

AI-powered calorimeter clustering with coatjava integration



Gregory Matousek
11-12-2024



Presentation Outline

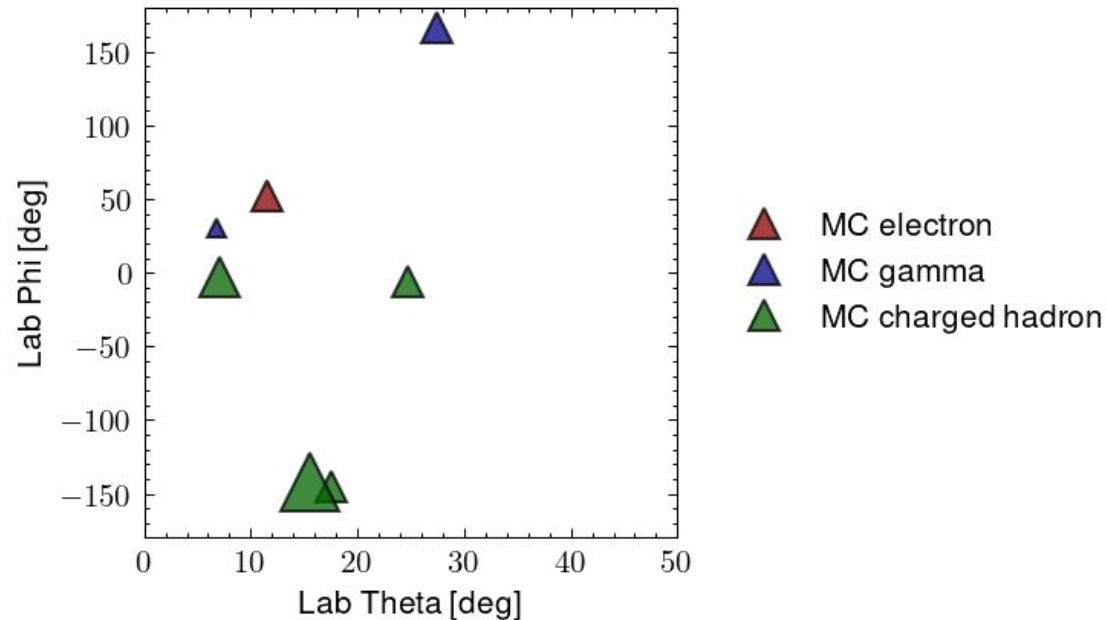
1. Why do we need to improve neutral clustering at CLAS12 (γ 's vs. n's)?
2. How does COATJAVA reconstruct clusters and what are its flaws?
3. Introduce **★Object Condensation★** – a *grid-free* machine learning approach to object clustering
4. GravNet nearest-neighbor model architecture – training parameters/features
5. Training metrics on Monte Carlo – How well does the model perform?
6. Custom COATJAVA pipeline for this project

Model Evaluation (COATJAVA vs. Object Condensation)

- A. Neutron Gun events
- B. Incoherent J/Psi production off deuterium (with the help of Richard Tyson)
- C. Monte Carlo DIS events

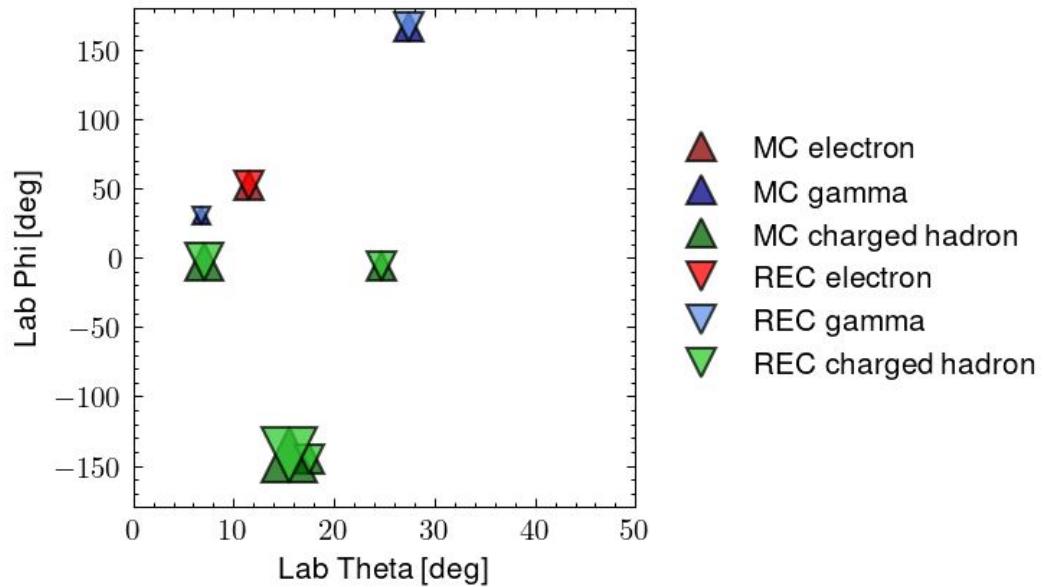
Neutral Clustering at CLAS12

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



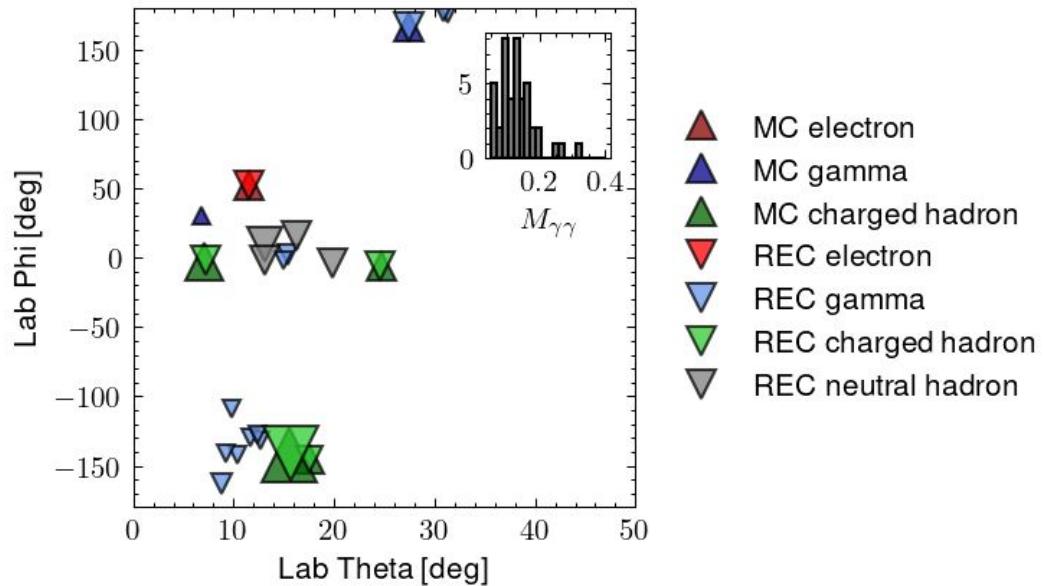
Neutral Clustering at CLAS12

- 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

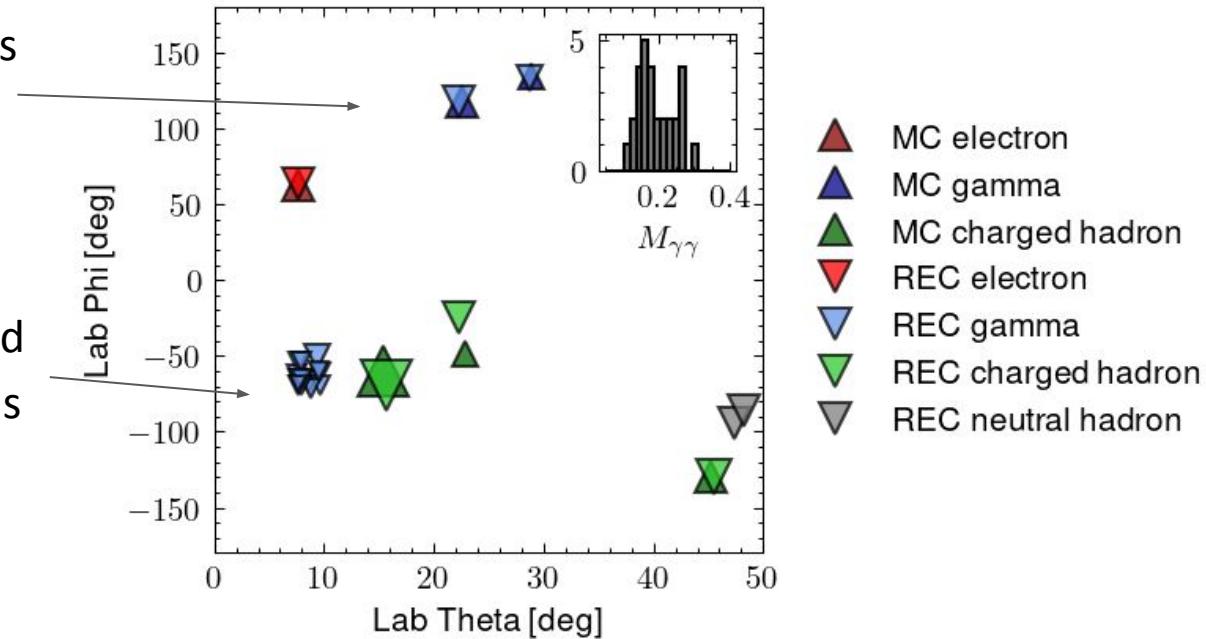


Non-combinatorial backgrounds emerge for π^0 studies for instance, where one of the photons in the pair is fake

Resolving the Photon Clustering Issue

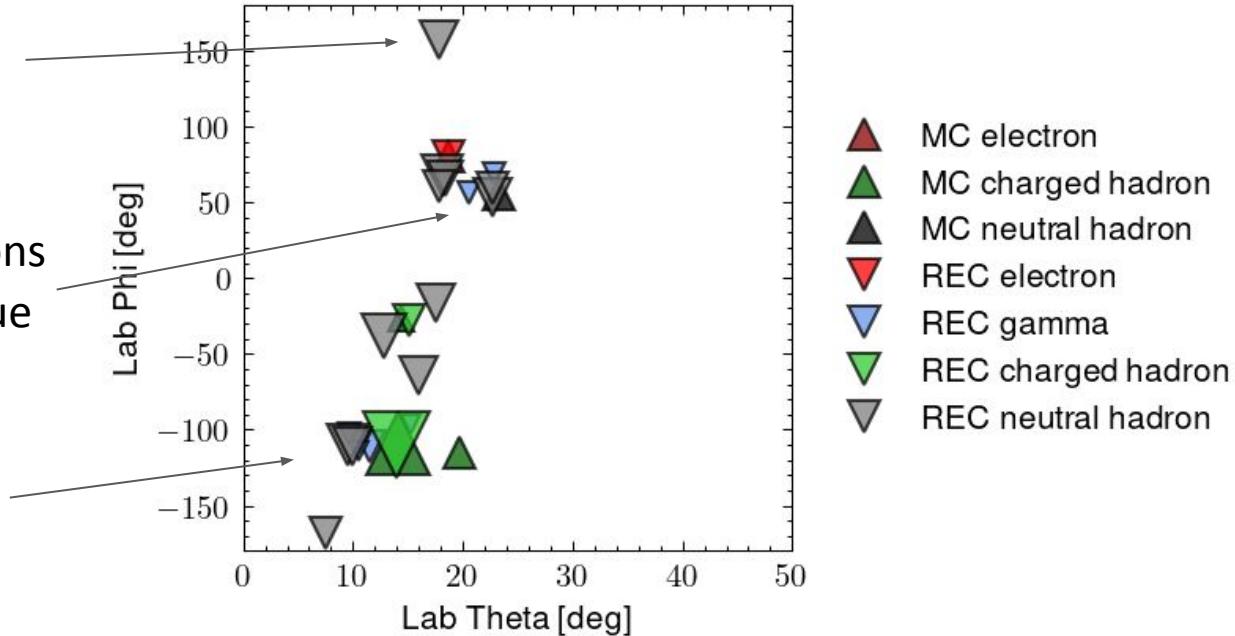
- It turns out the information in **REC::Calorimeter** and **REC::Particle** is plenty to address the false photon backgrounds
- Unlikely for false photons to collect around true photons
- More likely for false photons to collect around many other false photons

A simple **Gradient Boosted Tree** model with nearest neighbor features cleans up the photons at CLAS12

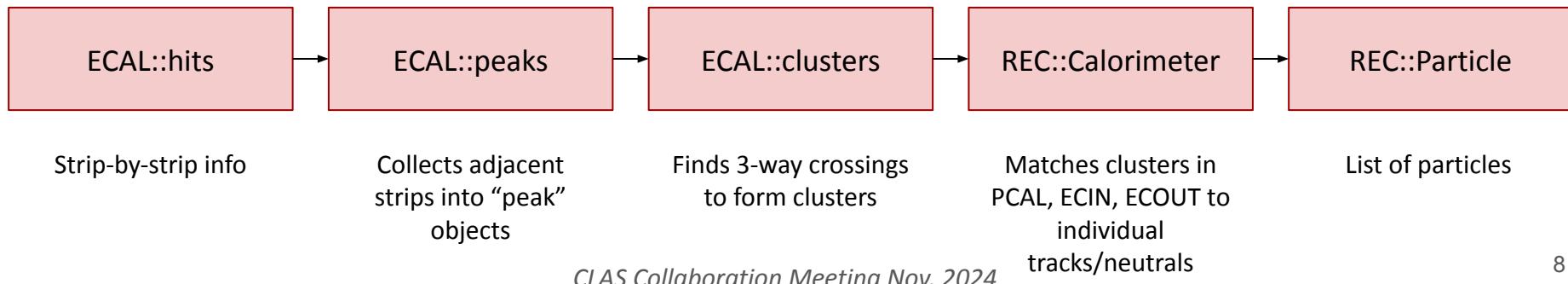


Why Neutrons pose a challenge

- It turns out the information in **REC::Calorimeter** and **REC::Particle** is **NOT ENOUGH** to address the **false neutron** backgrounds
- Neutrons can be fairly separated in (θ, ϕ) yet definitely false
- Bundles of false neutrons can be found near a true neutron
- They can also be found with no true neutron nearby

