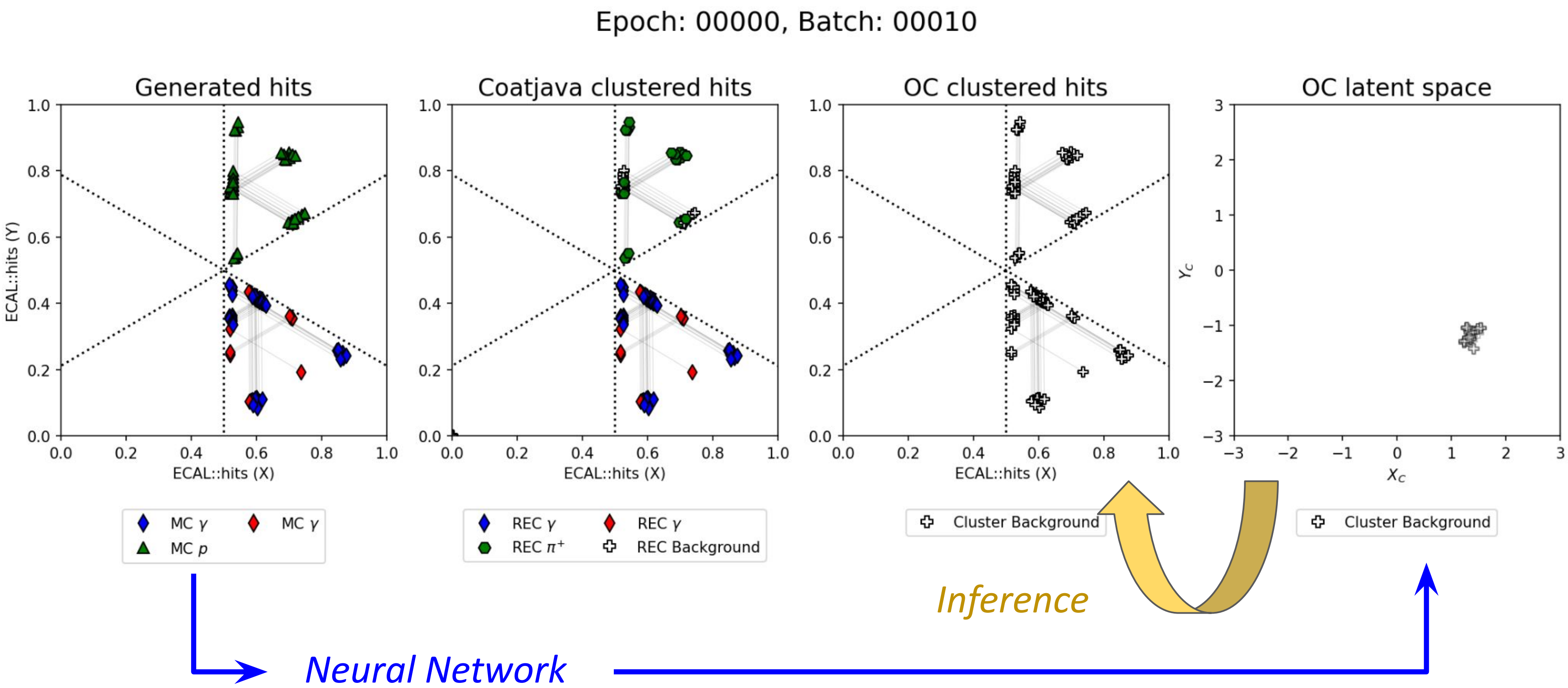
In latent space, these hits currently have low β … the model thinks they are background



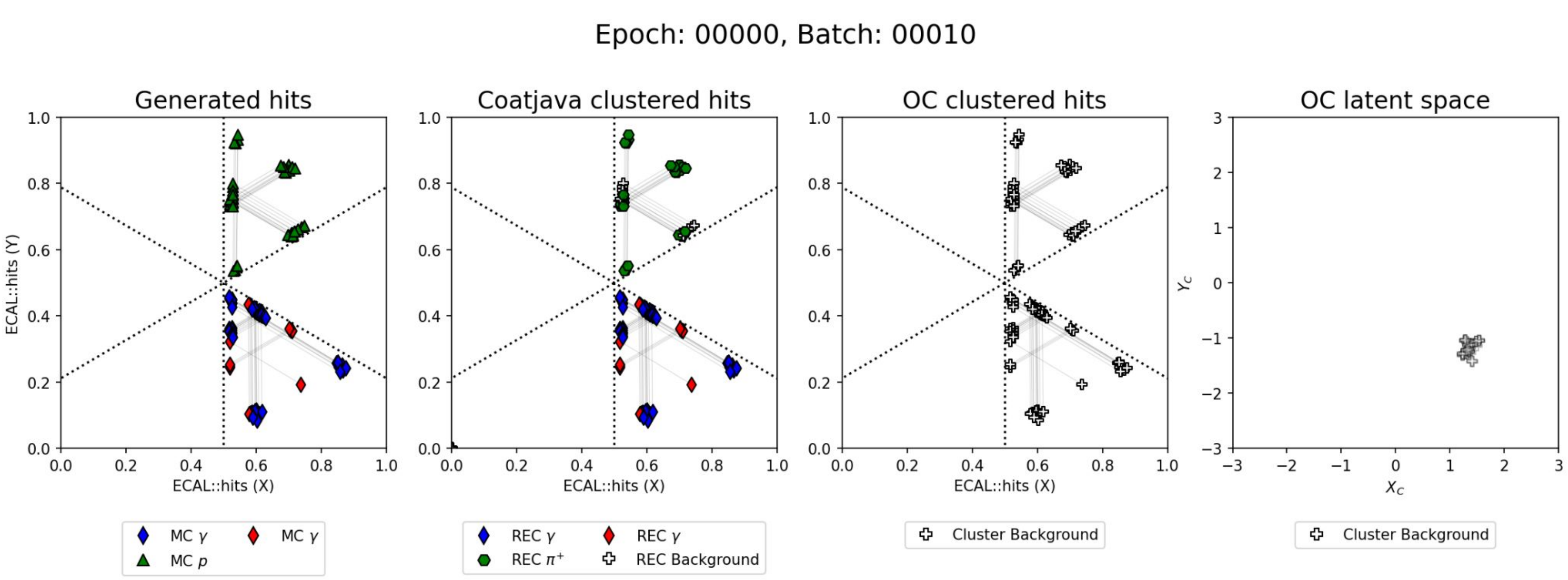
Train more!

Neural Network

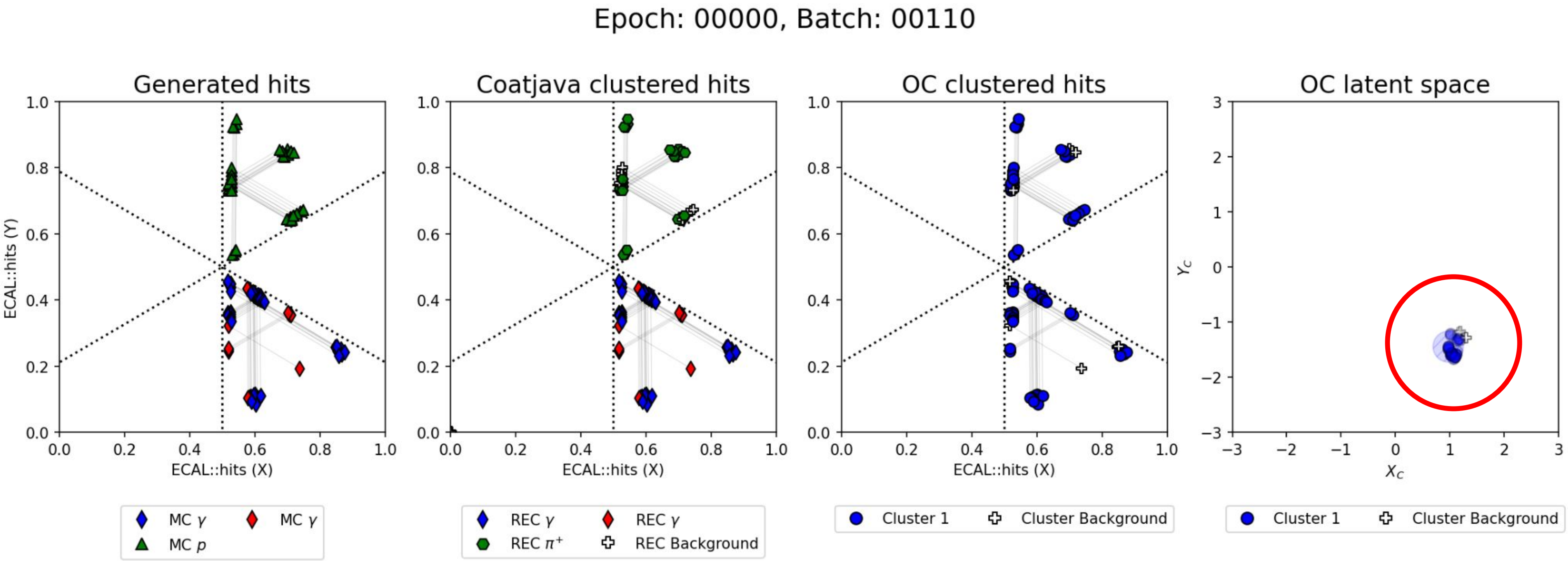
Minimizing the Object Condensation Loss



Minimizing the Object Condensation Loss

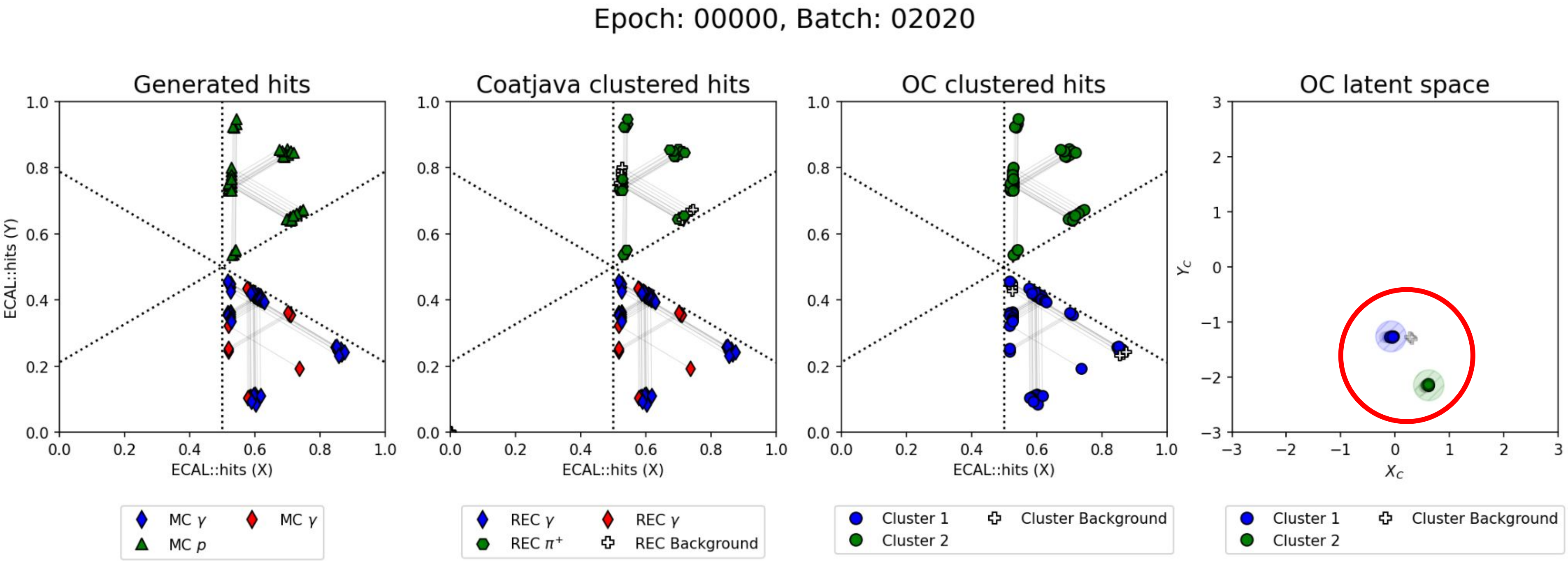


Minimizing the Object Condensation Loss



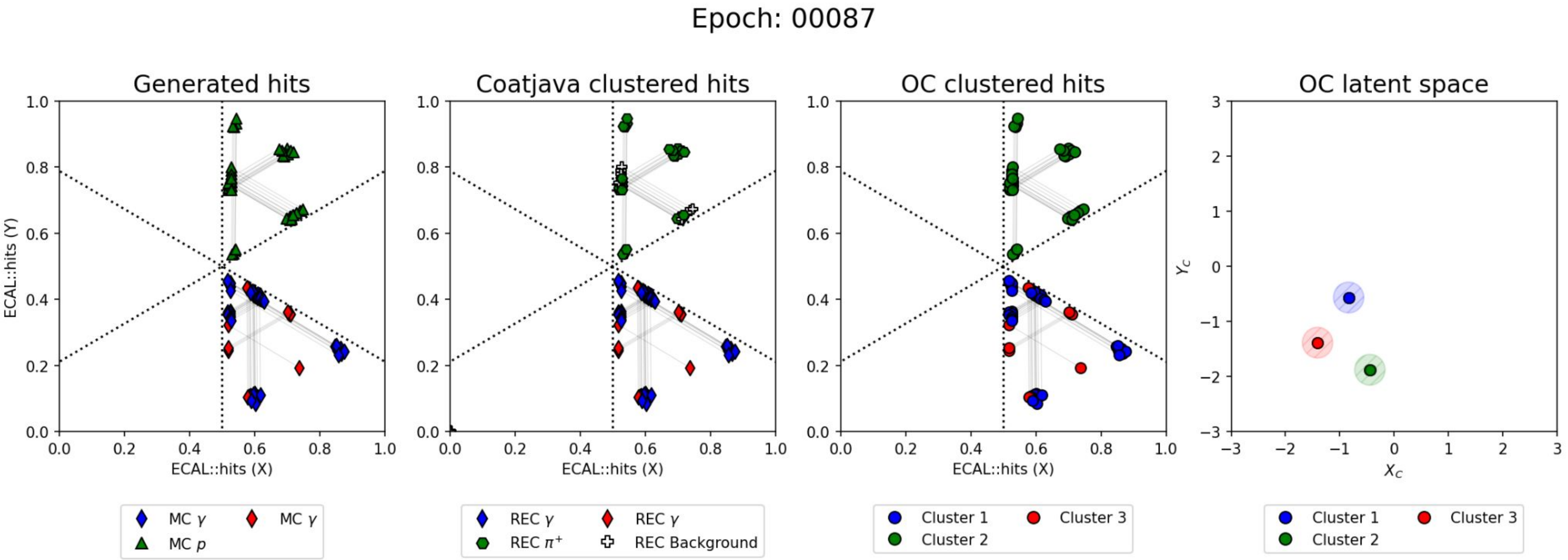
★ Model **learns** that these hits are **not noise**, but considers them one big cluster