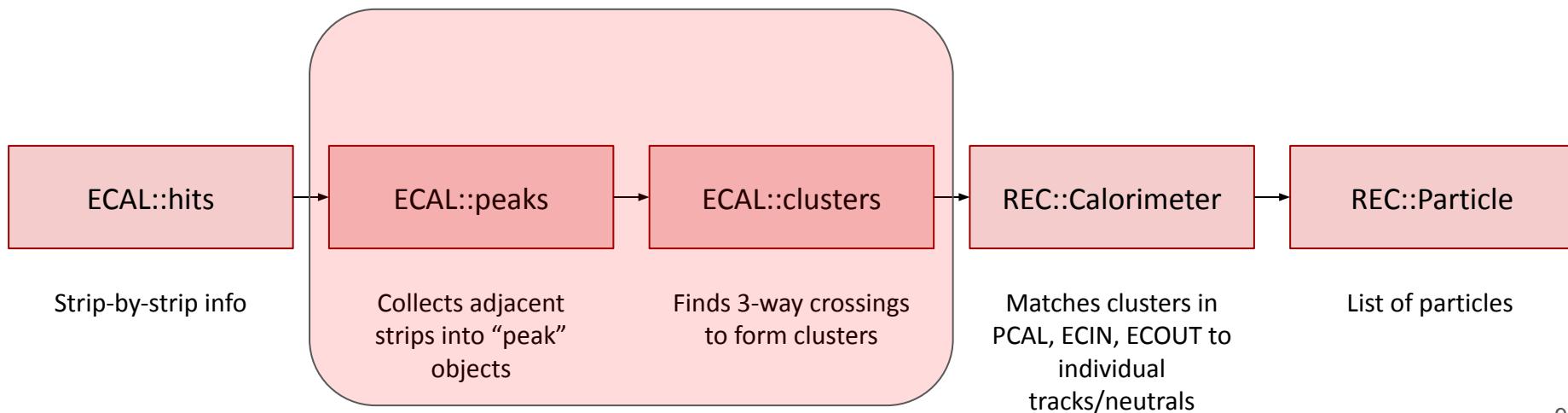


Relevant COATJAVA EventBuilder Pipeline



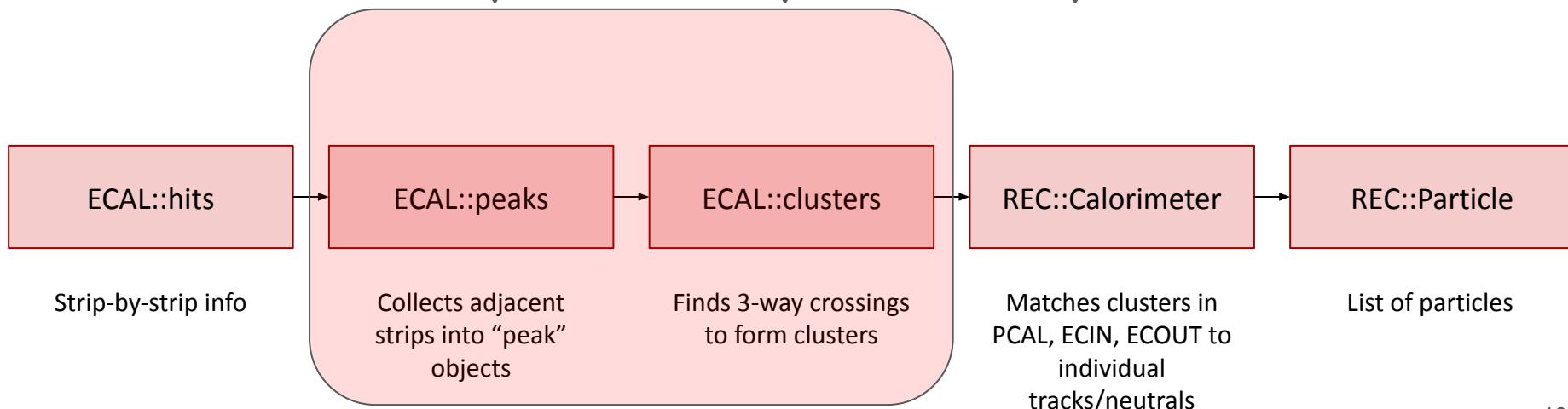
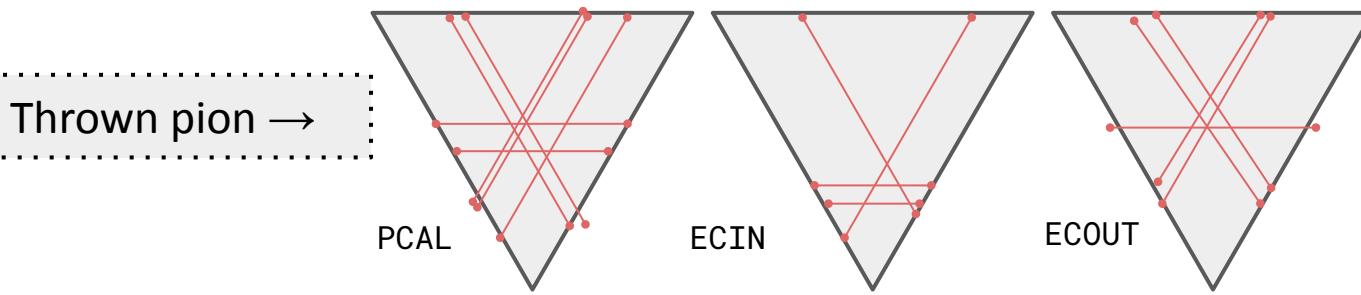
Relevant COATJAVA EventBuilder Pipeline

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



Relevant COATJAVA EventBuilder Pipeline

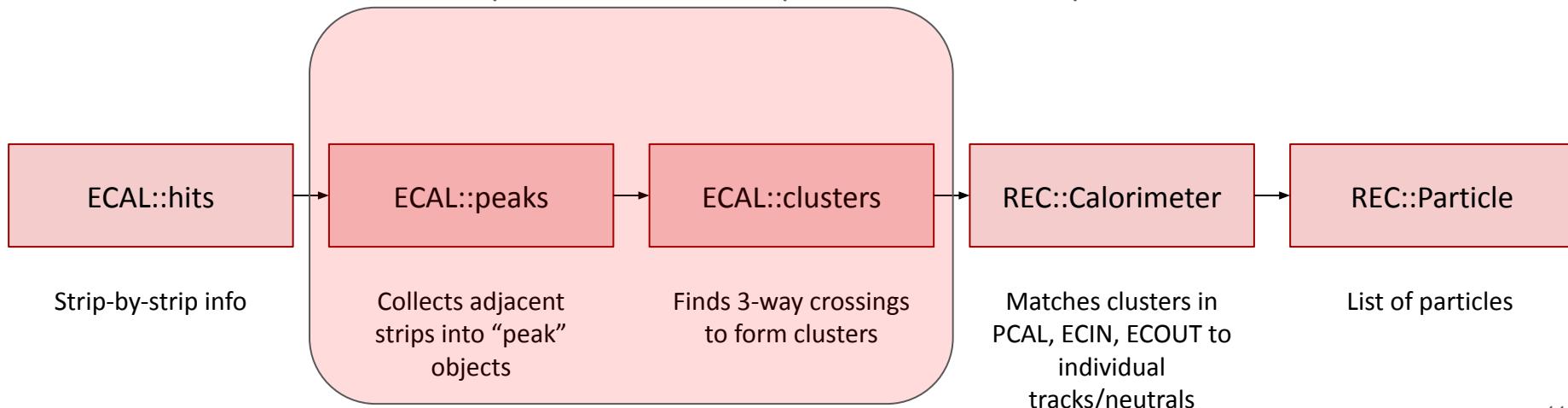
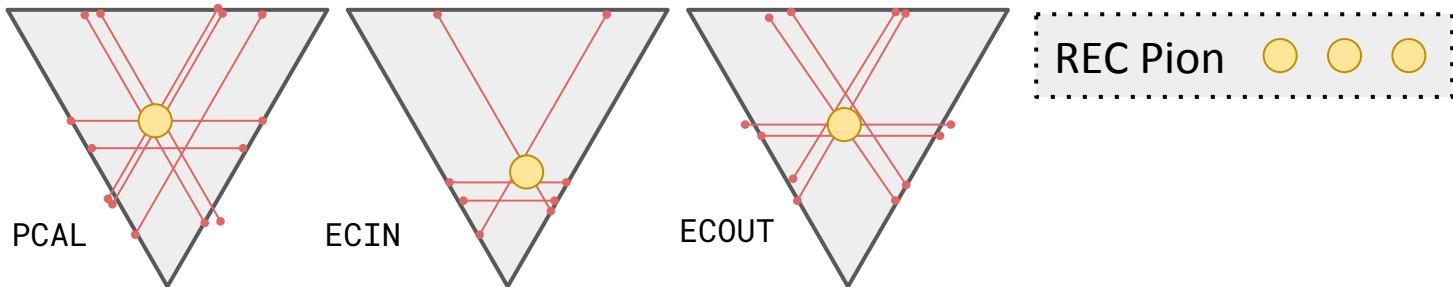
★ **Issues** in this step lead to faulty clustering of excess neutral particles



Relevant COATJAVA EventBuilder Pipeline

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

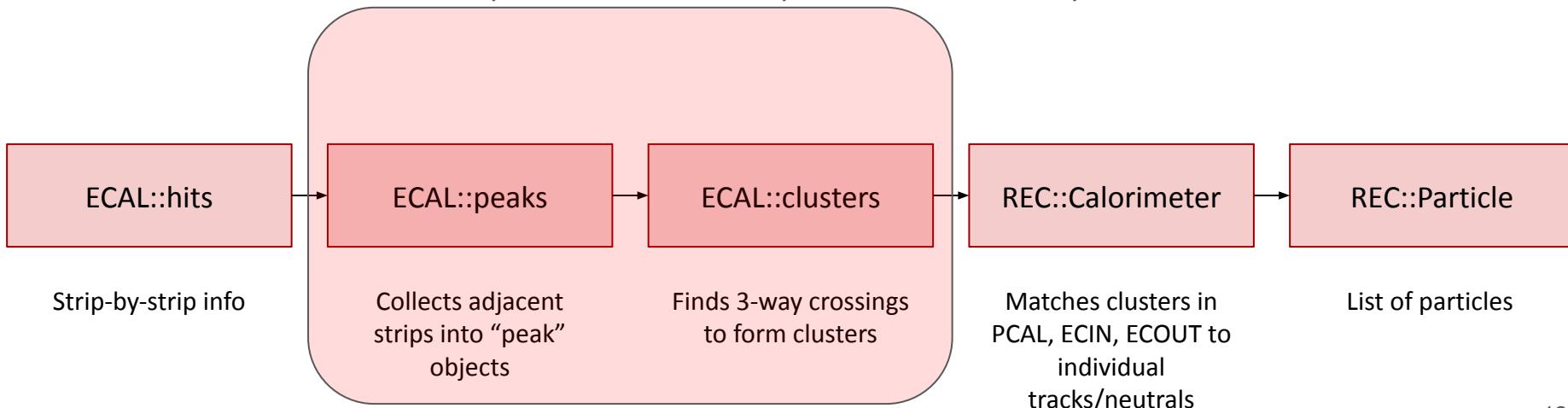
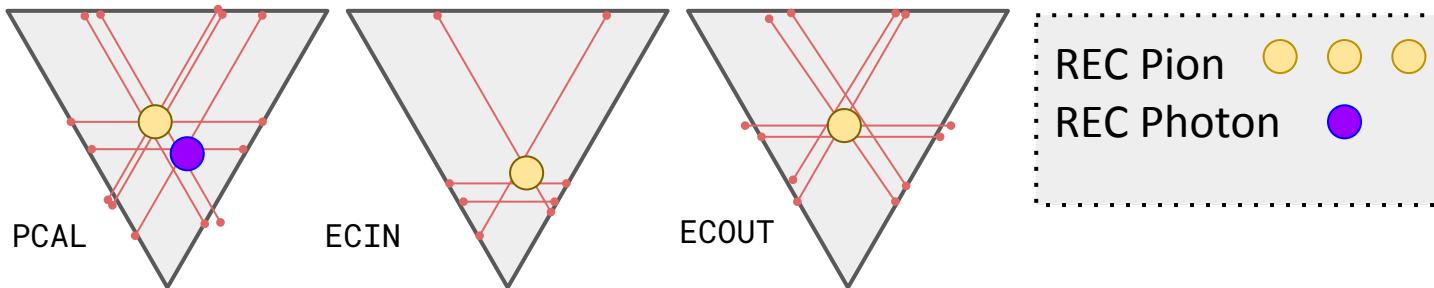
... *Coatjava* may find 3 clusters in and correctly associate them with one another...



Relevant COATJAVA EventBuilder Pipeline

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

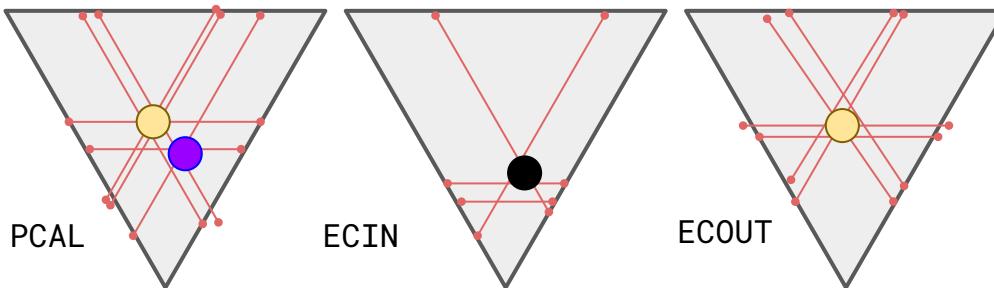
... *Coatjava* may find 3 clusters in and correctly associate them with one another... but it may accidentally **find more!**



Relevant COATJAVA EventBuilder Pipeline

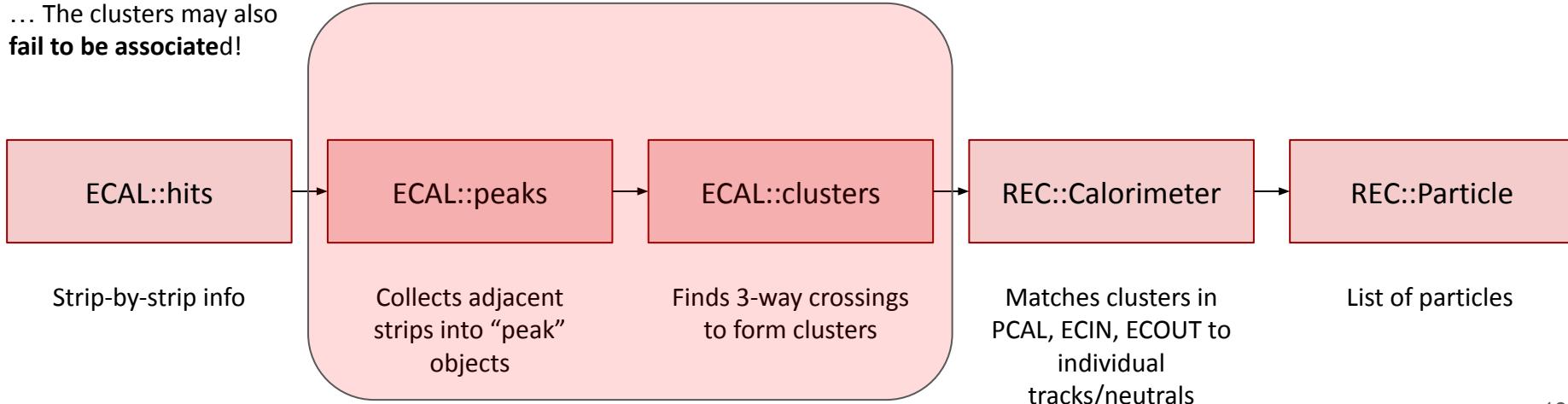
★ **Issues** in this step lead to faulty clustering of excess neutral particles

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



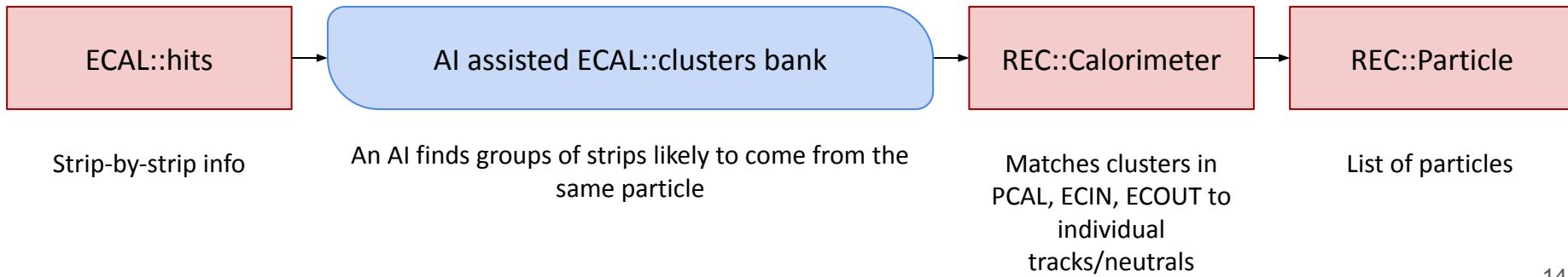
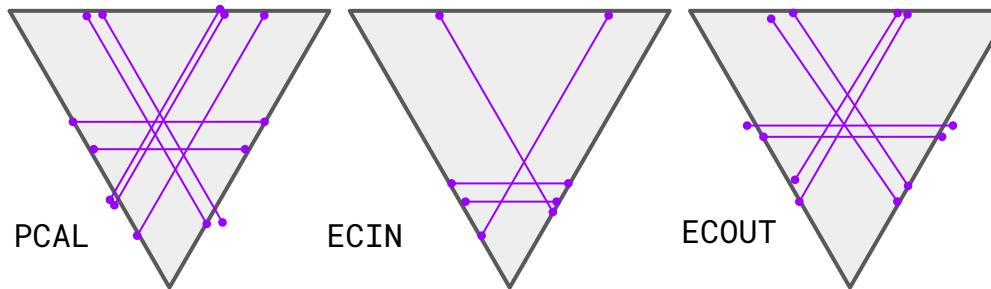
REC Pion ● ●
REC Photon ● ●
REC Neutron ● ●

... The clusters may also fail to be associated!



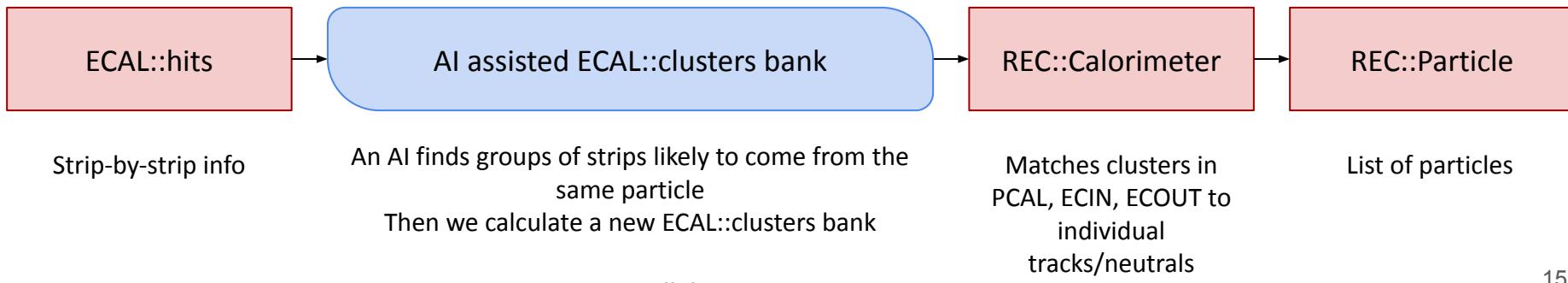
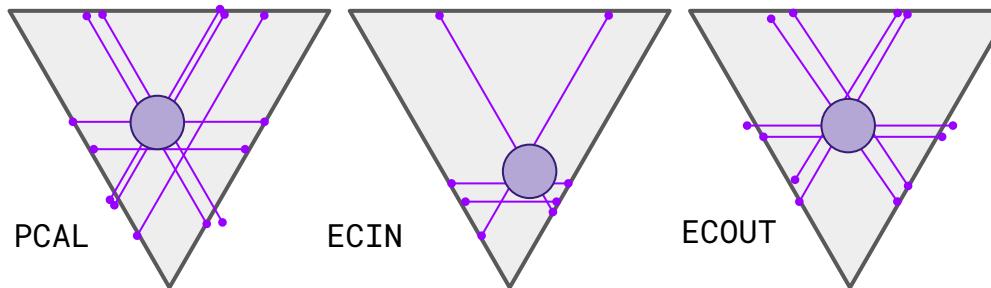
AI-assisted Neutral Clustering

- ★ Our AI organizes groups of strips separate **single objects** (particles)



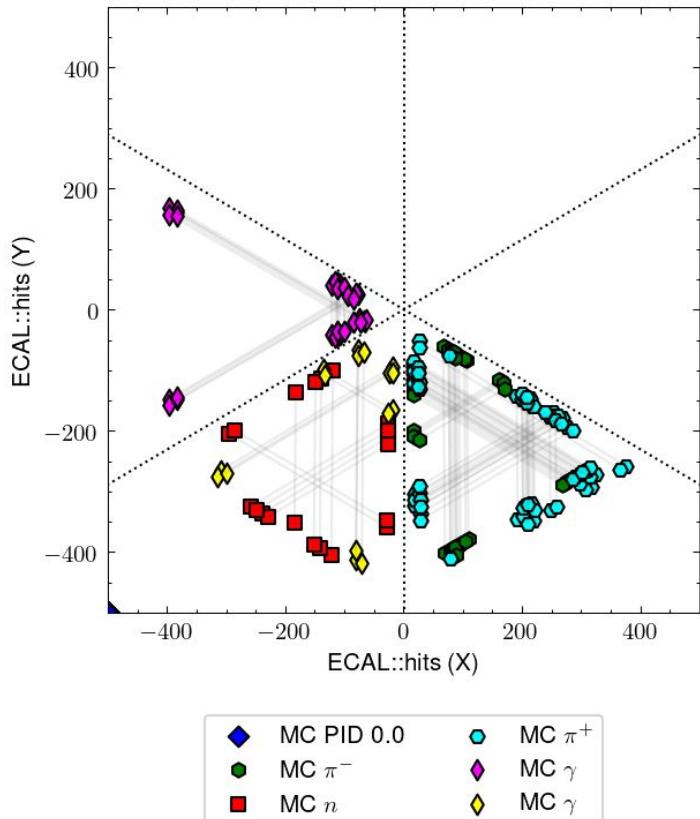
AI-assisted Neutral Clustering

- ★ Our AI organizes groups of strips separate **single objects** (particles)
- ★ Then we manually calculate one cluster (x,y,z,E,t) for each ECAL type



Sample Monte Carlo Event

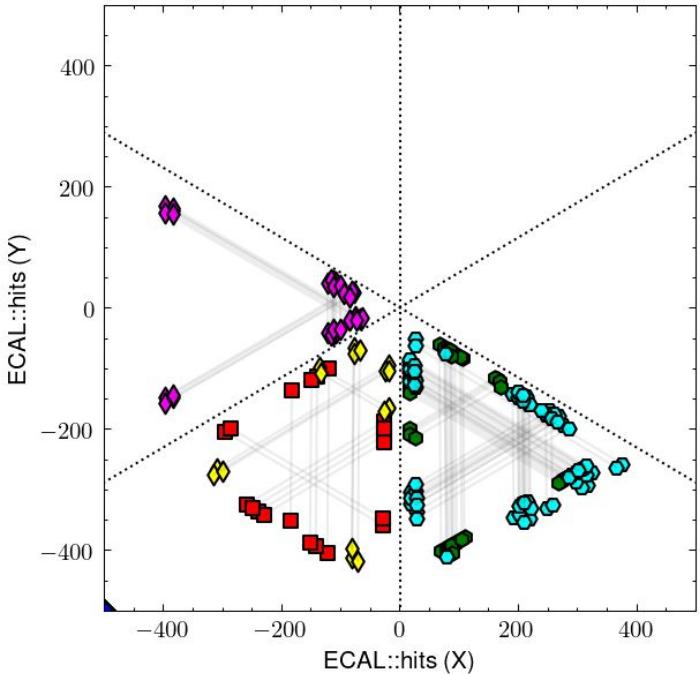
Monte Carlo Truth Level



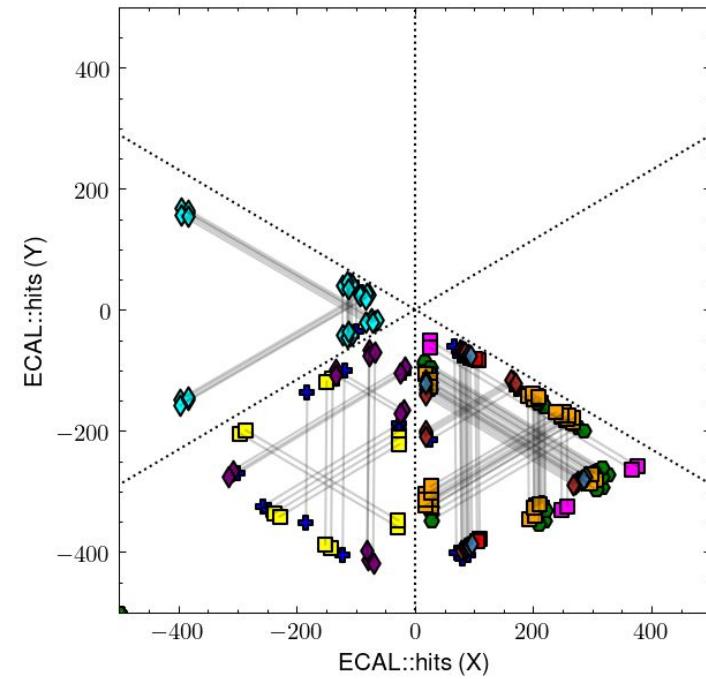
- Left Plot shows the final state Monte Carlo particles generated in SIDIS that are *responsible* for the ECAL strip hits
 - **Colors** → Different particles
 - **Shapes** → Different MC PIDs
- PCAL, ECIN, and ECOUT are overlaid
- For each strip hit, there is an “origin” and “endpoint” (x, y, z) as well as edep and timing
- In general, Coatjava looks for **3-way intersections** in the PCAL, ECIN, and ECOUT (separately) to create *clusters*
- Track \leftrightarrow Cluster matching determines if we need to make a neutral particle