Minimizing the Object Condensation Loss



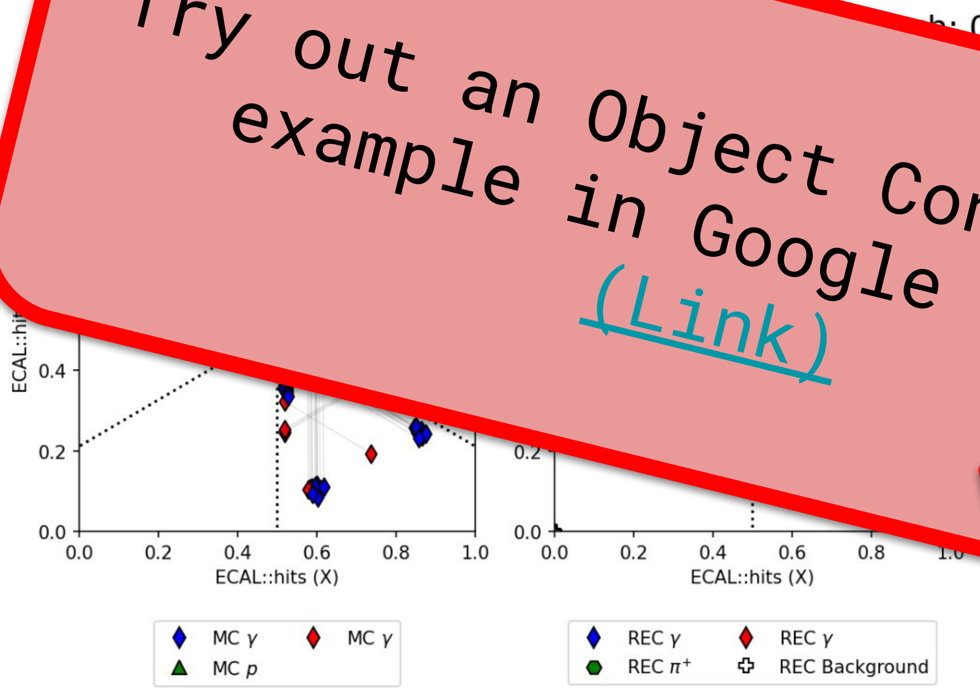
★ Later, the model is able to figure out **different sectors** likely contain **different particles**

Minimizing the Object Condensation Loss

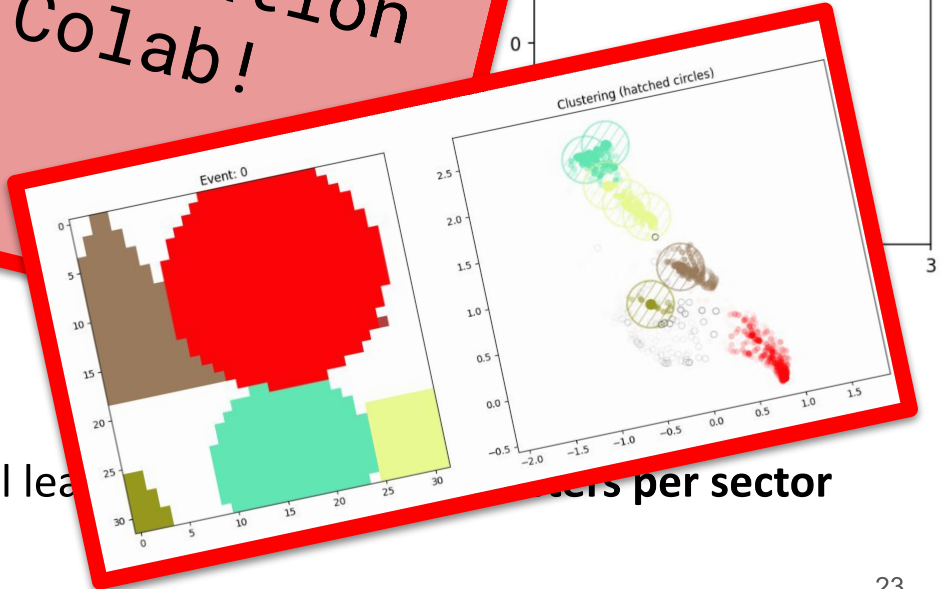


★ After some additional training, the model learns to find **multiple clusters per sector**

…izing the Object Condensation Loss



Try out an Object Condensation example in Google Colab!
[(Link)]



★ After some additional training, the model lea… **…ers per sector**

Network Architecture

Event-by-Event Input: $x \in \mathbb{R}^{V \times F}$

V = 150 (strips per event)

F = 17 (features per strip)

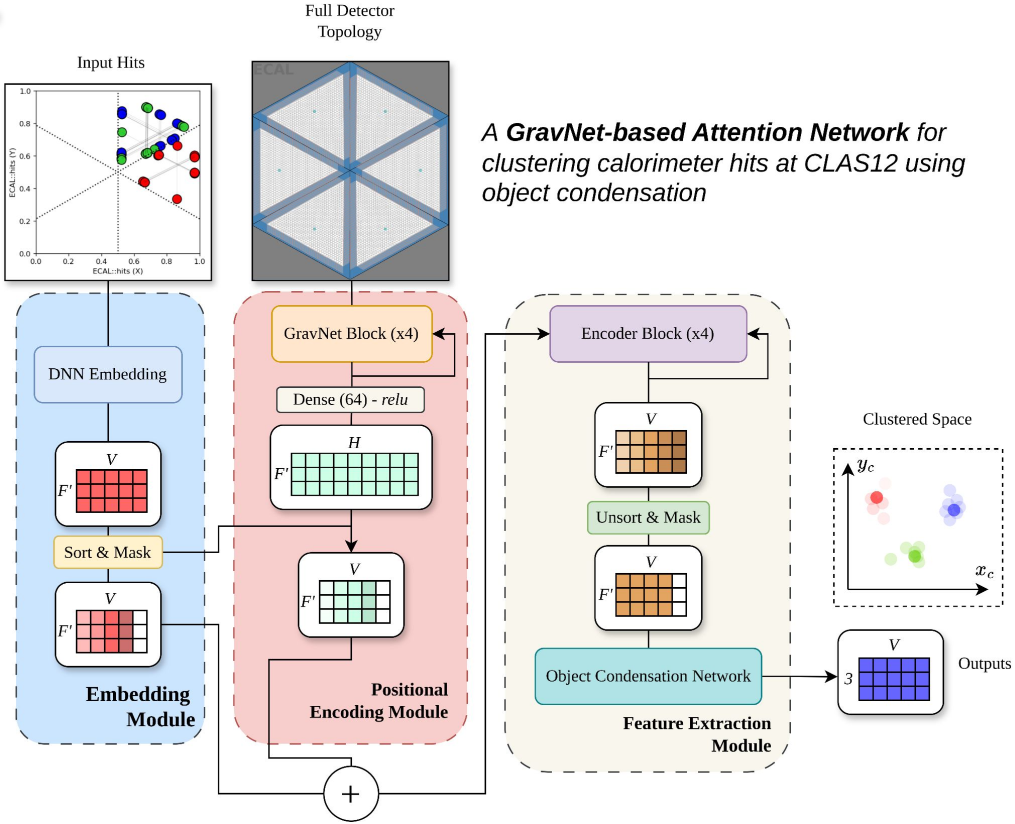
➔ Two (x,y,z) for strip endpoints

➔ Energy and time

➔ 9 one-hot encodings for layer

Event-by-Event Output: $y \in \mathbb{R}^{V \times 3}$

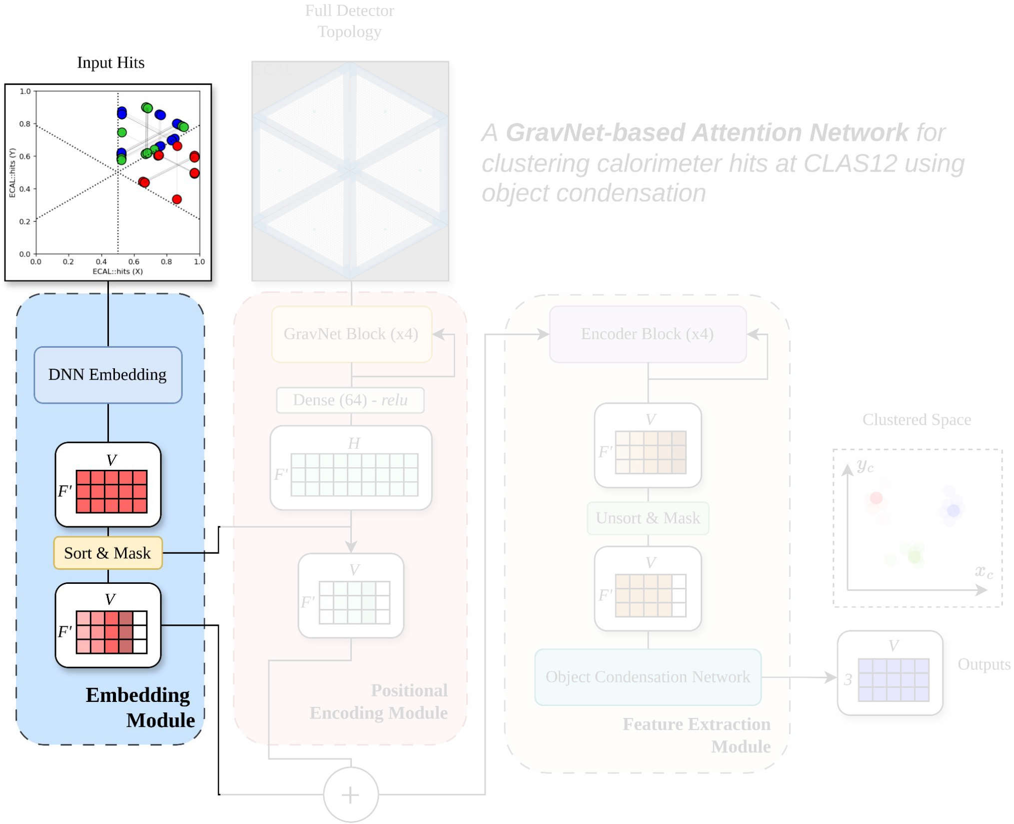
➔ (x_c, y_c, β) for object condensation



*A **GravNet-based Attention Network** for clustering calorimeter hits at CLAS12 using object condensation*

Network Architecture

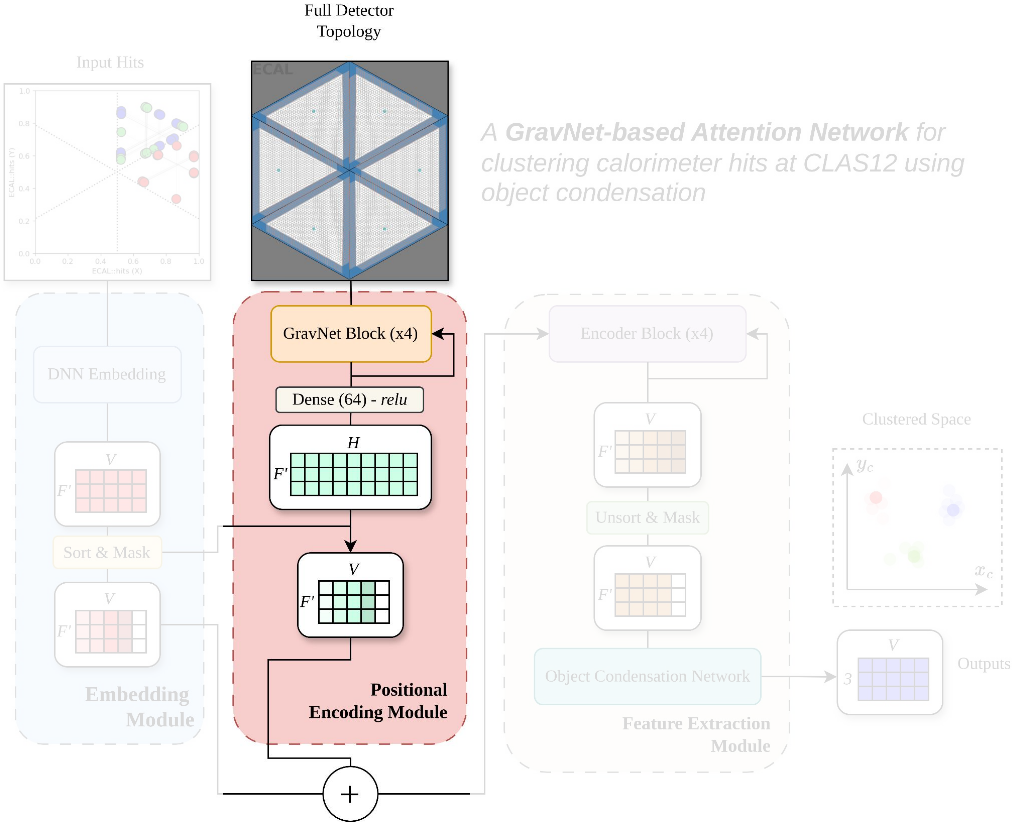
Embedding Module learns hidden representation for each strip in an event. Sorts hits into a sequence.



Network Architecture

Embedding Module learns hidden representation for each strip in an event. Sorts hits into a sequence.