Sample Monte Carlo Event

Monte Carlo Truth Level



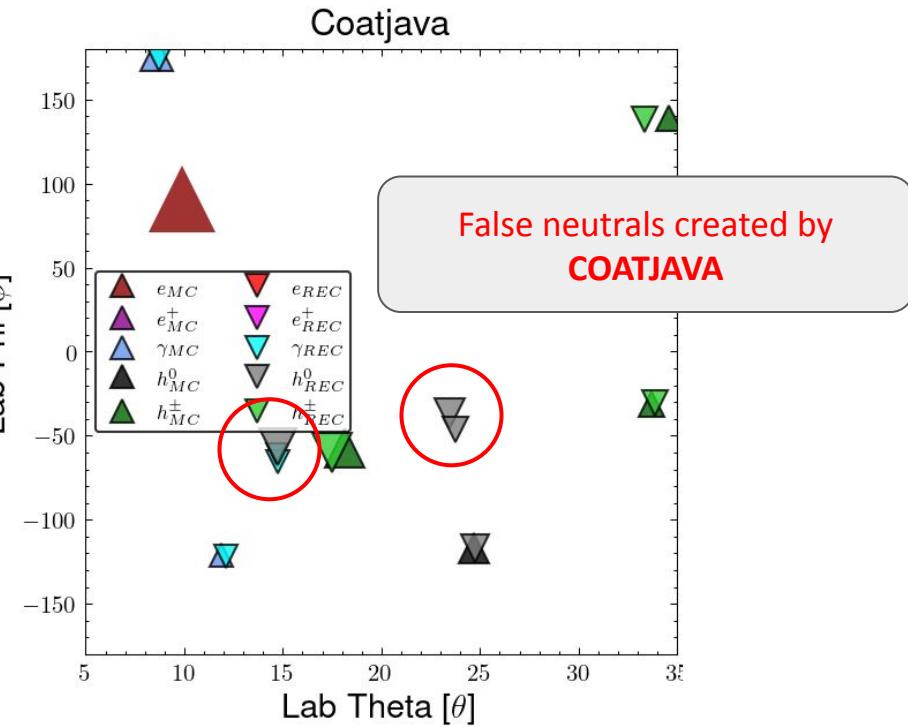
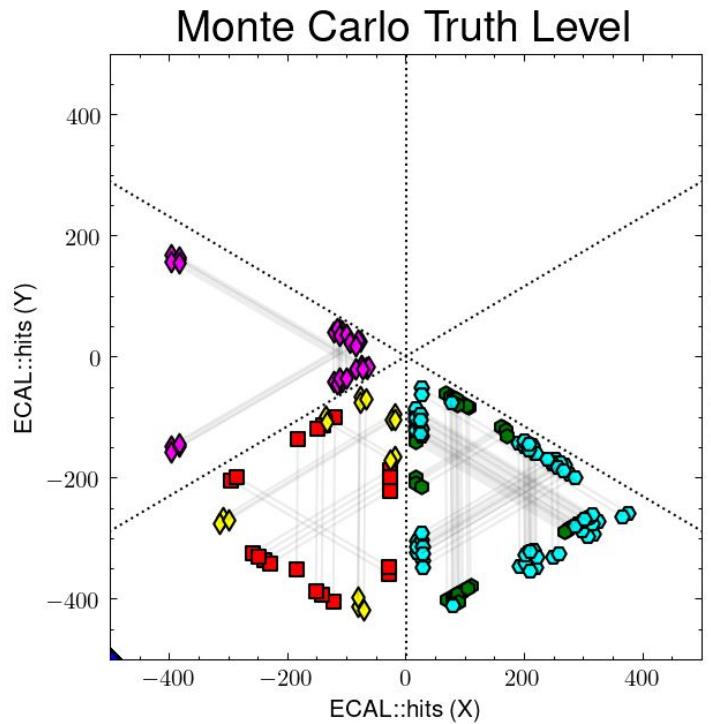
coatjava



- ◆ MC PID 0.0 ● MC π^+
- MC π^- ♦ MC γ
- MC n ♦ MC γ

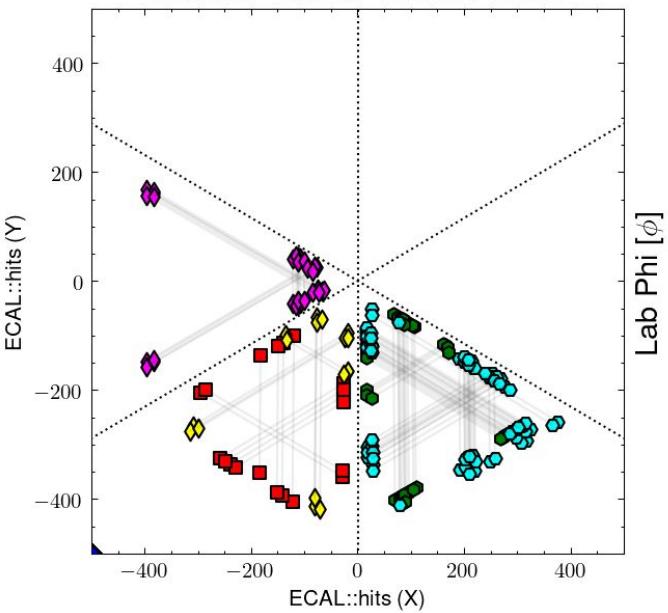
- + REC No match ■ REC n
- REC π^+ □ REC n
- REC n ♦ REC γ
- ♦ REC γ ♦ REC γ
- REC n ♦ REC γ

Sample Monte Carlo Event

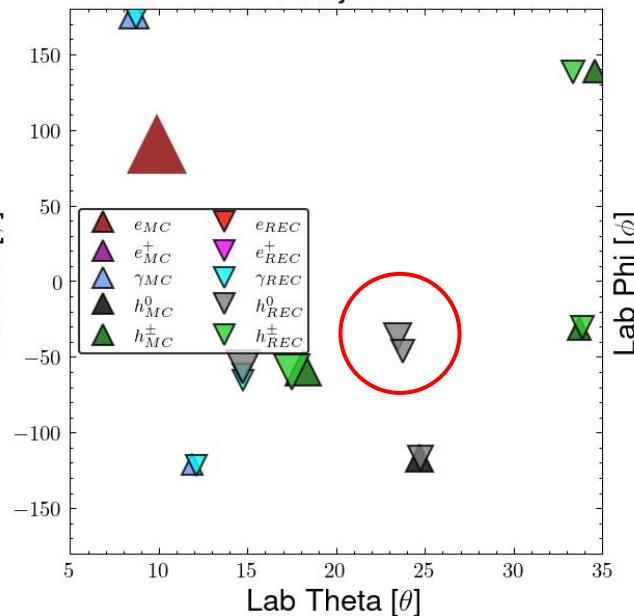


Sample Monte Carlo Event

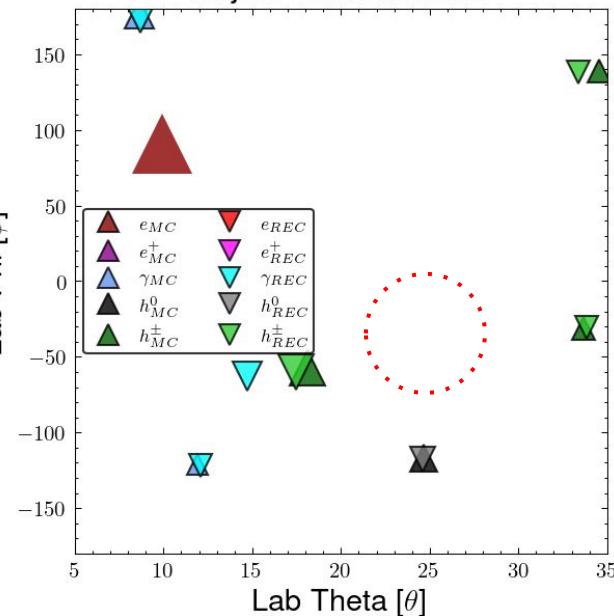
Monte Carlo Truth Level



Coatjava



Object Condensation

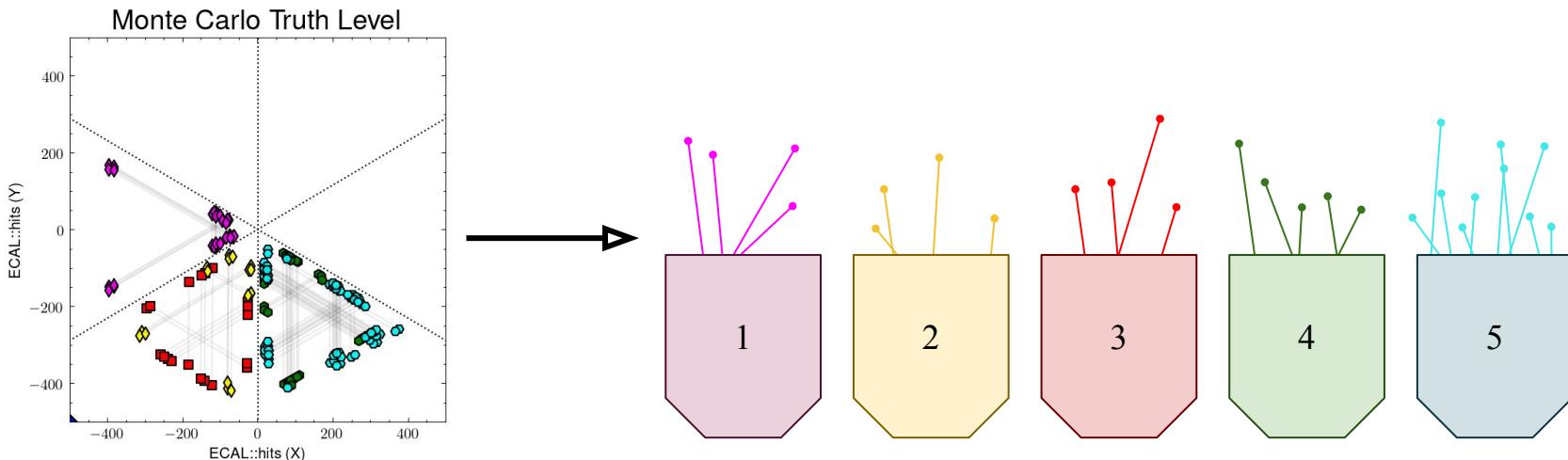


New **Machine Learning** approach effectively reduces the number of false neutrals without losing true neutrals

Defining the Problem

- **Input:** *Point Cloud* of ECAL strips with several features (layer, sector, E, t, x, y, z, etc)
 - For training 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*

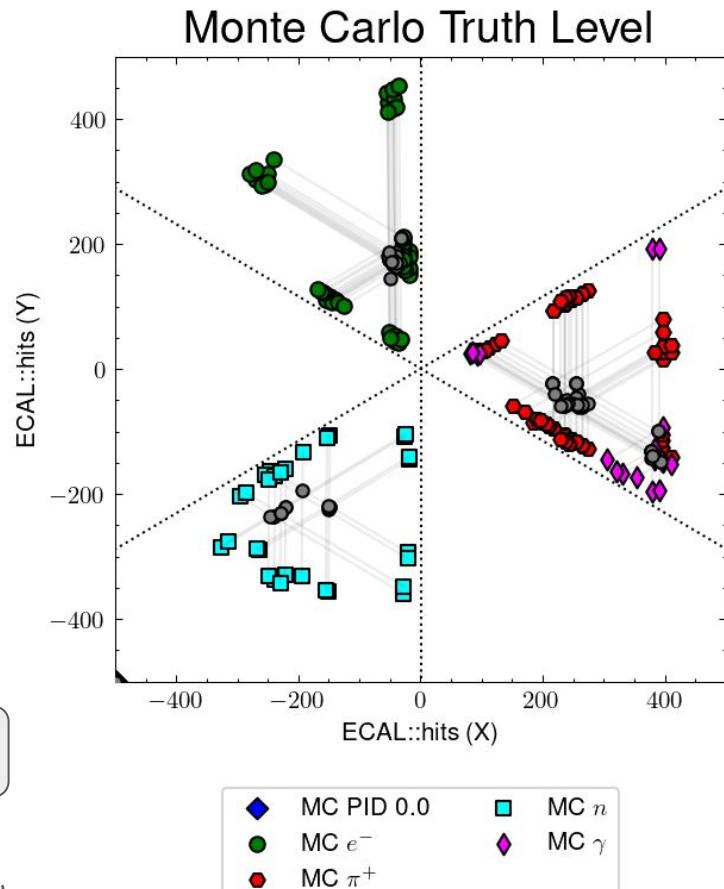
This is a much more abstract version of Image-within-Image classification



ML Input Considerations

Model Input Features (22)

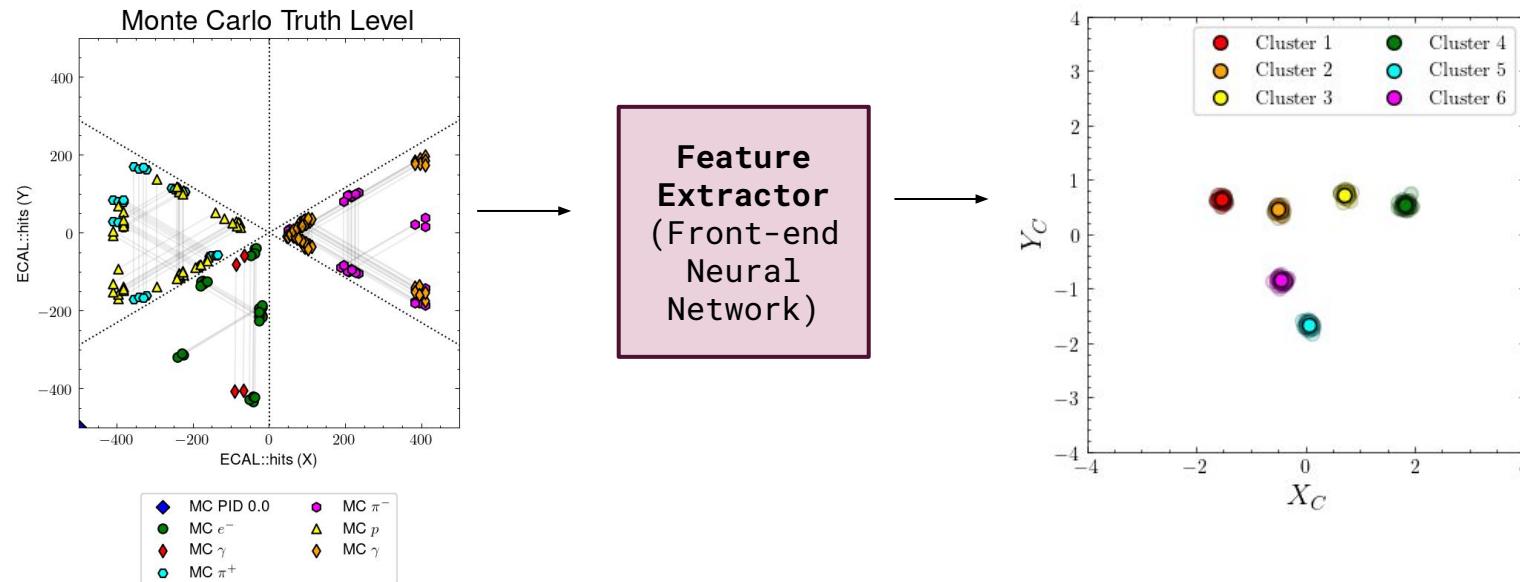
- [+] Strip Origin Point (x_1, y_1, z_1)
- [+] Strip End Point (x_2, y_2, z_2)
 - Red features scaled [-500,500] → [0,1]
 - Blue features are scaled [550-950] → [0,1]
- [+1] Energy Deposition (already [0,1])
- [+] Strip's most energetic centroid (x,y,z)
 - [+] One-hot encode for either 3 way or 2 way
- [+1] Timing Information [0,1000] → [0,1]
- [+] Layer
 - One-hot-encoded, 9 feature bits [0,1] total



Grey circles (right plot) show location of the energetic centroids

What is ★Object Condensation★?

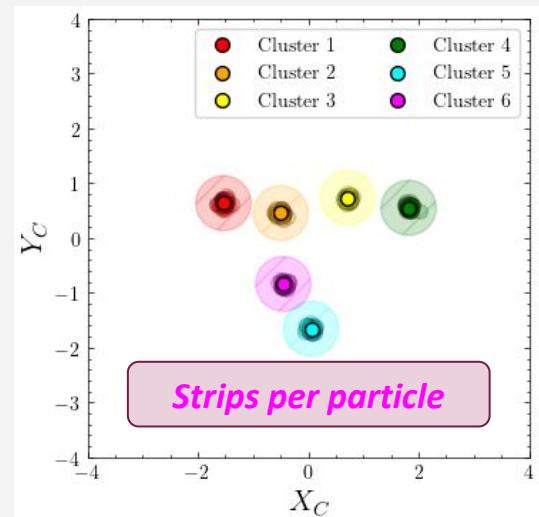
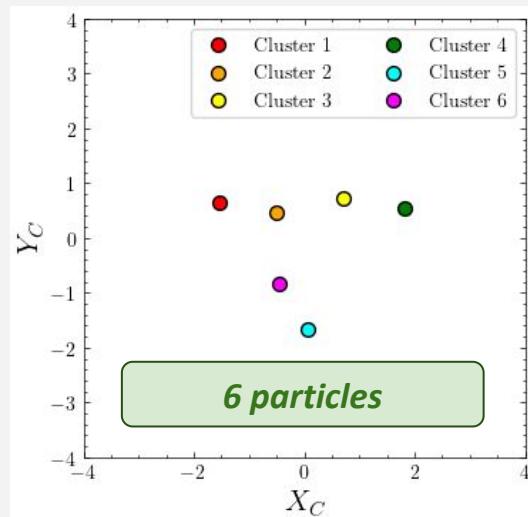
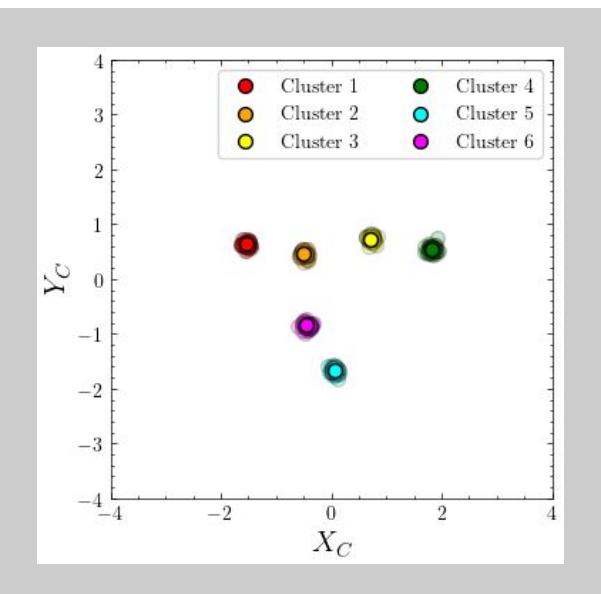
- Object Condensation defines a loss function that a neural network will try to minimize
- If this loss function is minimized, the Point Cloud is mapped to a clustered latent space
- Each ECAL strip learns its own point in the latent space (x_c , y_c) as well as a brightness ($0 < \square < 1$)
- For each object (particle) only one latent space pixel is “bright” (\square near 1)



What is ★Object Condensation★?

By viewing this clustered latent space (x_c , 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



★ Object Condensation★ Recap

- Input $\rightarrow v_{in}(N, F)$
 - **N**: Number of nodes (in our case number of strips)
 - **F**: Number of features per node (in our case 22)
- Output $\rightarrow v_{out}(N, 3) \rightarrow$ i.e. each strip learns 3 variables
 - $v_{out}[:, 0]$ is the x-coordinate in a latent space (called x_c)
 - $v_{out}[:, 1]$ is the y-coordinate in a latent space (called y_c)
 - $v_{out}[:, 2]$ is the brightness of the node (strip) in the latent space [0,1] (called \square)



Object Condensation (OC) defines a loss function $L(x_c, y_c, \square)$ 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 $\square \sim 1$ (**coward loss**)

Object Condensation Loss

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

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

Attractive Loss

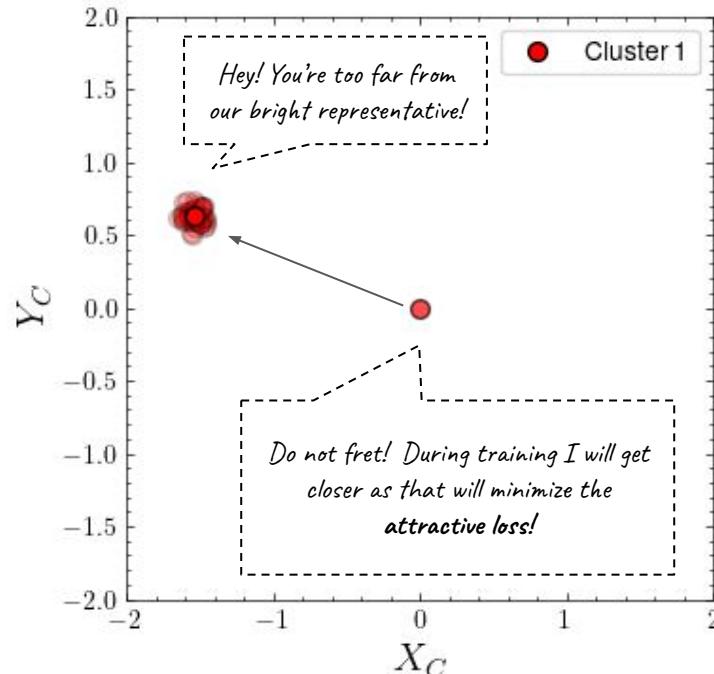
Each individual strip calculates one piece of the **attractive loss**

Very similar to E&M $\mathbf{U} = \mathbf{q}\mathbf{V}$

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

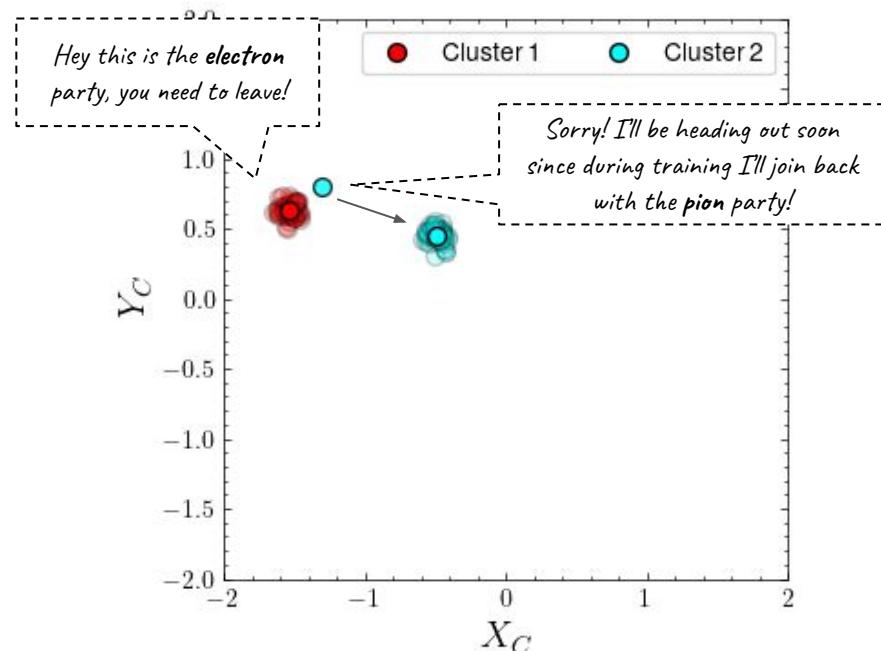
The *brightest* strip for particle (k) is αk

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



Object Condensation Loss

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



Repulsive Loss

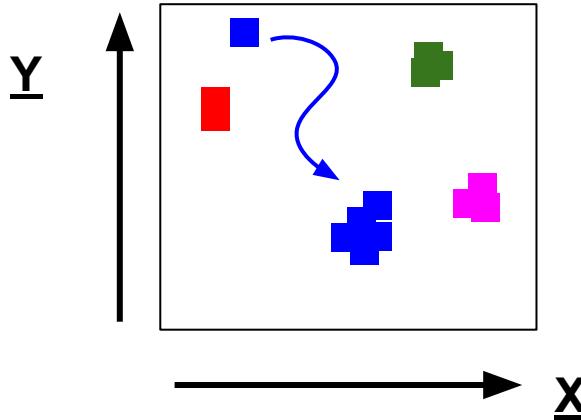
Each individual strip calculates **K-1** pieces of the **repulsive loss**

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

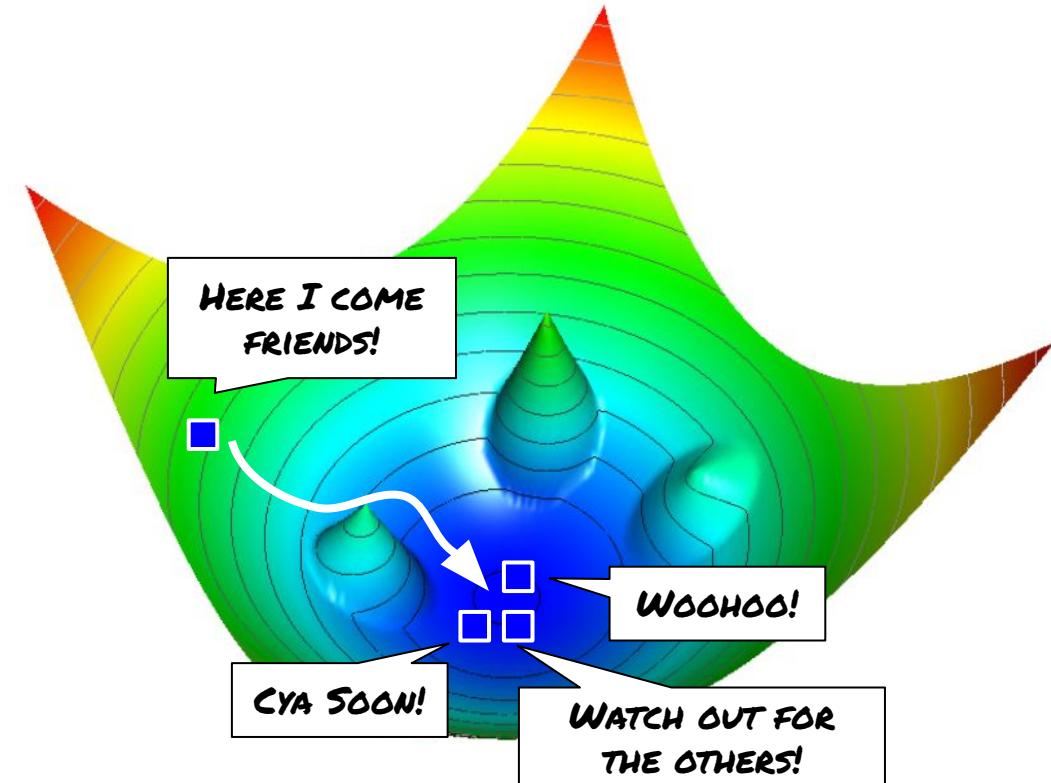
For each strip (j), punish the loss function the closer it is to the *brightest beta* of any other particle (k)