Positional Encoding Module trains GravNet [1] layers on full ECal geom.
⟶ GravNet: Dynamically built k-NNs.
⟶ Short range context.
⟶ Added to hit tokens in sequence.

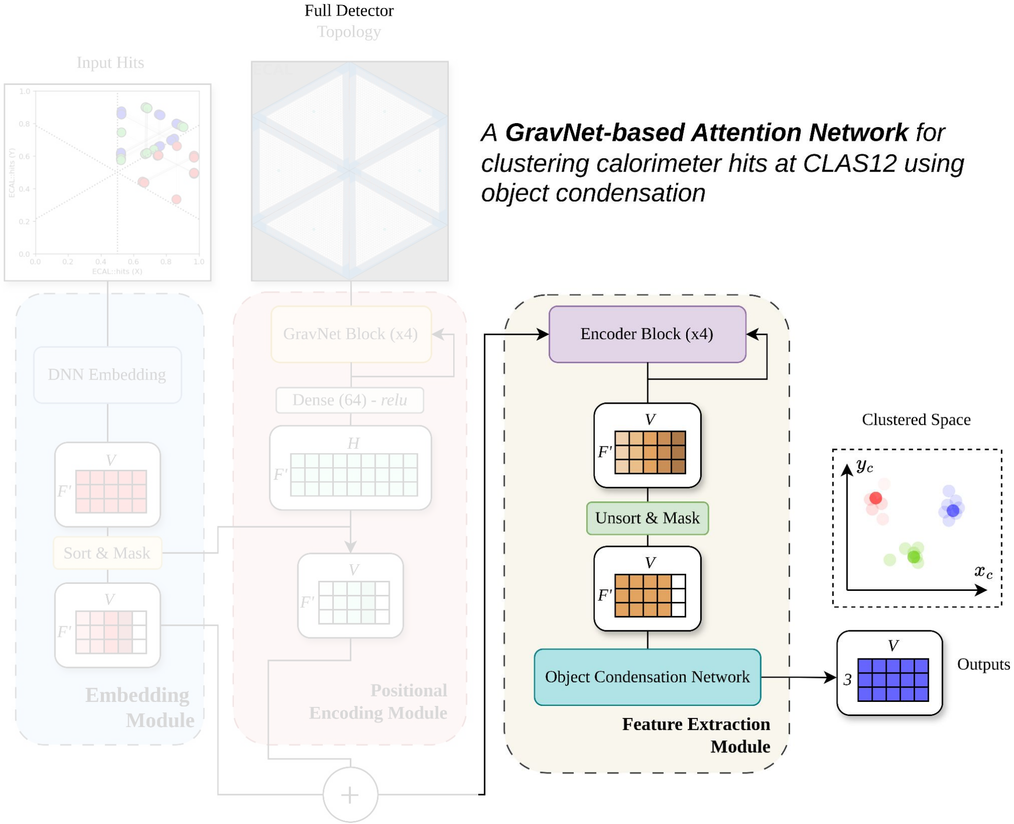


Network Architecture

Embedding Module learns hidden representation for each strip in an event. Sorts hits into a sequence.

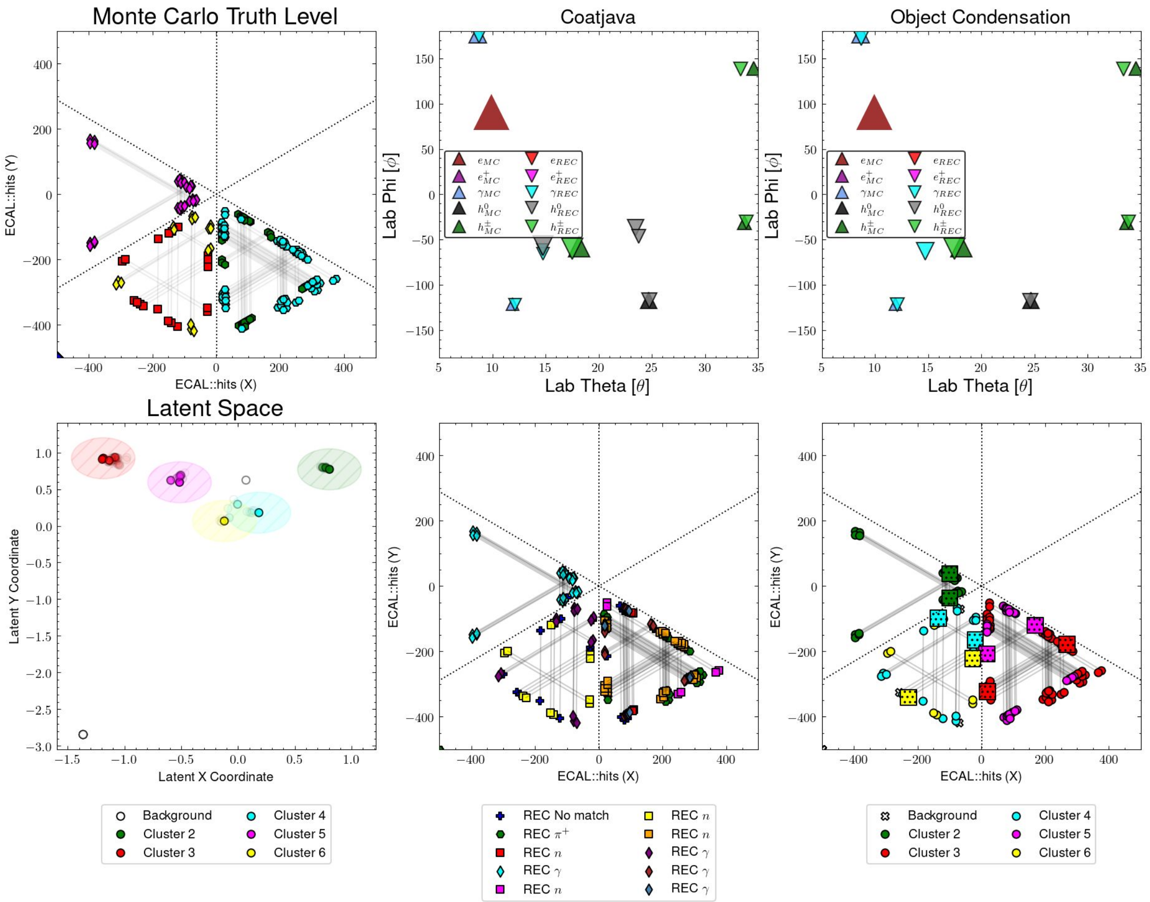
Positional Encoding Module trains GravNet [1] layers on full ECal geom.
⟶ GravNet: Dynamically built k-NNs.
⟶ Short range context.
⟶ Added to hit tokens in sequence.

Feature Extraction Module uses encoder [2] layers and DNNs to map each hit to a location in latent space
⟶ Long range context



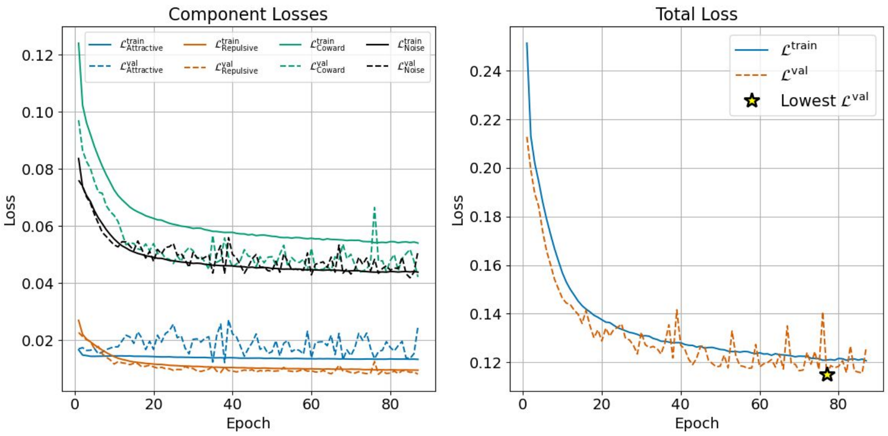
*A **GravNet-based Attention Network** for clustering calorimeter hits at CLAS12 using object condensation*

Sample Event Result



Evaluation of AI-clustering

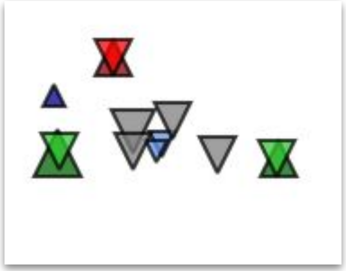
- Trained on 1M simualted DIS events [1]
- Model state @ 10 early stopping rounds
- Some hyperparameter optimization
  - Number of GravNet layers
  - Hidden dimension of encoder
- Integrated into analysis pipeline [2]



To compare **coatjava** and **object condensation** we define a metric: *trustworthiness*

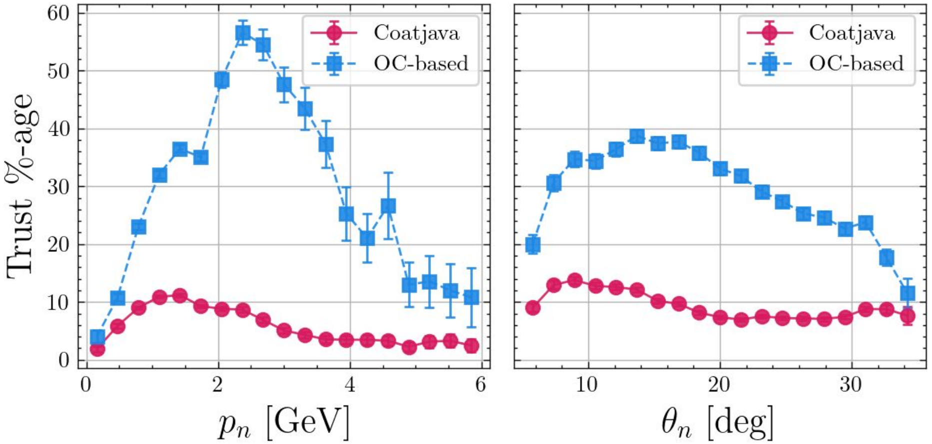
A reconstructed neutron is *trustworthy* if:

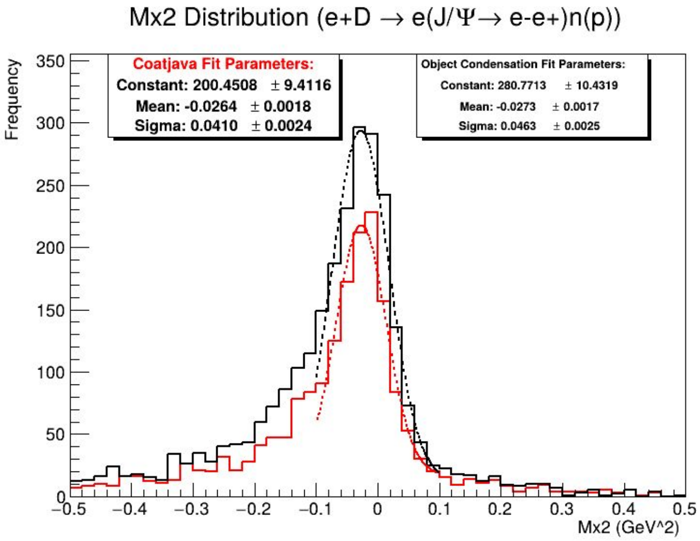
- There is a generated neutron within $\Delta\theta<4$, $\Delta\phi<4$
- There is no other reconstructed neutron within $\Delta\theta<4$, $\Delta\phi<4$



Model Evaluation on Neutrons

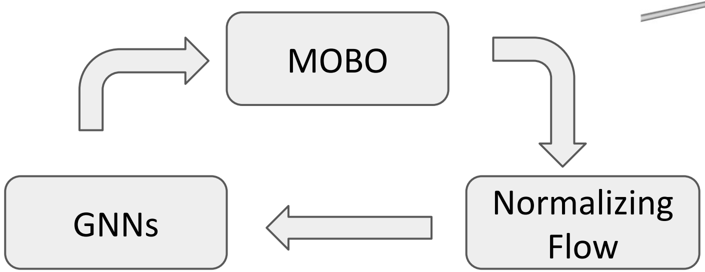
- (Left) Trustworthy-% for Coatjava and Object Condensation neutrons
  - 300% increase in reconstruction efficiency with 16% increase in yields
  - Object Condensation model ***outperforms*** existing CLAS12 reconstruction for neutrons
- (Right) Sample exclusive incoherent J/ψ for **Coatjava** and **Object Condensation**
  - 40% increase in yields when using 4-fold coincidence
  - Expected to increase with more analysis cuts

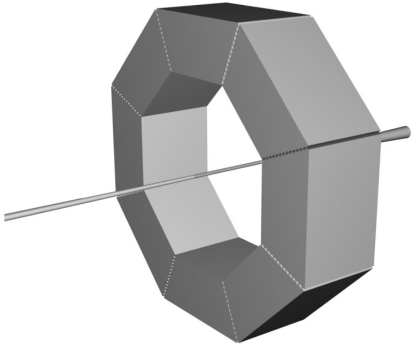


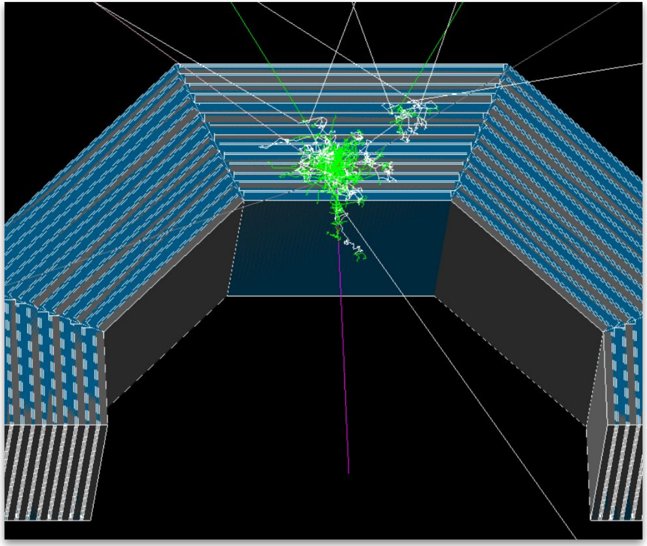


EIC 2nd Detector KLM

- A **barrel KLM** (K-long muon) detector has been proposed for the **EIC 2nd detector**
  - Measuring particle energy, momentum, and PID
  - Sampling calorimeter — layers built with **steel** and **scintillator**
- Project Goal is to optimize the detector design using **Geant4 + ParticleGun**
- Several AI models were developed for pipeline
  - NF ⟶ Reduce sim. time
  - GNN ⟶ Predict KLM energy
  - MOBO ⟶ Find best config.