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

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

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

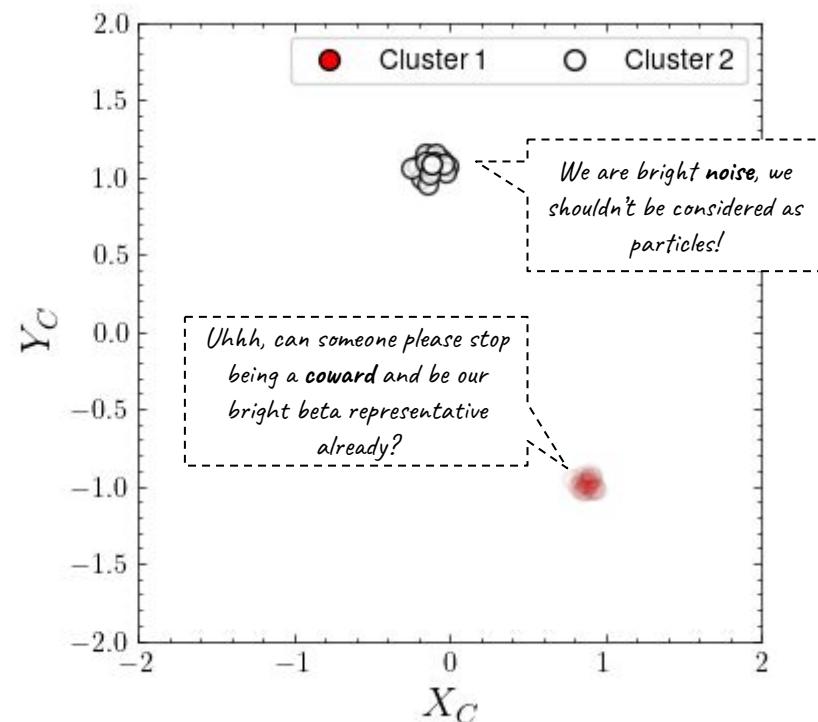
Coward Loss

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

Noise Loss

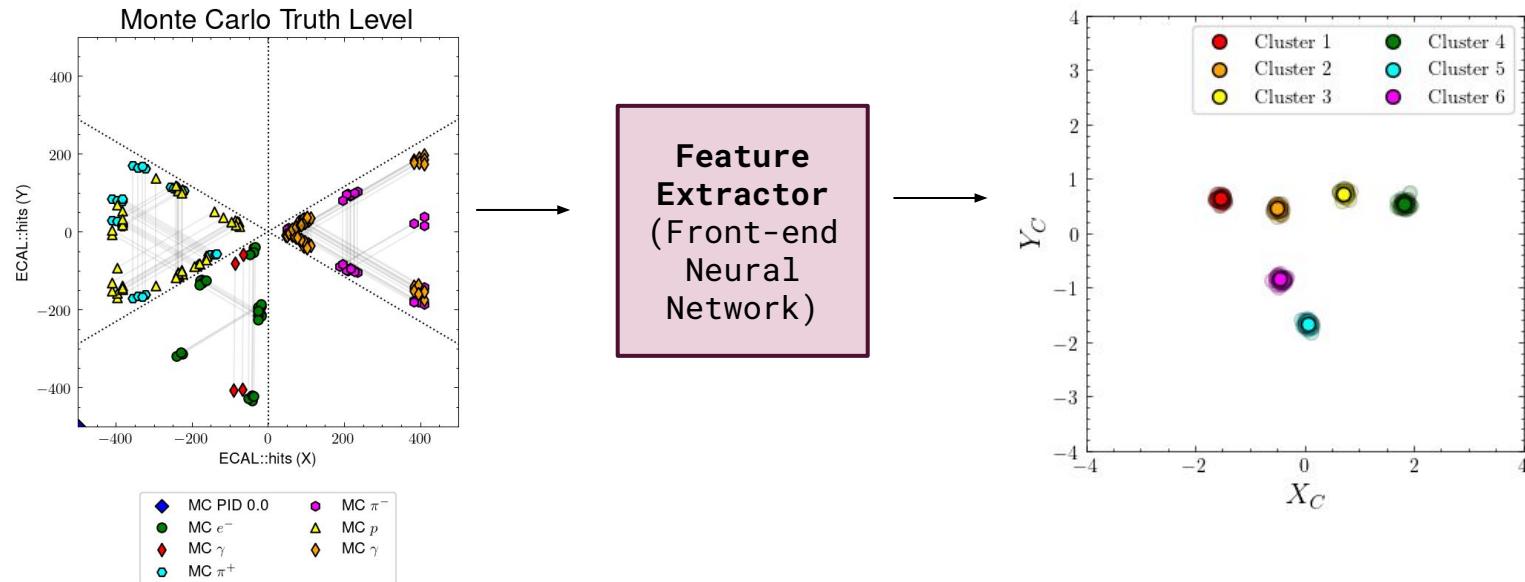
For each strip (i) punish the **noise loss** if the strip is noise (ex: 0-padded) and has a high *brightness beta*

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



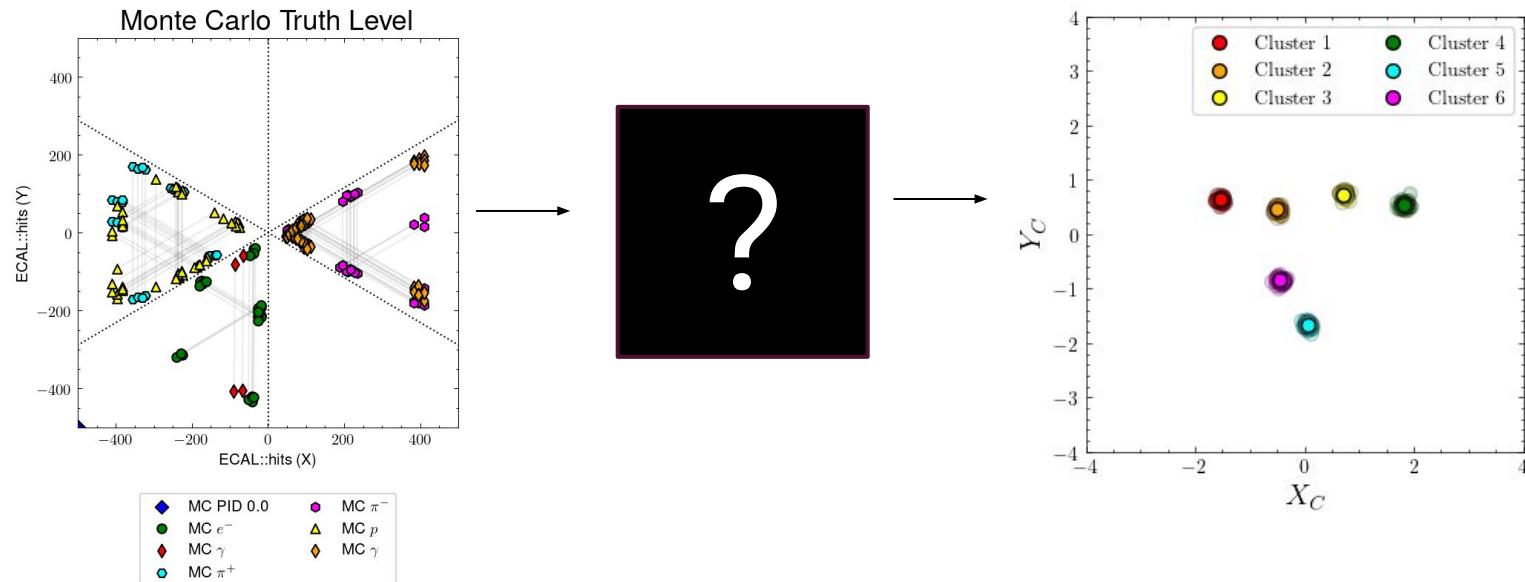
Object Condensation Summary

- A feature extractor maps the (N, F) strip input space into a (x_c, y_c) space
- The weights/biases of the feature extractor are backpropagated to minimize four Object Condensation losses (**attractive, repulsive, coward, noise**)

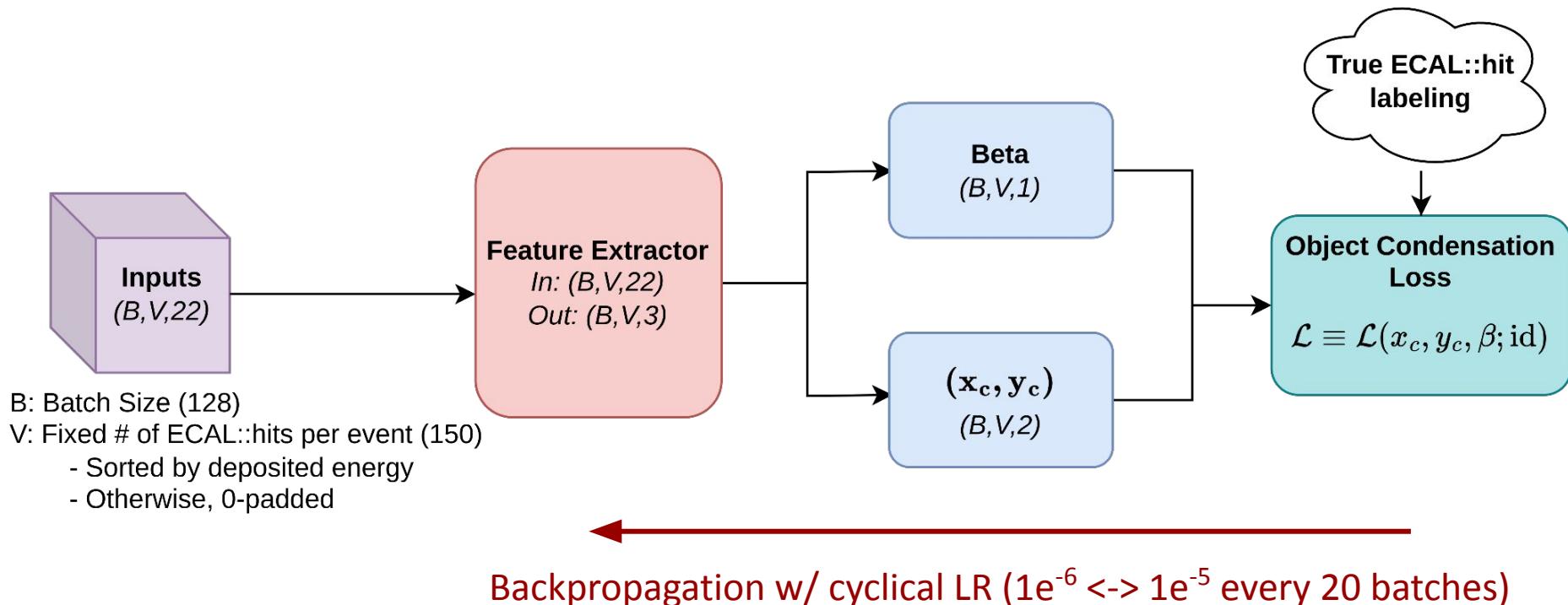


Object Condensation Summary

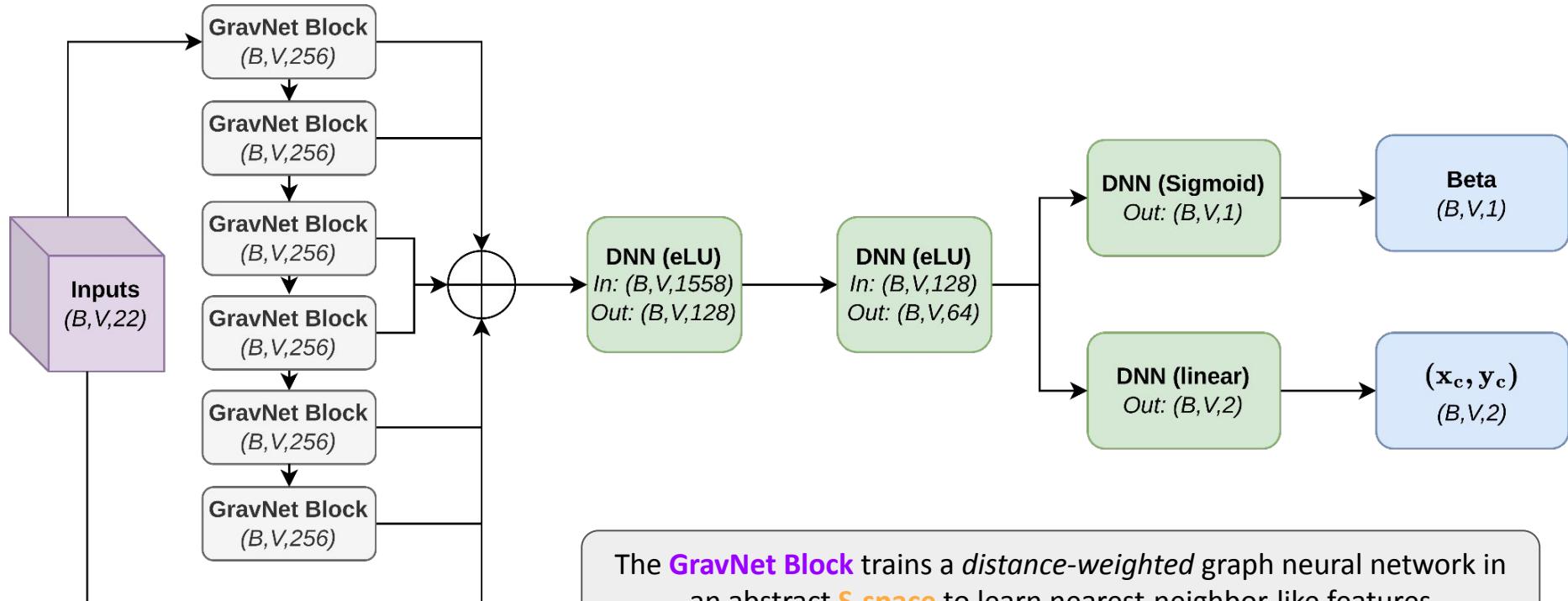
What is the Feature Extractor we use?



Model Architecture



Feature Extractor

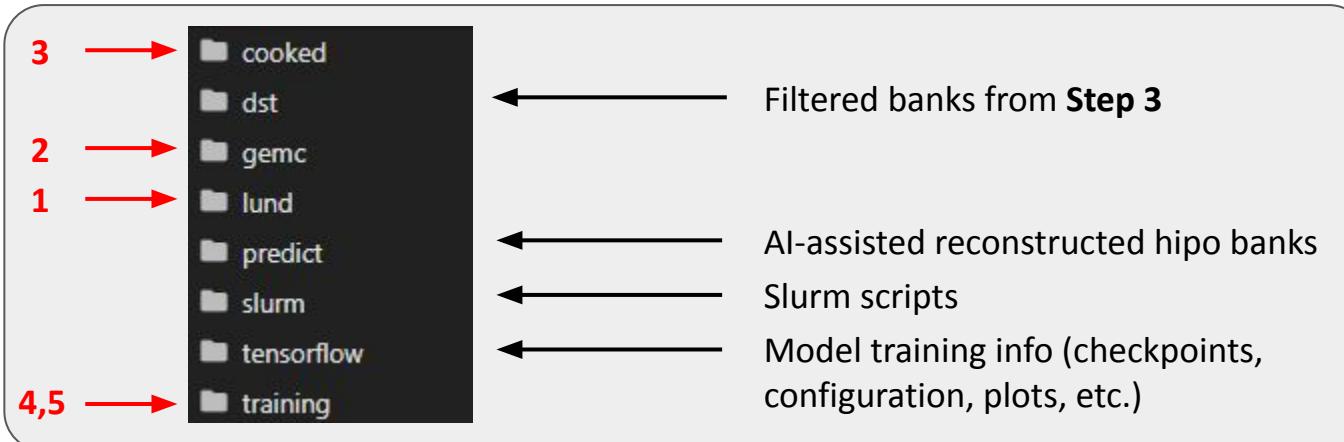


The **GravNet Block** trains a *distance-weighted* graph neural network in an abstract **S-space** to learn nearest-neighbor-like features

Subsequent **GravNet Blocks** are concatenated with the original input and fed into **DNNs** to extract **Beta** and (x_c, y_c) for each strip

Data Generation (see [repository](#)

1. 1M **e+p** DIS events simulated using **clasdis** split into 1000 batches
2. Detector readout simulated using **gemc** with MC::True saved (see [here](#) )
3. Create ECAL::hits and other familiar banks with **recon-util**
 - a. Custom [coatjava fork](#)  uses MC::True to give true/rec pid to each strip
4. Use **hipopy** to read ECAL::hits into csv files (see [here](#) )
 - a. Also parse REC::Particle (for comparisons) and MC::Particle (for training)
5. Preprocess the csv files into h5 files for training (ex: scaling features, get centroids)



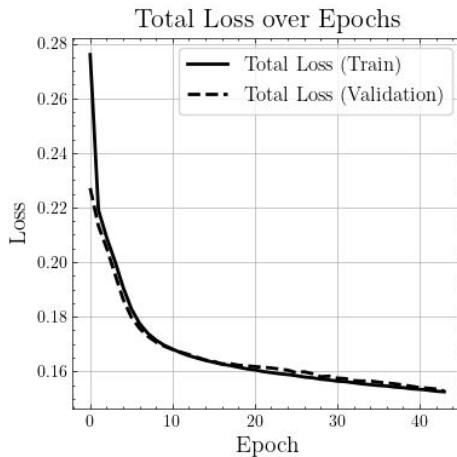
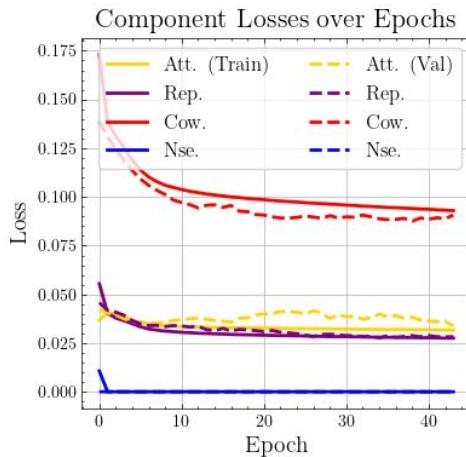
Training Information

- ❖ 128 events per batch
- ❖ 100 epochs (~45 per 24hrs training on 1 TitanRTX GPU)
- ❖ 80% - 20% Training/Validation splitting
- ❖ GravNet Feature Extractor Information
 - 10 GravNet blocks – 10 nearest neighbors – 4 S-space dimensions – 32 hidden features – 256 output
 - Total trainable parameters = 637,934
- ❖ Hyperparameters
 - $q_{\min} = 0.1$
 - $s_B = 1$ (only noise considered is 0-padded anyway, not hard for model to figure that out)
 - $t_B = 0.5$ (minimum *brightness* threshold)
 - $t_D = 0.28$ (radius of cluster in latent space, **decided by eye, but need to develop rigorous metric**)
- ❖ Cyclical Learning rate between 10^{-6} \leftrightarrow 10^{-5} every 20 batches (help navigate out of local minima)
- ❖ Stopping Procedure
 - Stop if validation loss does not improve for 10 epochs (not yet seen)

Loss Function

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

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

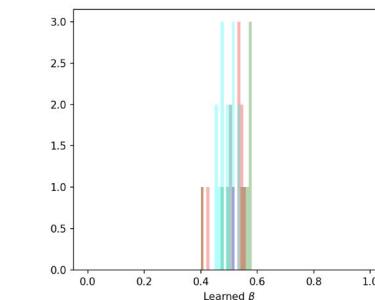
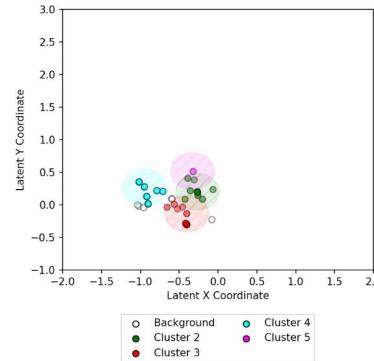
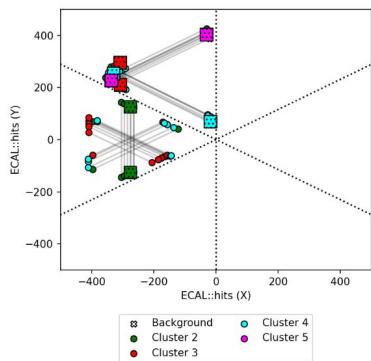
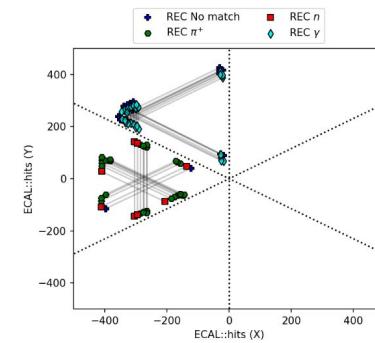
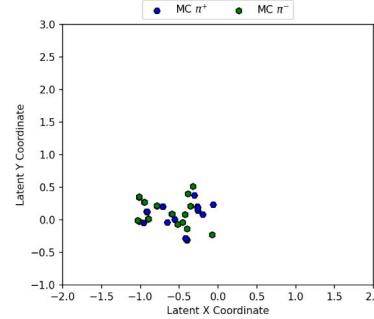
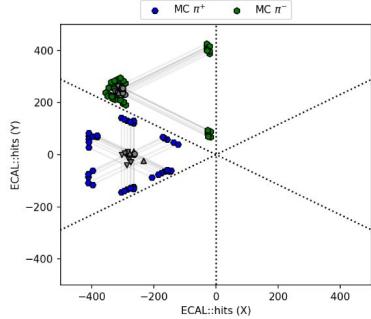


- ★ Per epoch evaluation of **Obj. Condensation Loss**
- ★ Solid lines = Training // Dotted lines = Validation
- ★ Since validation loss is evaluated at the end of each epoch, the fast-learning early epochs have validation loss < train loss
- ★ Overfitting not yet seen, but training stopped at 24 hours. [Working on extending training](#)

Training Visualization (Event A)

Smaller model used to make plots quickly

Epoch 1, Batch 0000, Loss = 1.0922



Top Left

Monte Carlo true hits

Top Middle

True hits in latent space $[x_c, y_c]$
(colors match TL plot)

Top Right

REC::Particle reco hits

Bottom Left

Hits clustered in latent space

* Box = Brightness Beta *

Bottom Middle

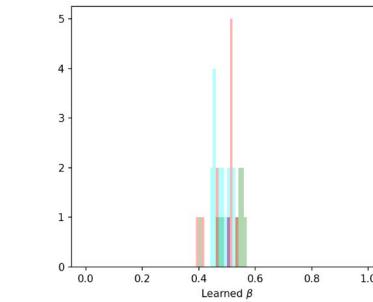
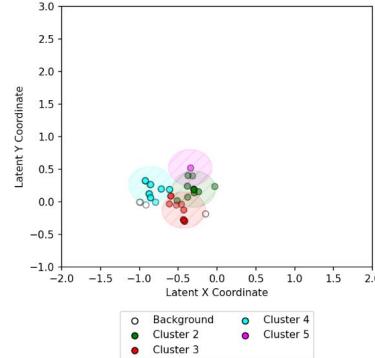
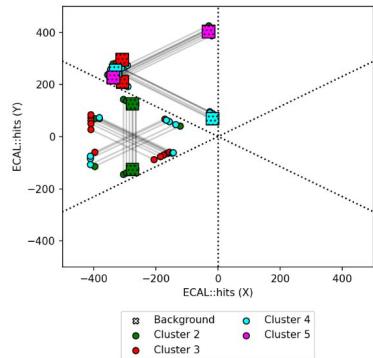
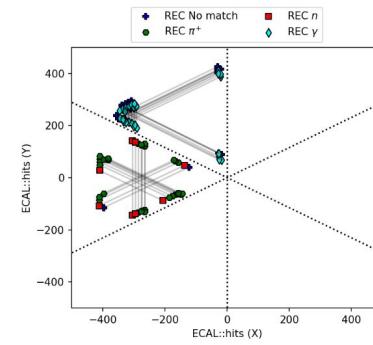
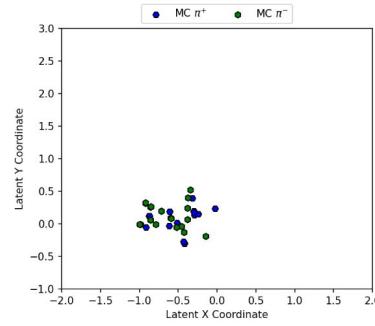
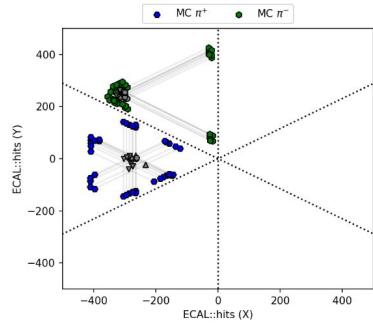
Latent space with $t_B=0.5$ and
 $t_D=0.28$ used to determine
clusters (colors match BL plot)

Bottom Right

Histogram of the strip
brightness (\square) values

Training Visualization (Event A)

Epoch 1, Batch 0001, Loss = 1.0721



Top Left

Monte Carlo true hits

Top Middle

True hits in latent space $[x_c, y_c]$
(colors match TL plot)

Top Right

REC::Particle reco hits

Bottom Left

Hits clustered in latent space

* Box = Brightness Beta *

Bottom Middle

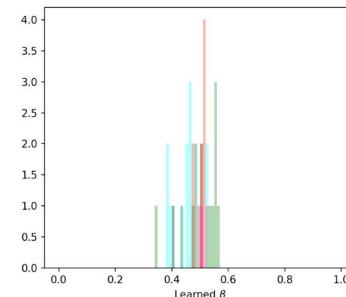
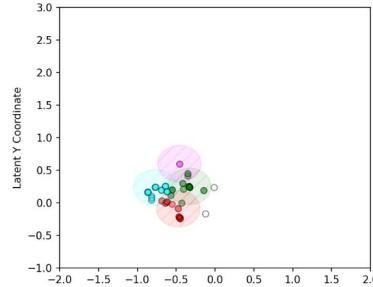
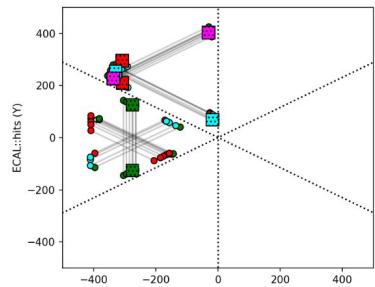
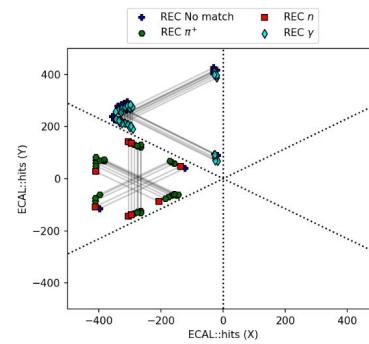
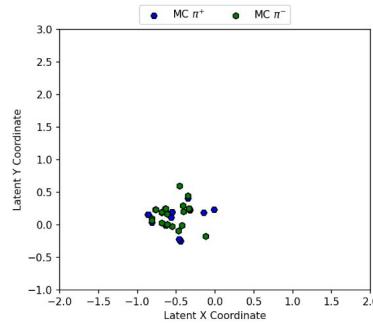
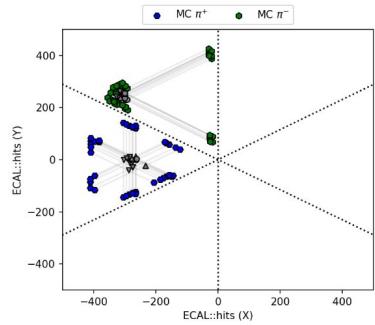
Latent space with $t_B=0.5$ and
 $t_D=0.28$ used to determine
clusters (colors match BL plot)

Bottom Right

Histogram of the strip
brightness (\square) values

Training Visualization (Event A)

Epoch 1, Batch 0002, Loss = 1.0444



Top Left

Monte Carlo true hits

Top Middle

True hits in latent space $[x_c, y_c]$
(colors match TL plot)