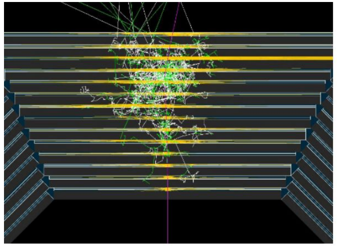




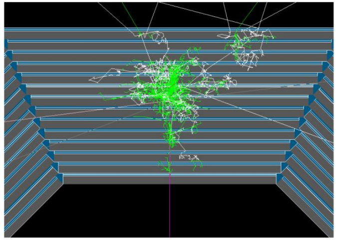


KLM Detector Geometry

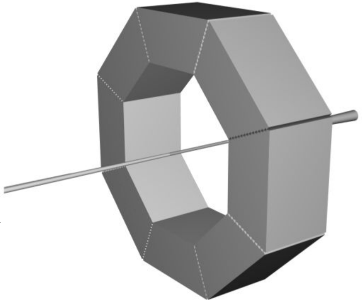
- Barrel detector formed by 8 identical **staves** — surrounding the inner detectors
- Each **superlayer** built of **scintillator strips** and **steel sheets**
- SiPMs at both ends — timing and energy
  - Many optical photons → costly simulation
- Fixed outer radius constraint
  - Optimal # of superlayers? Steel/scintillator ratio?

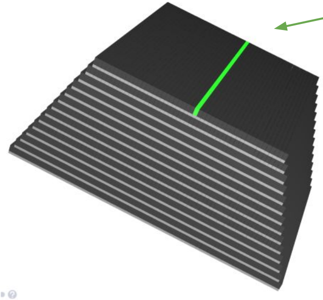


(c) With Optical Photons



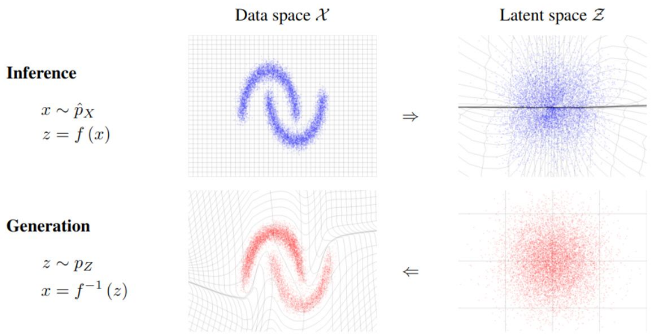
(d) No Optical Photons

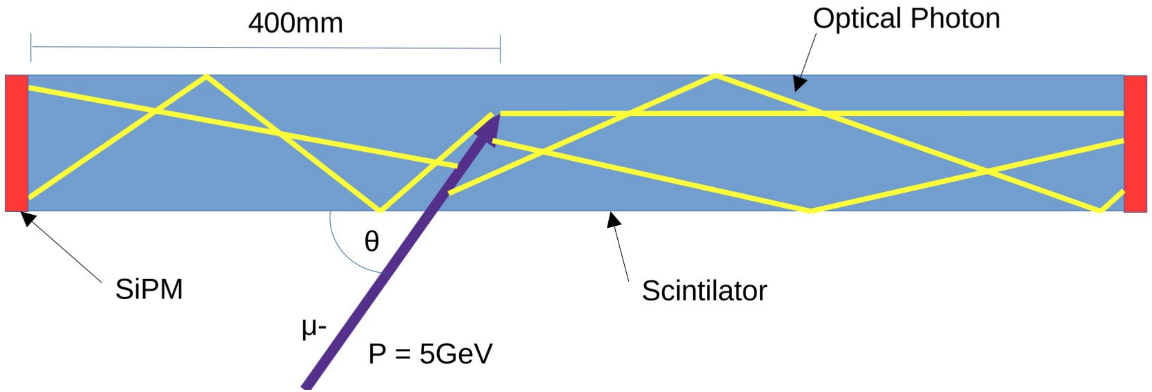




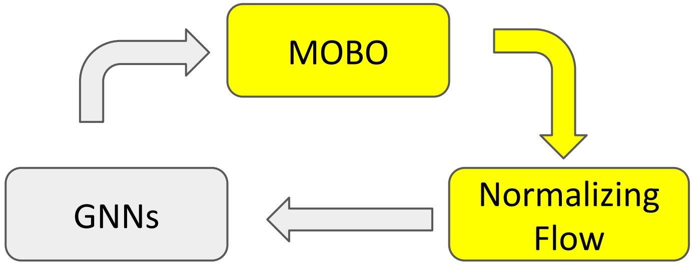
Parameterizing Optical Photons (NF)

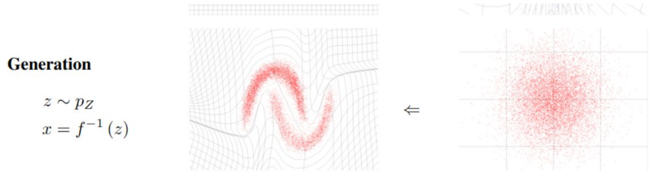
- We want to optimize the KLM → **test** different scintillator thicknesses…
  - **Problem:** We need a lot of statistics to quantify KLM performance. These simulations are costly
  - **Solution:** Parameterize the **photon yield** and **arrival time** using Normalizing Flow (NF) [1]
- For a large set of different scintillator thickness (ex: 1cm, 3cm, 5cm) **ParticleGun** events are used to model N_γ and $t_{arrival}$ as a function of PID, momentum, position

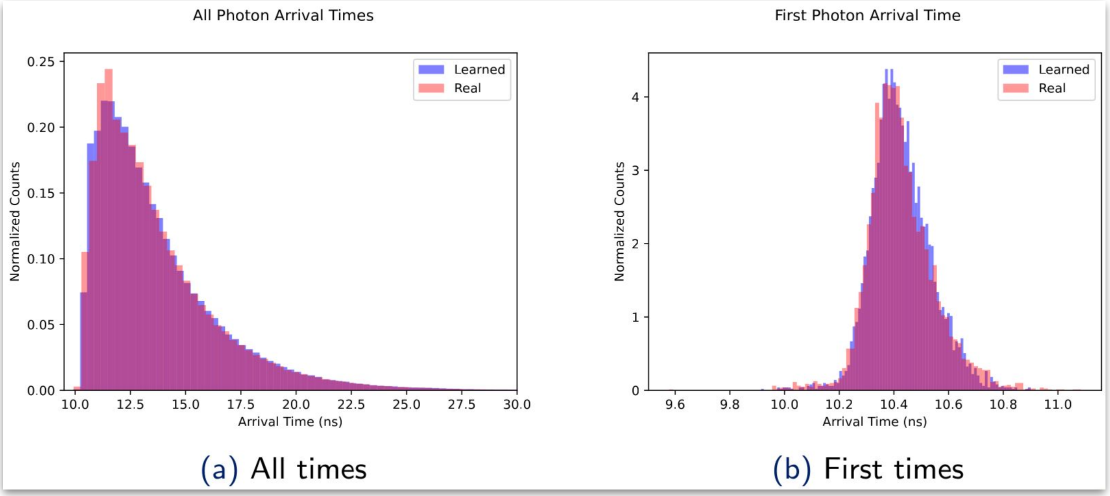




Parameterizing Optical Photons (NF)

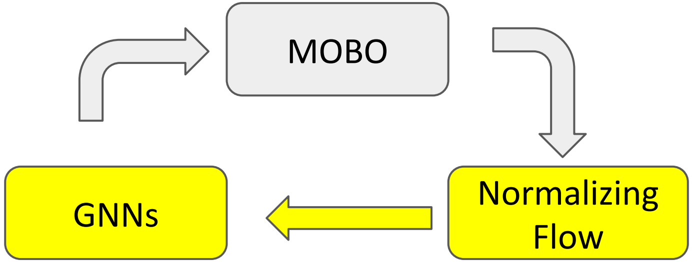






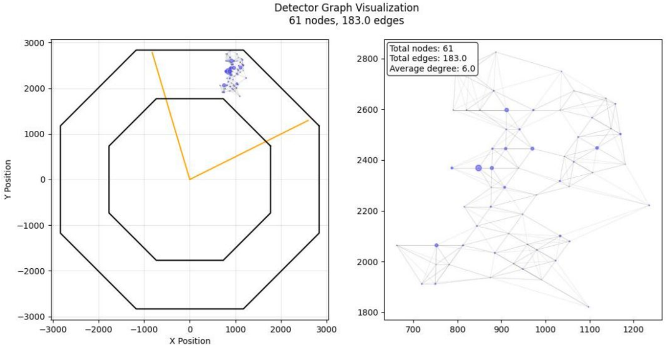
- Normalizing Flows are able to reduce simulation time by **x100**
- **Currently simulated a set of NFs for different thicknesses,** pick closest from set
- KLM resolution on the order of 100 picoseconds
- With the KLM response parameterized → obtain PID efficiency, resolution

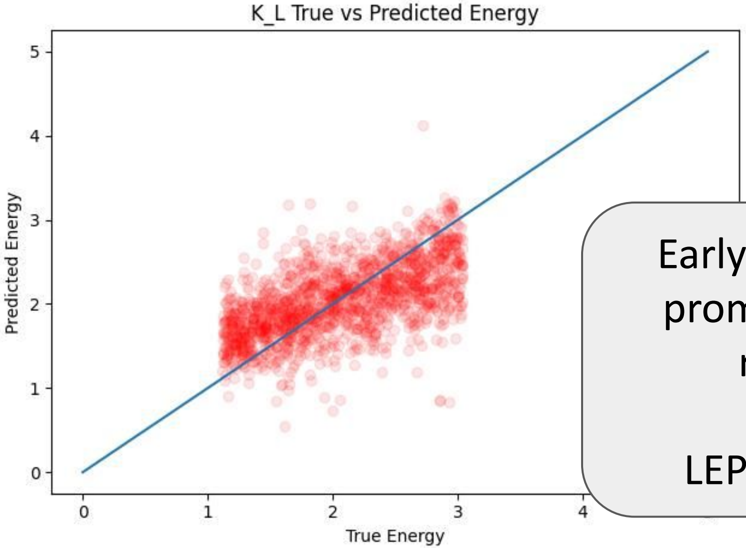
Graph Neural Networks for the KLM



- K^0_L particle gun at different **(E, θ, ϕ)**
- SiPM readouts treated as point cloud with 6-NN's

Preliminary GNN built to measure energy resolution

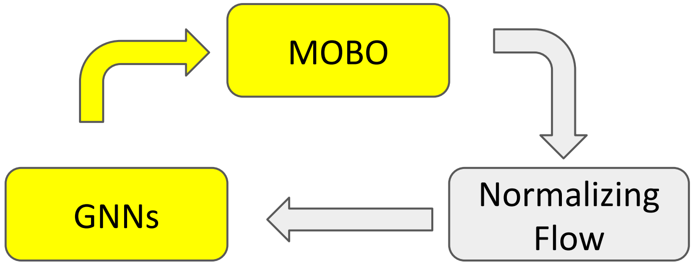


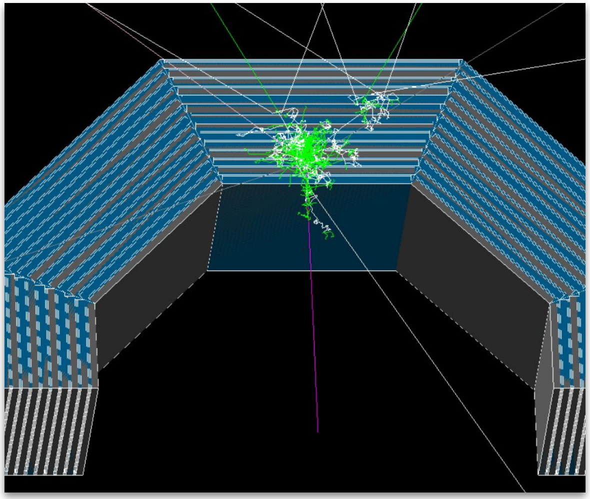


Early work shows a promising **30%/√E** resolution

LEP ⟶ 100%/√E

Ongoing Work





- Soon to be implemented **classifier-GNN** for an additional objective to be maximized
- Construct **Ax** & **Botorch** pipeline for searching the detector design space
    - Number of superlayers
    - Steel/scintillator thickness ratio per superlayer

BoTorch

- Consider the need for **KLM** clustering (overlapping neutrals?)

Conclusion/Future Outlook

An **Object Condensation-based** pipeline was developed to perform clustering of calorimeter hits at CLAS12 – first usage of clustering AI on hodoscopic detectors.

- Increases neutron reconstruction efficiency by +300%.
- Plans to incorporate computer vision models like Mask RCNN and MaskFormer to improve results ⟶ one hit can belong to multiple objects
- New collaboration with Argonne National Lab to apply AI to **ePIC barrel ECAL**

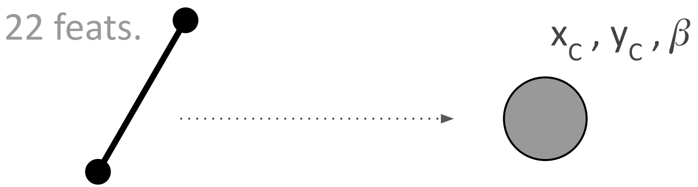
A **Multi-objective optimization** framework using Normalizing Flows and GNNs is in development for a future EIC KLM detector

- Current work (ex: by Connor Pecar) is being performed to optimize the ePIC dRICH geometry

Extra Slides

★Object Condensation★ Recap

- Input → $v_{in}(N, F)$
  - **N:** Number of nodes (in our case number of strips)
  - **F:** Number of features per node (in our case 22)



- Output → $v_{out}(N, 3)$ → *i.e. each strip learns 3 variables*
  - $\mathbf{v_{out}}$**[: , 0]** is the x-coordinate in a latent space (called $\mathbf{x_C}$)
  - $\mathbf{v_{out}}$**[: , 1]** is the y-coordinate in a latent space (called $\mathbf{y_C}$)
  - $\mathbf{v_{out}}$**[: , 2]** is the *brightness* of the node (strip) in the latent space [0,1] (called □)

Object Condensation (OC) defines a loss function $L(x_c, y_c,$ □$)$ that is *minimum* if…

1. The (x_C , y_C) of nodes that belong to the same cluster are close (**attractive loss**)
2. The (x_C , y_C) of nodes that belong to different clusters are far (**repulsive loss**)
3. Only one node per cluster has a large brightness □~1 (**coward loss**)

Object Condensation Loss

$$L_V = \frac{1}{N}\sum_{j=1}^{N} q_j \sum_{k=1}^{K}\left(M_{jk}\breve{V}_k(x_j) + (1 - M_{jk})\hat{V}_k(x_j)\right).$$

$$q_i = \operatorname{arctanh}^2 \beta_i + q_{\min}$$

Attractive Loss

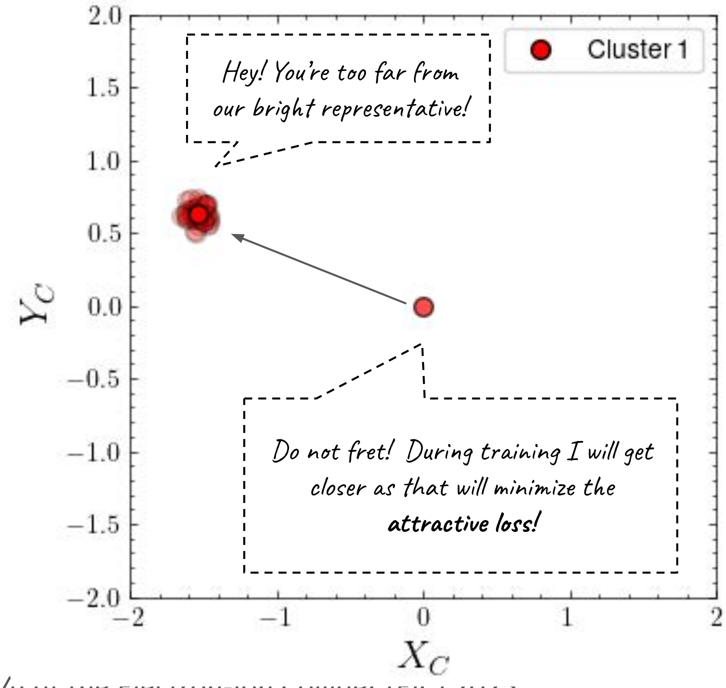
Each individual strip calculates one piece of the **attractive loss**

Very similar to E&M **U = qV**

For each strip (j), punish the loss function the further it is from the *brightest beta* for its particle (k)

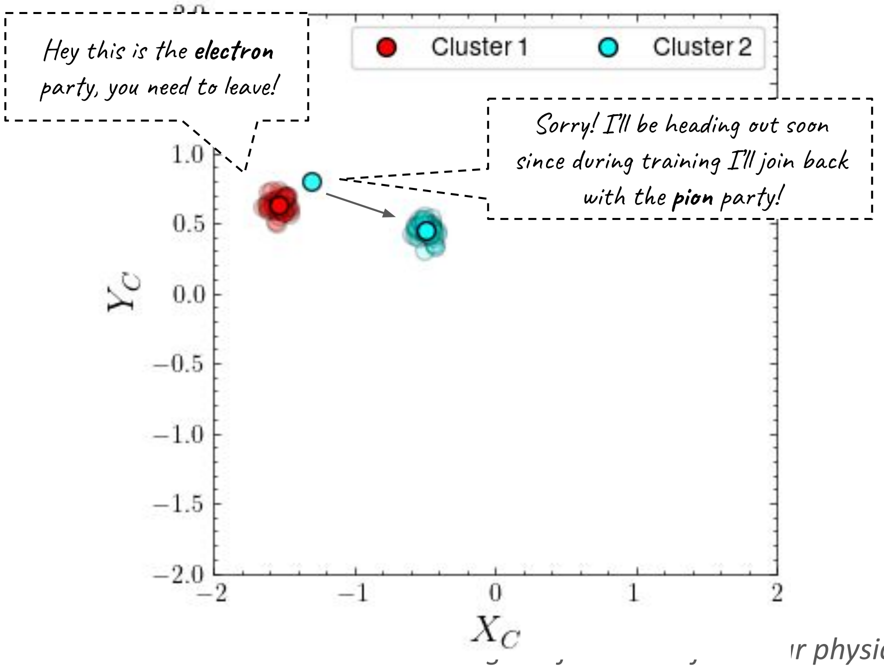
The *brightest* strip for particle (*k*) is αk

$$\breve{V}_k(x) = ||x - x_\alpha||^2 q_{\alpha k}$$



Object Condensation Loss

$$L_V = \frac{1}{N}\sum_{j=1}^{N} q_j \sum_{k=1}^{K}\left(M_{jk}\breve{V}_k(x_j) + (1 - M_{jk})\hat{V}_k(x_j)\right).$$



Repulsive Loss

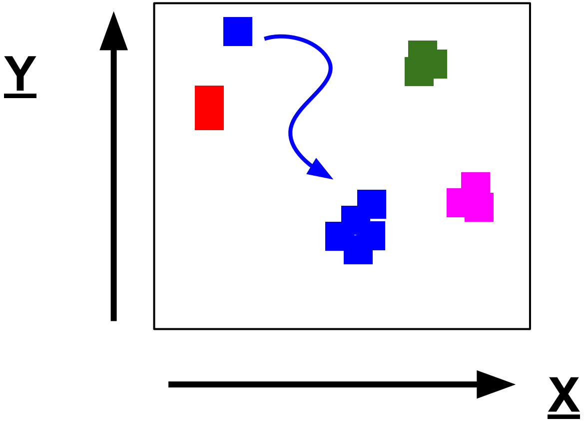
Each individual strip calculates **K-1** pieces of the **repulsive loss**

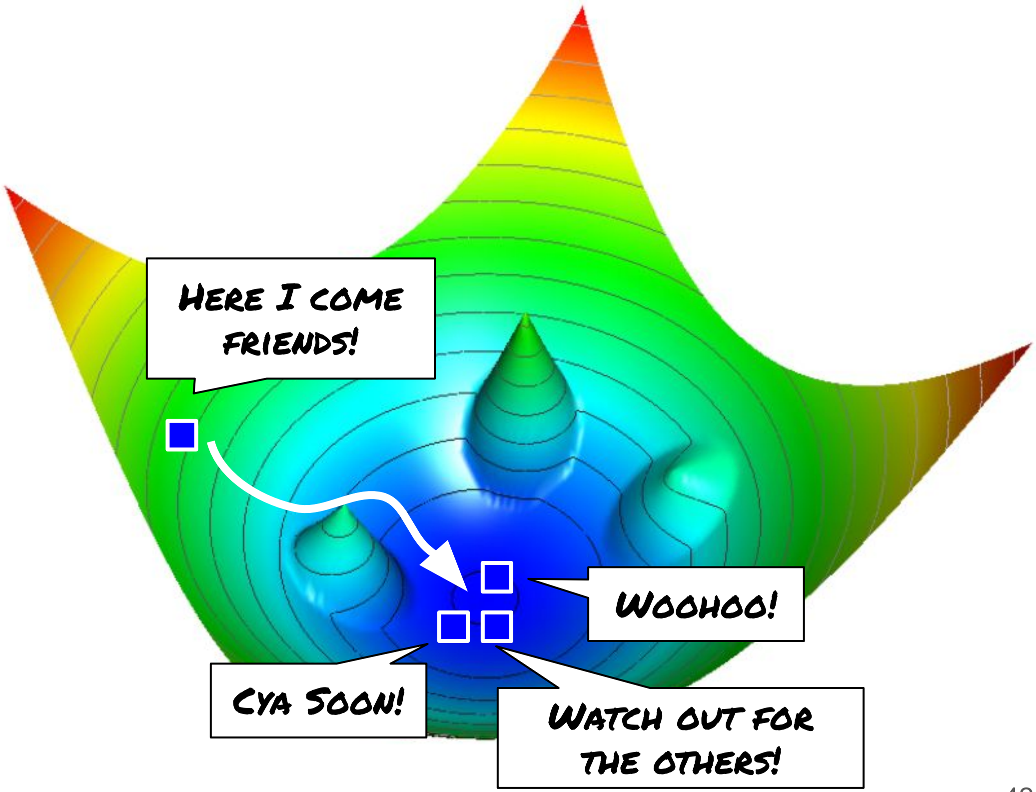
$$\hat{V}_k(x) = \max(0, 1 - ||x - x_\alpha||)q_{\alpha k}.$$

For each strip (j), punish the loss function the closer it is to the *brightest beta* of <u>any other particle</u> (k)

The *brightest* strip for particle (*k*) is αk

Object Condensation Attractive & Repulsive





(Right) The total potential V experienced by the blue square as it navigates past 3 unaffiliated objects (peaked condensation points) towards its clustering home (the bottom of the well, another condensation point)

Object Condensation Loss

$$L_\beta = \frac{1}{K}\sum_k (1-\beta_{\alpha k}) + s_B \frac{1}{N_B}\sum_i^N n_i \beta_i,$$

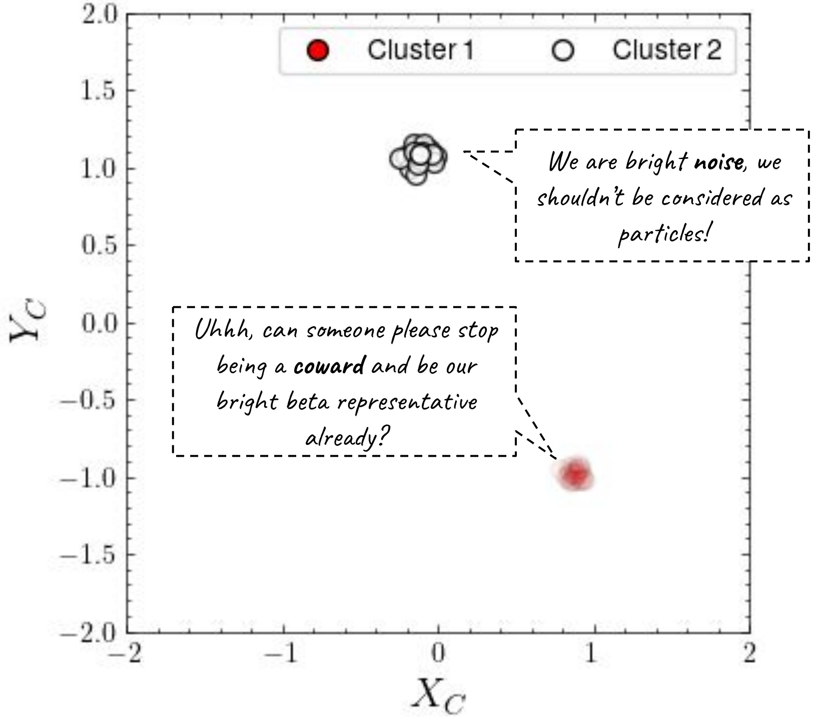
Coward Loss

For each particle (k) , punish the **coward loss** if the object's *brightest beta* is dim (near 0)

Noise Loss

For each strip (i) punish the **noise loss** if the strip is noise (ex: 0-padded) and has a high *brightness beta*

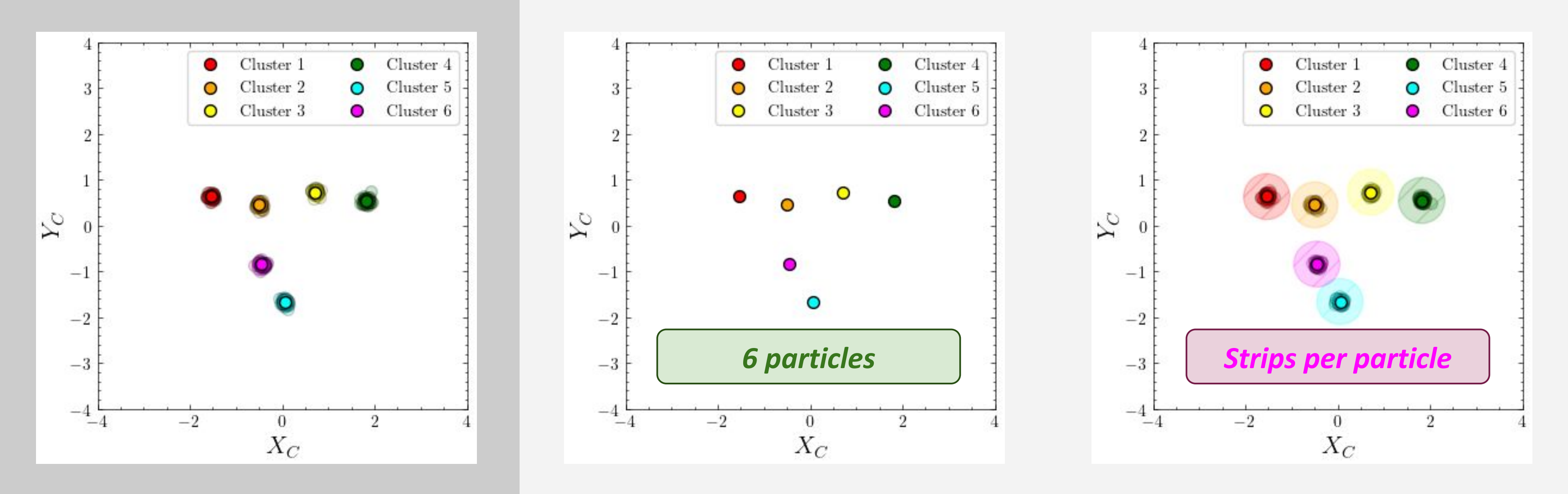
Here n_i is a bit that is 1 if strip (i) is noise



What is ★Object Condensation★?

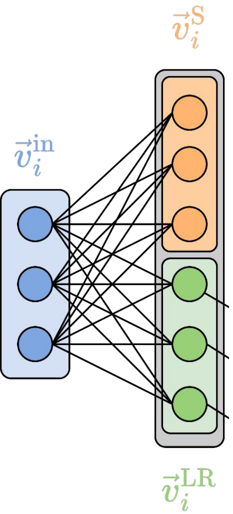
By viewing this clustered latent space ($\mathbf{x_c}$, $\mathbf{y_c}$) we can get…

- ➢ The number of particles – threshold away the dim □'s and count them!
- ➢ The strips for each particle – for a bright □, collect all dim □'s within some radius



GravNet Layer

$\mathbf{v}_i^{\mathbf{in}}$ → Strip i's Input vector to GravNet



Hyperparameters

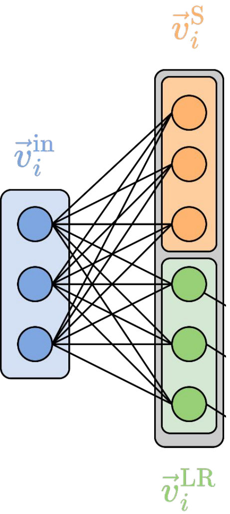
 # S-dims, **# Learned Features**

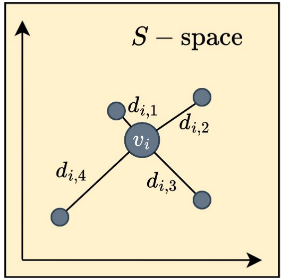
Procedure (for each strip)

1. A DNN produces a set of coordinates in **S-space** and hidden features $\mathbf{v}^{\mathbf{LR}}$

E

GravNet Layer





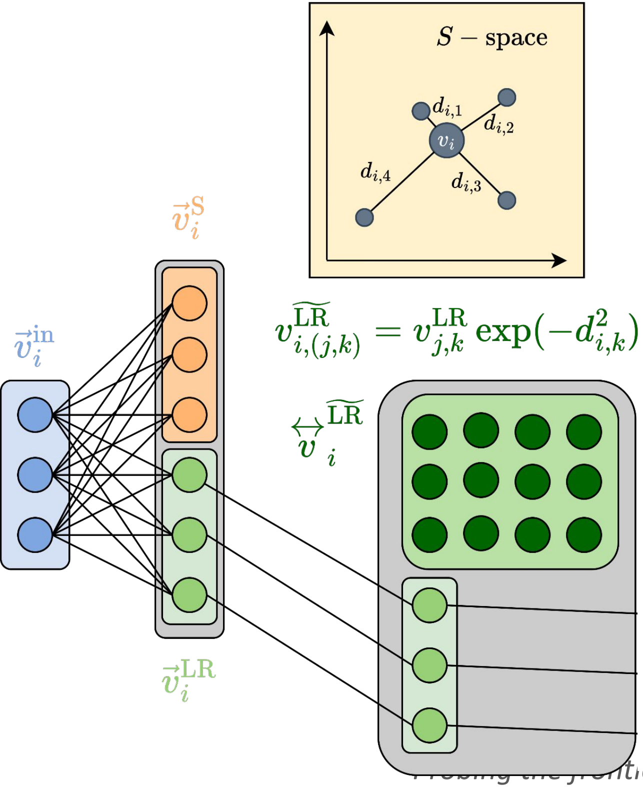
Hyperparameters

 # S-dims, **# Learned Features**, **# S-Neighbors**

Procedure (for each strip)

1. A DNN produces a set of coordinates in S-space and hidden features $\mathbf{v}^{LR}$
2. Calculate the distance $\mathbf{d}_{i,k}$ for **K** neighbors

GravNet Layer



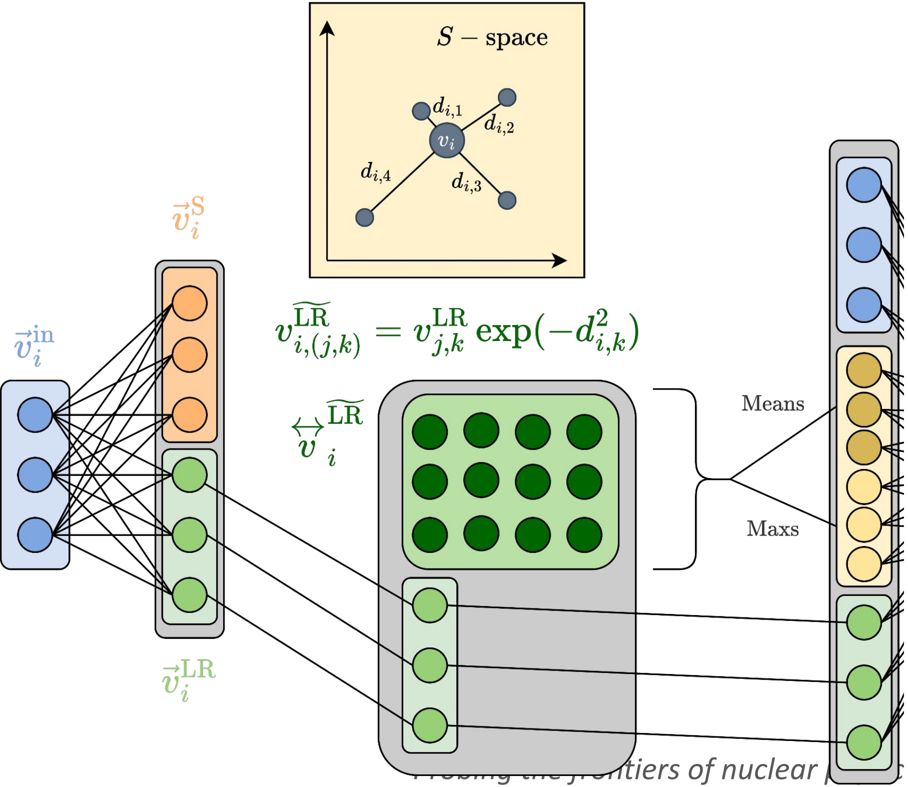
Hyperparameters

 # S-dims, **# Learned Features**, **# S-Neighbors**

Procedure (for each strip)

1. A DNN produces a set of coordinates in S-space and hidden features $\mathbf{v}^{\mathbf{LR}}$
2. Calculate the distance $\mathbf{d}_{\mathbf{i,k}}$ for **K** neighbors
3. Calculate distance-weighted *j-th* learned **(LR)** feature of the **K** neighbors of strip *i*

GravNet Layer



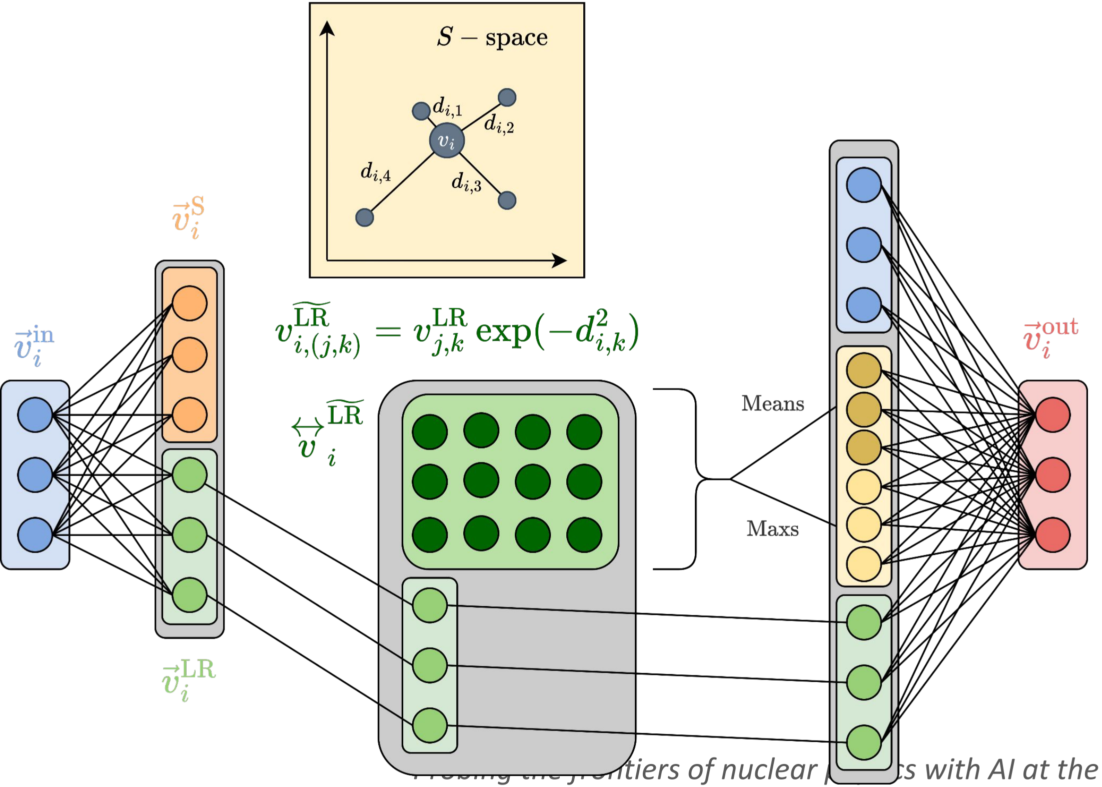
Hyperparameters

 # S-dims, **# Learned Features**, **# S-Neighbors**

Procedure (for each strip)

1. A DNN produces a set of coordinates in **S-space** and hidden features $\mathbf{v}^{LR}$
2. Calculate the distance $\mathbf{d}_{i,k}$ for **K** neighbors
3. Sum the distance-weighted *j-th* learned **(LR)** feature of the **K** neighbors of strip *i*
4. Calculate the **mean** & **max** of each learned features nearest neighbors. Concatenate v^{in} , v^{LR} and the mean(+)max of v\tilde{LR}

GravNet Layer



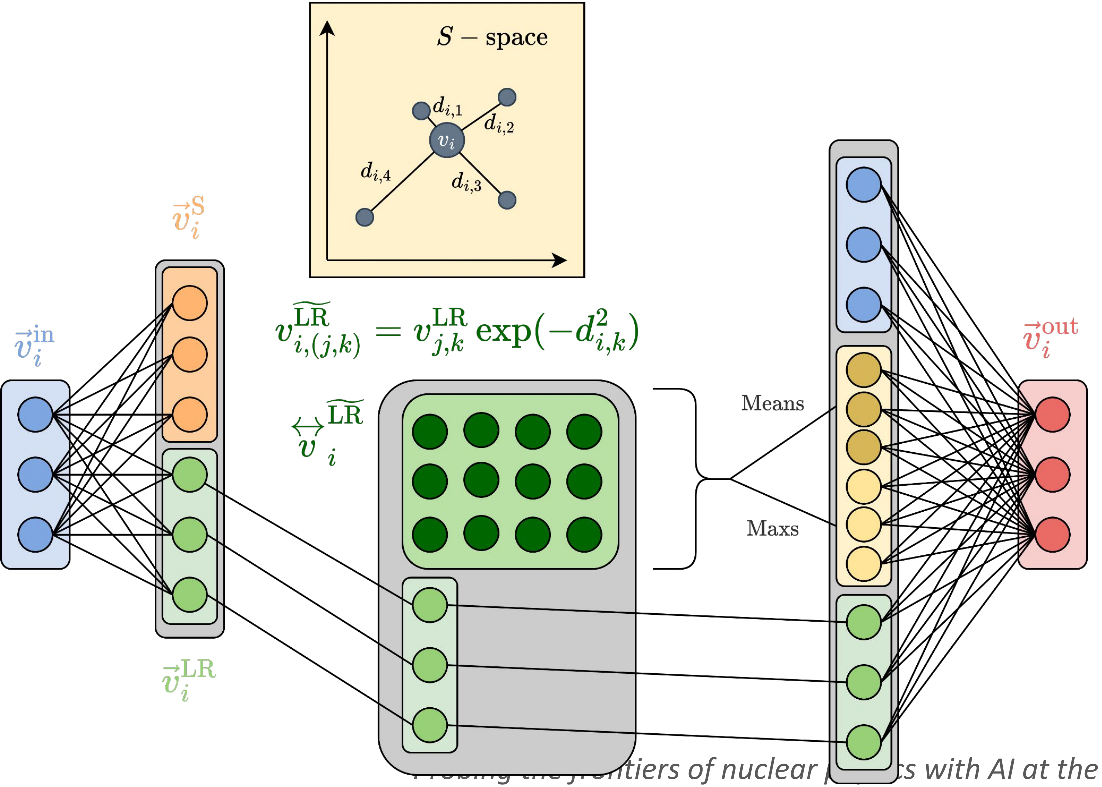
Hyperparameters

 # S-dims, **# Learned Features**, **# S-Neighbors**, **# output features**

Procedure (for each strip)

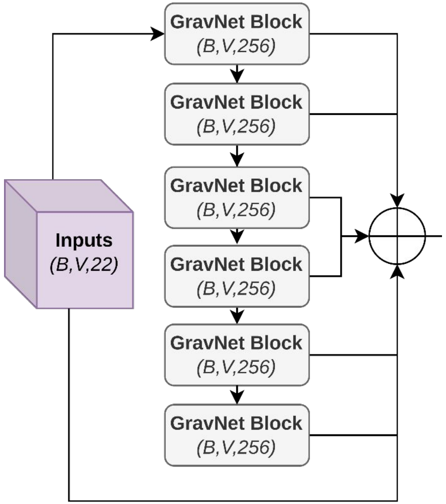
1. A DNN produces a set of coordinates in **S-space** and hidden features $\mathbf{v^{LR}}$
2. Calculate the distance $\mathbf{d_{i,k}}$ for **K** neighbors
3. Sum the distance-weighted *j-th* learned **(LR)** feature of the **K** neighbors of strip *i*
4. Calculate the **mean** & **max** of each learned features nearest neighbors. Concatenate v^{in} , v^{LR} and the mean(+)max of v^{\tilde{LR}}
5. DNN the final result to a new output vector $\mathbf{v^{out}}$

GravNet Layer

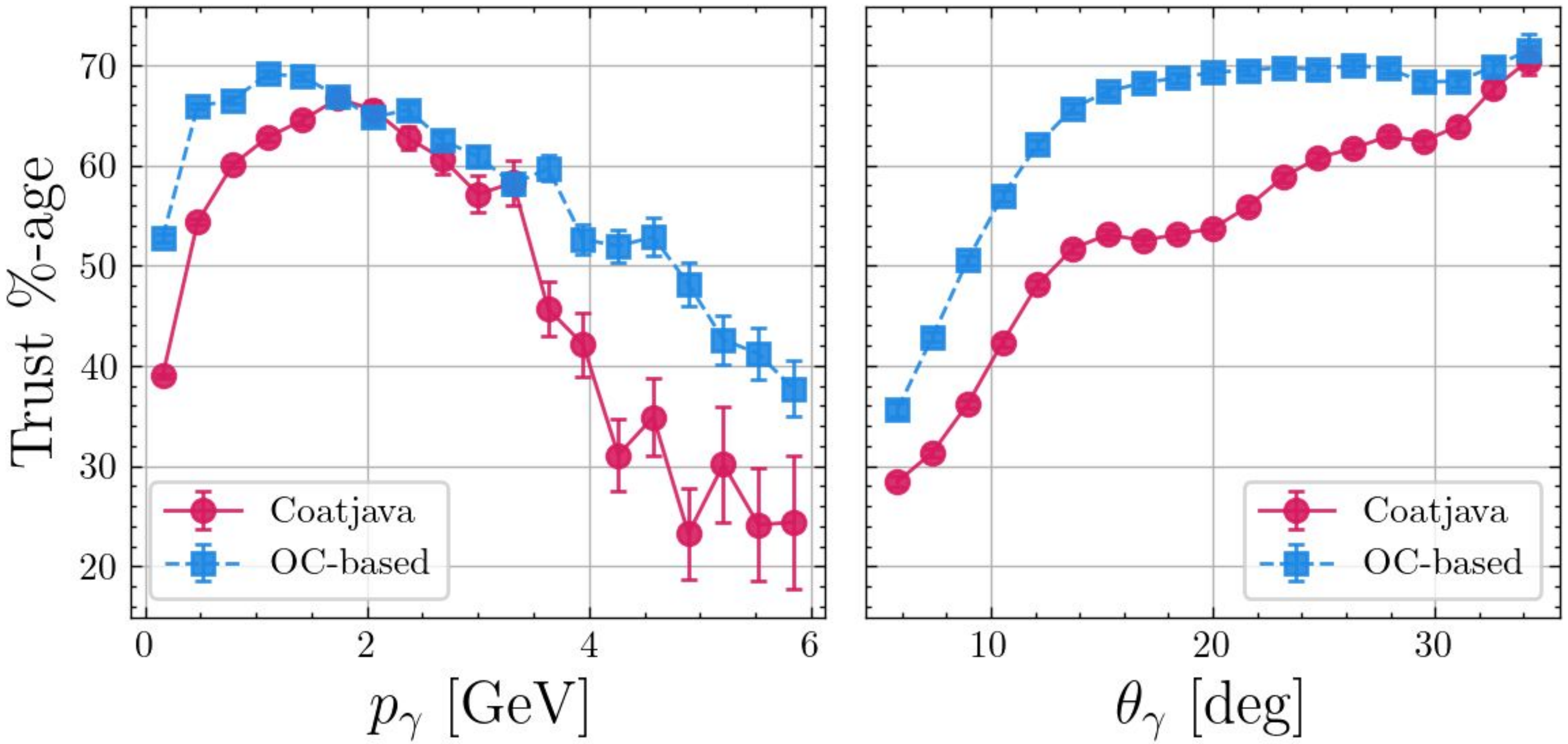


The 1st **v**$^{\mathbf{in}}$ to the GravNet blocks have <u>no neighbor info</u>

Each output of the block provides higher level nearest neighbor features



Trustworthiness of Photons



Network Architecture (Embedding Module)

Input: $x \in \mathbb{R}^{V \times F}$ *where V=150, F=17*

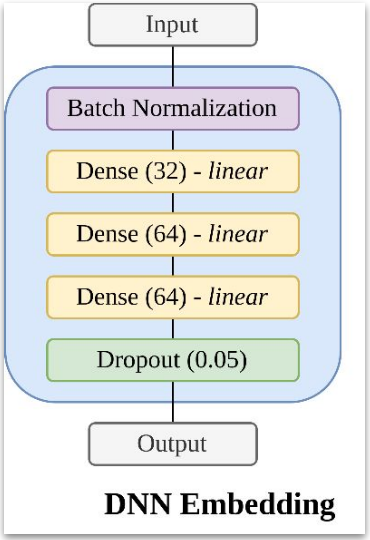
Output: $z \in \mathbb{R}^{V \times F'}$ *where F'=64*

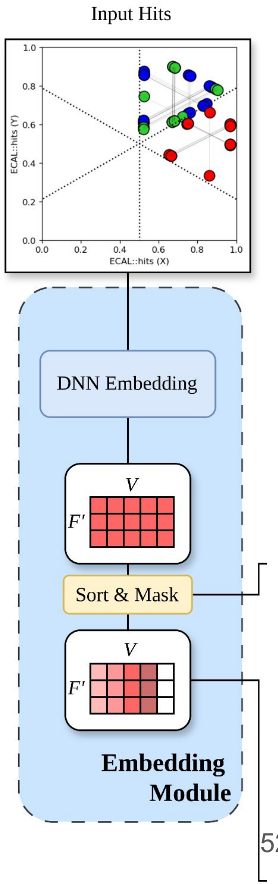
Main Idea

- Strip features fed through fully-connected dense layers
- Dropout applied during training

At CLAS12, there are 2448 unique ECal strips

- We sort along the **V** dimension based on the strip ID
- This is important for the setting up the encoder later





Network Architecture (Positional Encoding Module)

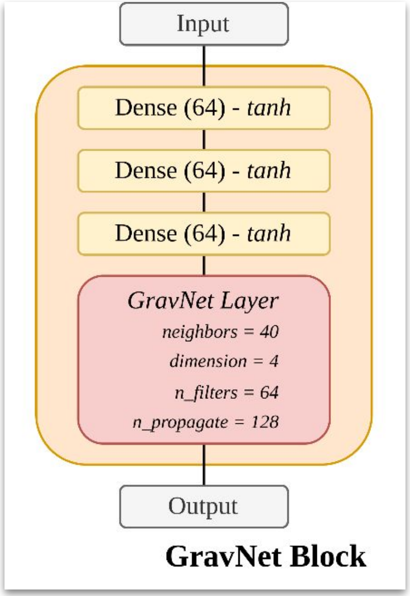
Input: $g \in \mathbb{R}^{H \times 6}$ *where H=2448 and the "6" features are the (x,y,z) of the strip's 2 endpoints*

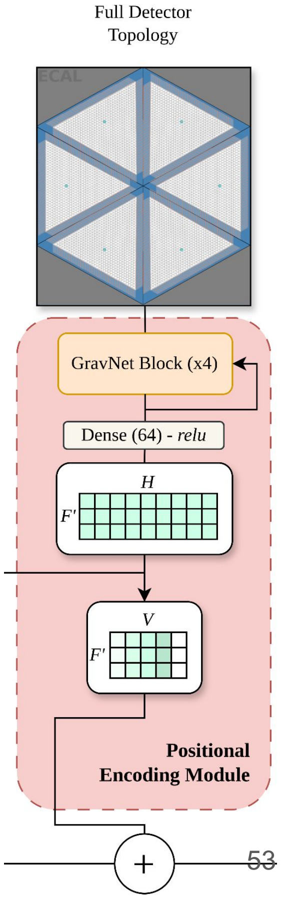
Output: $g' \in \mathbb{R}^{V \times F'}$ *where F'=64*

Main Idea

- Full, fixed ECal detector topology passed through GravNet layers (dynamically built k-NNs)
- Each strip at CLAS12 *learns* a unique representation based on its geometry

Using the Embedding Module sorting, we gather the relevant strips from the Positional Encoding Module





Network Architecture (Feature Extraction Module)

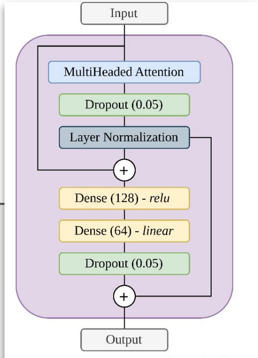
Input: $z + g' \in \mathbb{R}^{V \times F'}$ *where **z** and **g'** were outputs of embedding module and positional encoding (PE) module*

Output: $y \in \mathbb{R}^{V \times 3}$

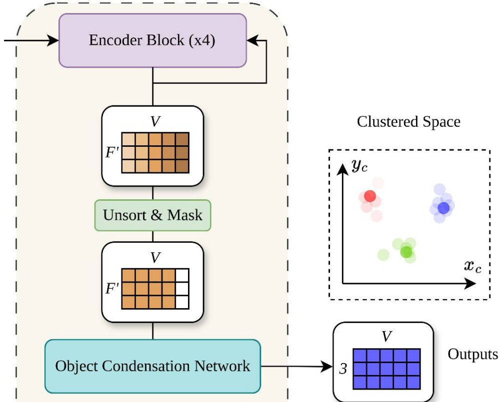
- ➢ Main structure follows a transformer encoder with self-attention + feed forward layers
- ➢ Individual ECal hits are treated like "tokens" in a sentence, positions learned by PE module
- ➢ Encoder allows long-range contextual information to build up during training

Sorting along **V** dimension is undone and sent through final network

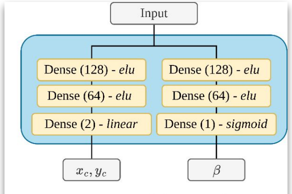
- ➢ Each hit learns a location ($\mathbf{x_c}$, $\mathbf{y_c}$) and brightness ($\boldsymbol{\beta}$)



Encoder Block



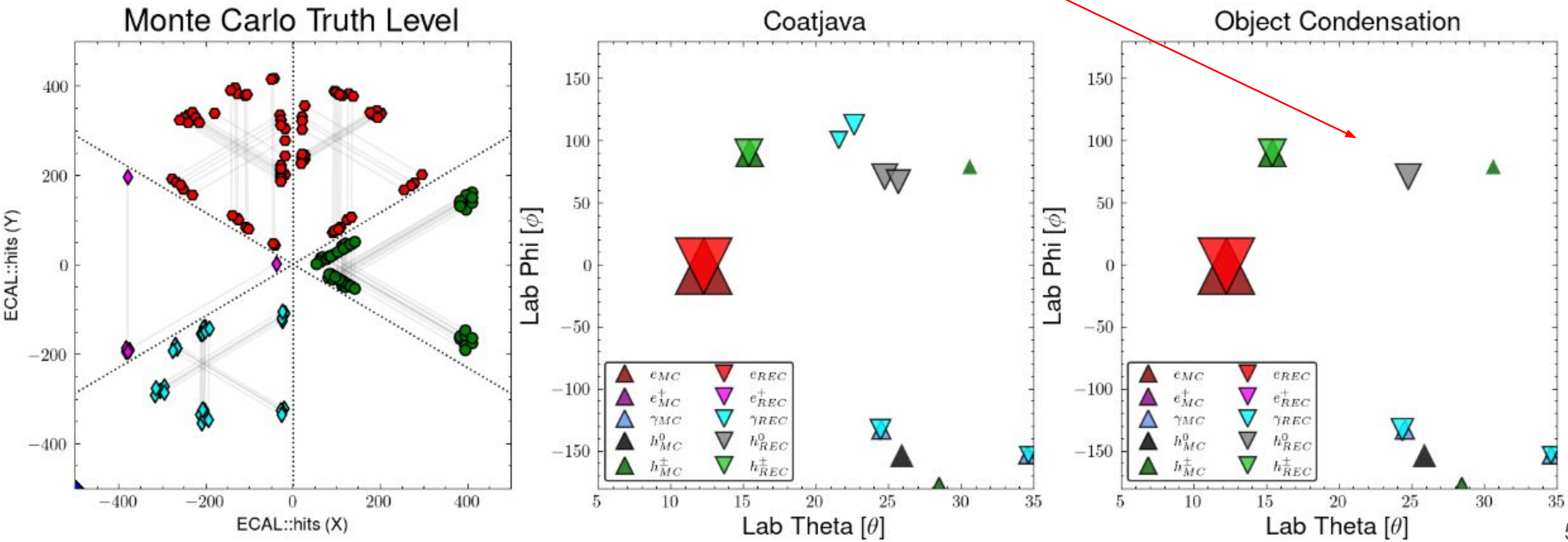
Feature Extraction Module



Object Condensation Network

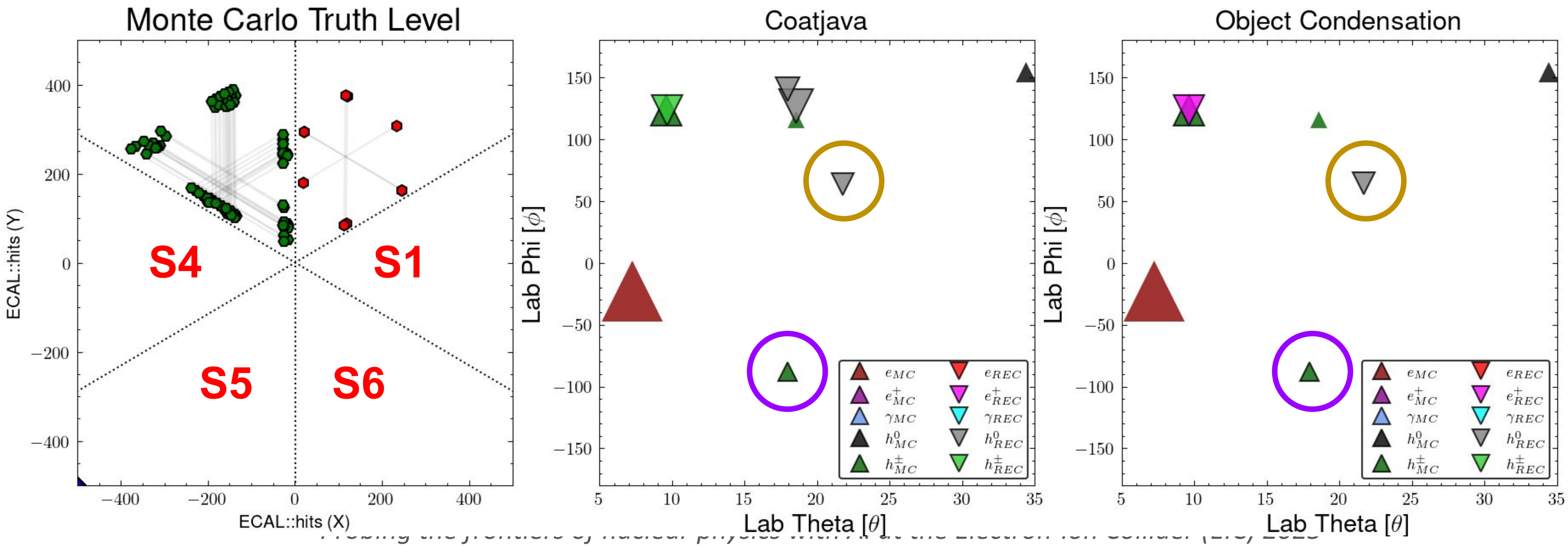
Multi-sector hits

- Coatjava and Object Condensation will be prone to scenarios where accidental neutral clustering is *unavoidable*
  - Ex: Below, a **Pi+** left hits in S2 and S3 (φ ~ 60 - 100 [deg])
  - Both Coatjava and Object Condensation find a stray neutral

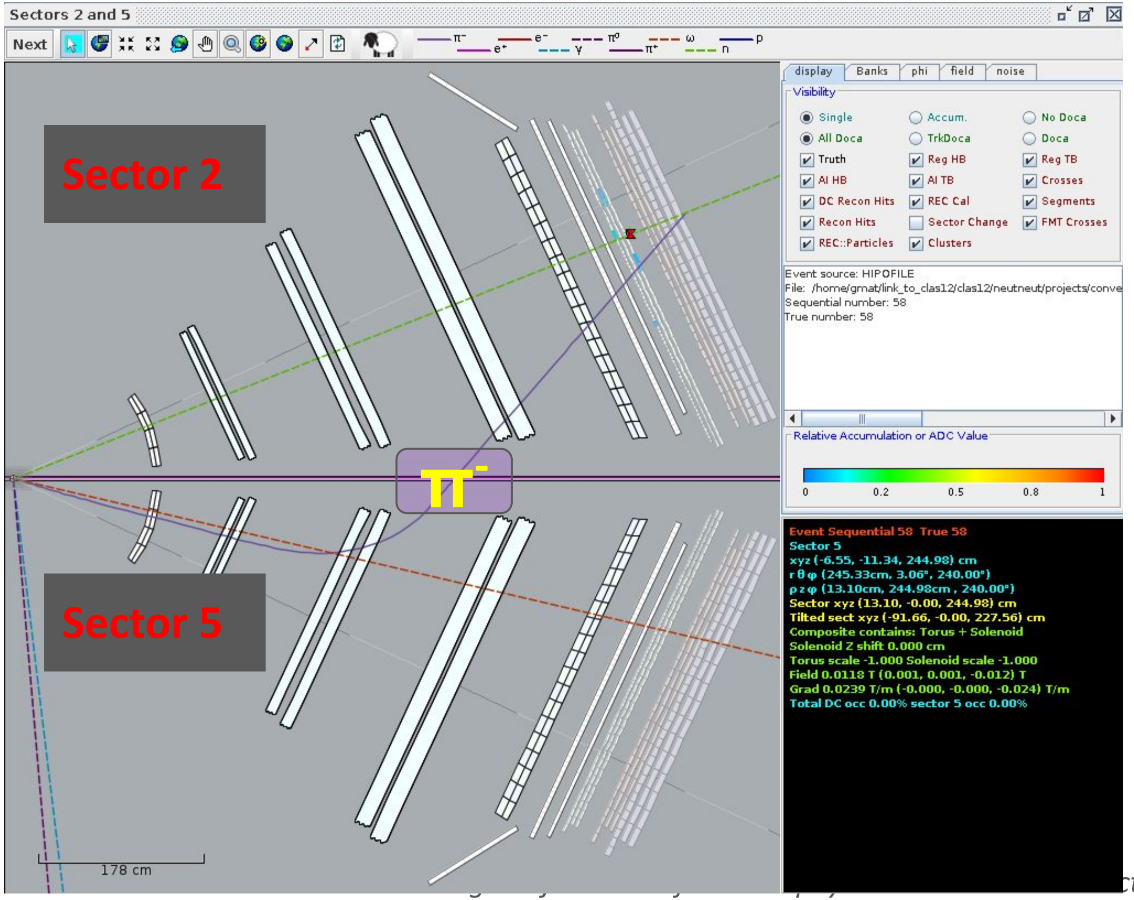


Intersector Tracks

- In this other example, a **Pi- Generated in Sector 5** crosses into sector 2. This pion leaves hits in **Sector 2 which is registered as a Neutron**
  - It *makes sense* that Object Condensation would see the **Sector 2** 3-way intersection as a viable cluster

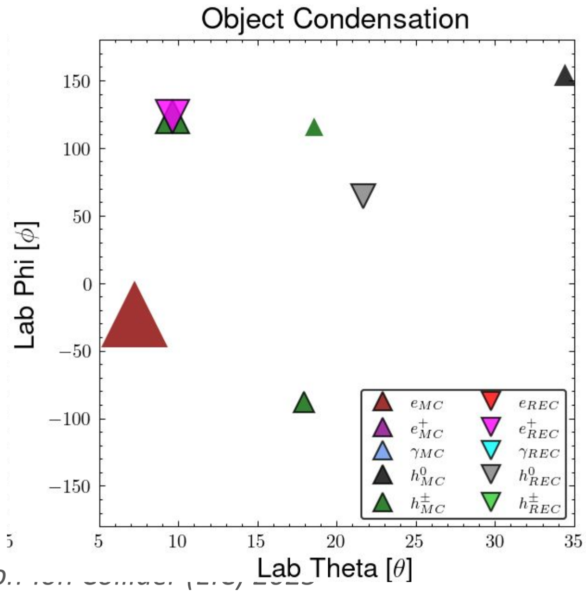


Intersector Tracks



Track might actually leave hits in all 3 DC's (albeit different sectors)

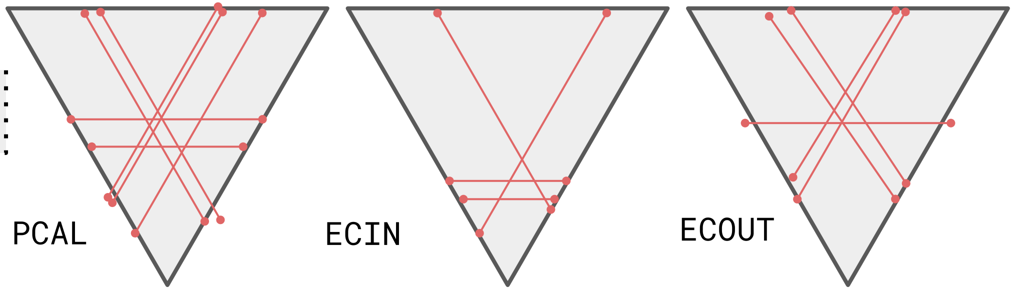
Does the track algorithm account for this in anyway?



Relevant COATJAVA EventBuilder Pipeline

★ **Issues** in this step lead to faulty clustering of excess neutral particles

Thrown pion →



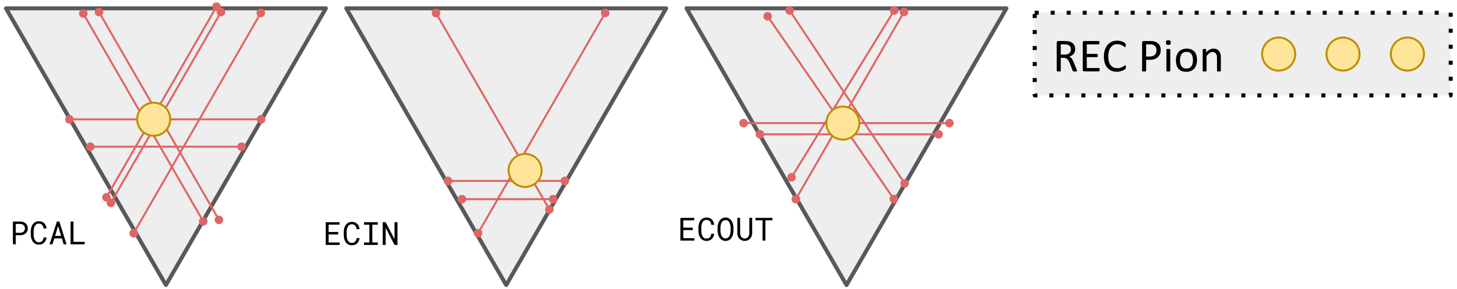
ECAL::hits → ECAL::peaks → ECAL::clusters → REC::Calorimeter → REC::Particle

- **ECAL::hits**: Strip-by-strip info
- **ECAL::peaks**: Collects adjacent strips into “peak” objects
- **ECAL::clusters**: Finds 3-way crossings to form clusters
- **REC::Calorimeter**: Matches clusters in PCAL, ECIN, ECOUT to individual tracks/neutrals
- **REC::Particle**: List of particles

Relevant COATJAVA EventBuilder Pipeline

★ **Issues** in this step lead to faulty clustering of excess neutral particles

… ***Coatjava*** may find 3 **clusters** in and correctly associate them with one another…



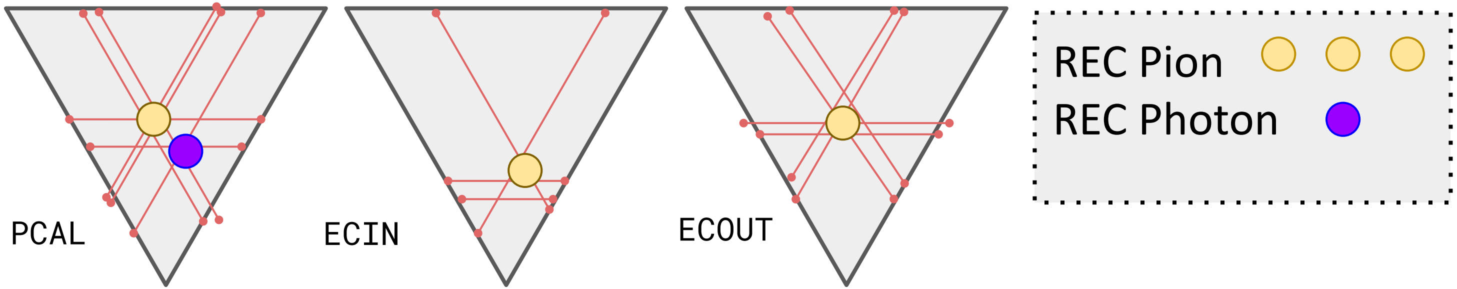
ECAL::hits → ECAL::peaks → ECAL::clusters → REC::Calorimeter → REC::Particle

- **ECAL::hits**: Strip-by-strip info
- **ECAL::peaks**: Collects adjacent strips into "peak" objects
- **ECAL::clusters**: Finds 3-way crossings to form clusters
- **REC::Calorimeter**: Matches clusters in PCAL, ECIN, ECOUT to individual tracks/neutrals
- **REC::Particle**: List of particles

Relevant COATJAVA EventBuilder Pipeline

★ **Issues** in this step lead to faulty clustering of excess neutral particles

… ***Coatjava*** may find 3 **clusters** in and correctly associate them with one another… but it may accidentally **find more**!



ECAL::hits → ECAL::peaks → ECAL::clusters → REC::Calorimeter → REC::Particle

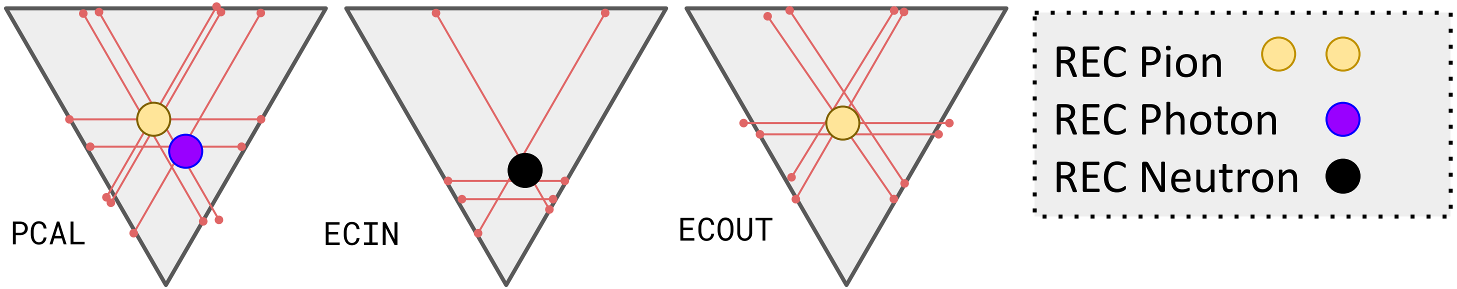
ECAL::hits	ECAL::peaks	ECAL::clusters	REC::Calorimeter	REC::Particle
Strip-by-strip info	Collects adjacent strips into "peak" objects	Finds 3-way crossings to form clusters	Matches clusters in PCAL, ECIN, ECOUT to individual tracks/neutrals	List of particles

Relevant COATJAVA EventBuilder Pipeline

★ **Issues** in this step lead to faulty clustering of excess neutral particles

… ***Coatjava*** may find 3 **clusters** in and correctly associate them with one another… but it may accidentally **find more**!

… The clusters may also **fail to be associated**!



ECAL::hits → ECAL::peaks → ECAL::clusters → REC::Calorimeter → REC::Particle

ECAL::hits	ECAL::peaks	ECAL::clusters	REC::Calorimeter	REC::Particle
Strip-by-strip info	Collects adjacent strips into "peak" objects	Finds 3-way crossings to form clusters	Matches clusters in PCAL, ECIN, ECOUT to individual tracks/neutrals	List of particles