Top Right

REC::Particle reco hits

Bottom Left

Hits clustered in latent space

* Box = Brightness Beta *

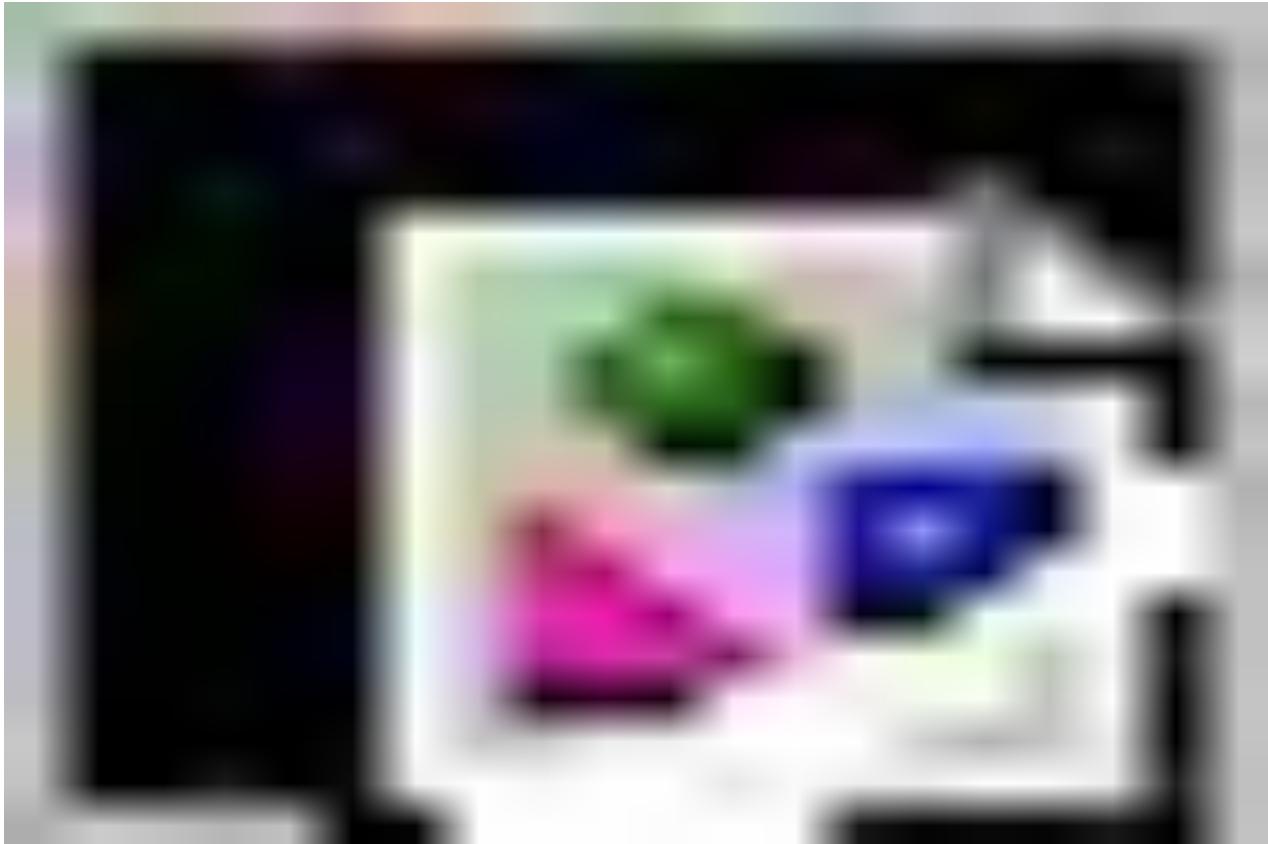
Bottom Middle

Latent space with $t_B=0.5$ and
 $t_D=0.28$ used to determine
clusters (colors match BL plot)

Bottom Right

Histogram of the strip
brightness (\square) values

Training Visualization (Event A)



Top Left

Monte Carlo true hits

Top Middle

*True hits in latent space $[x_c, y_c]$
(colors match TL plot)*

Top Right

REC::Particle reco hits

Bottom Left

*Hits clustered in latent space
* Box = Brightness Beta **

Bottom Middle

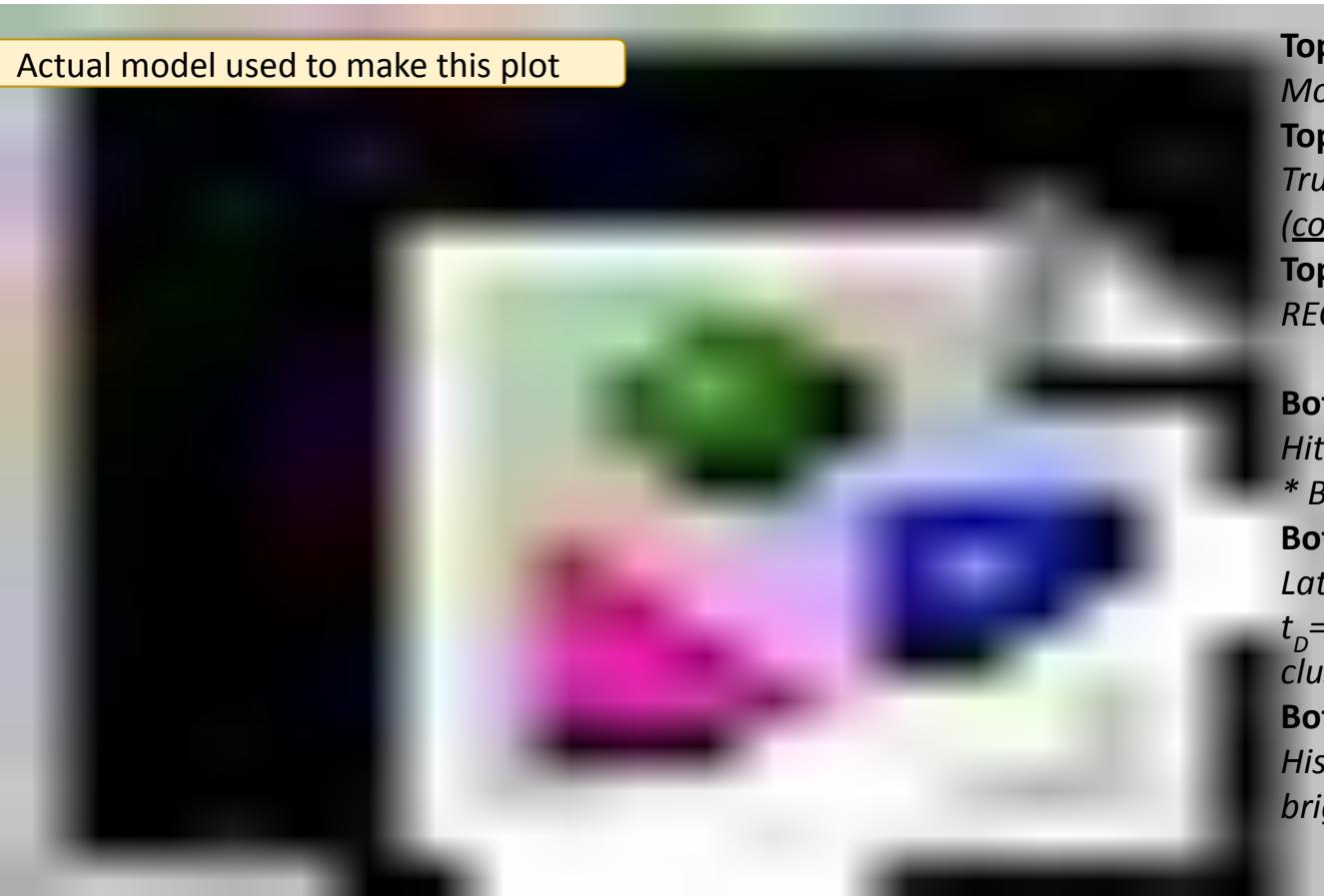
*Latent space with $t_B=0.5$ and
 $t_D=0.28$ used to determine
clusters (colors match BL plot)*

Bottom Right

*Histogram of the strip
brightness (\square) values*

Training Visualization (Event B)

Actual model used to make this plot



Top Left

Monte Carlo true hits

Top Middle

*True hits in latent space $[x_c, y_c]$
(colors match TL plot)*

Top Right

REC::Particle reco hits

Bottom Left

*Hits clustered in latent space
* Box = Brightness Beta **

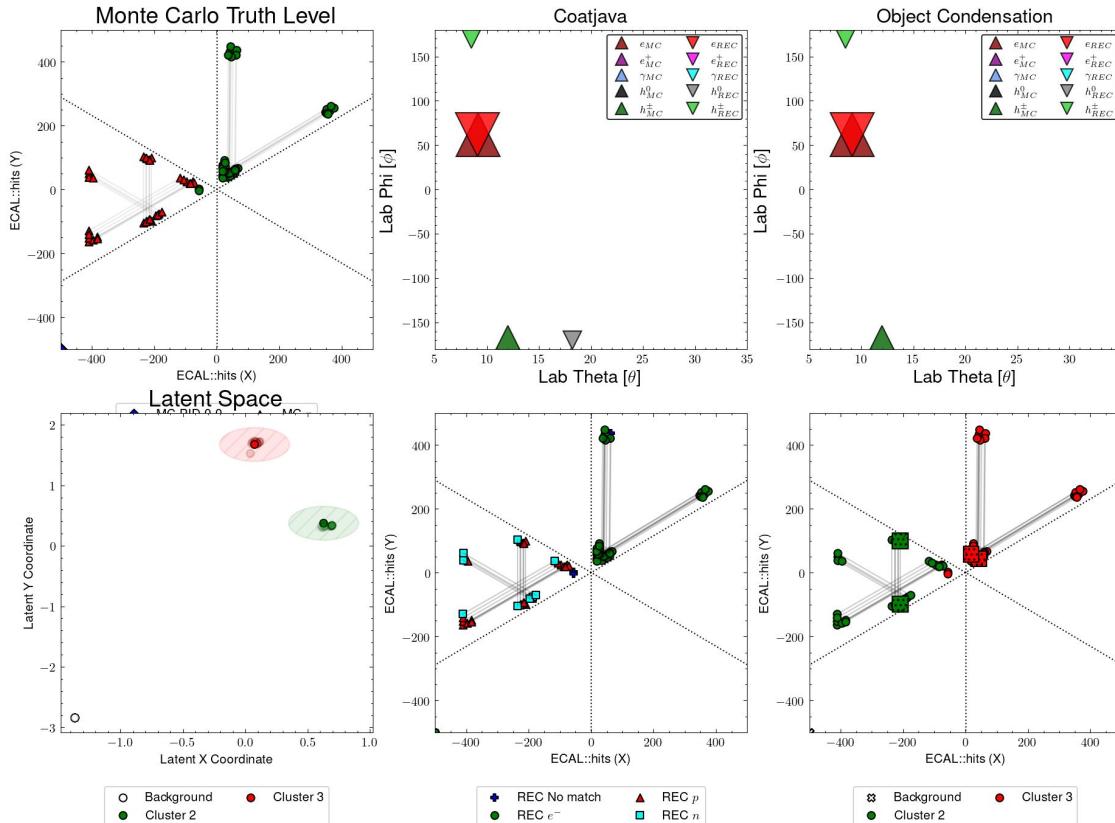
Bottom Middle

*Latent space with $t_B=0.5$ and
 $t_D=0.28$ used to determine
clusters (colors match BL plot)*

Bottom Right

*Histogram of the strip
brightness (\square) values*

Training Results (Event C)

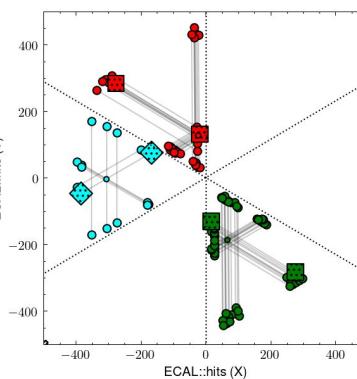
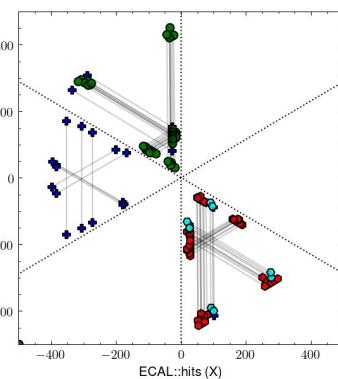
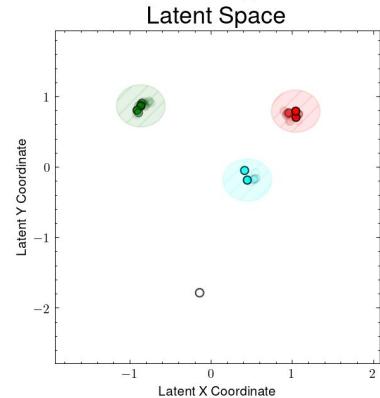
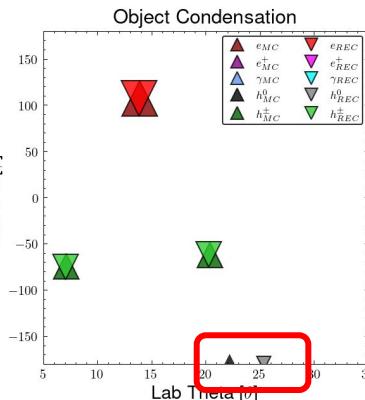
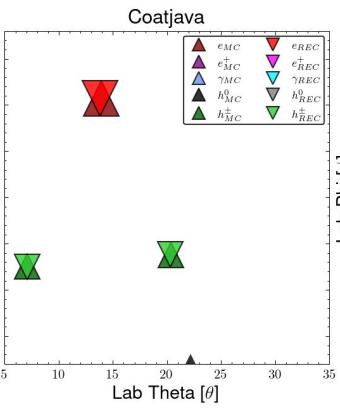
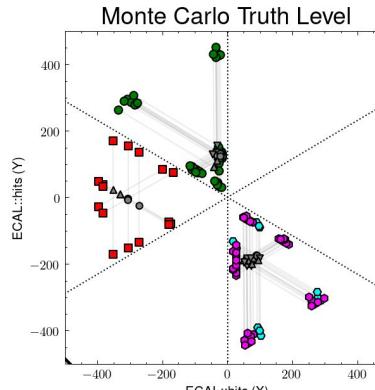


Coatjava (middle column)
reconstructs an extra false
neutral particle in Sector 4

Object Condensation (right
column) does not make this
mistake, finding one cluster here

*Obj. Cond.=More comprehensive
understanding of what should be
considered a cluster*

Training Results (Event D)



Coatjava (middle column) does not find the neutron in Sector 4

Object Condensation (right column) does find this neutron but the ambiguity of the 3-way intersection leads to a misreconstruction of **theta**

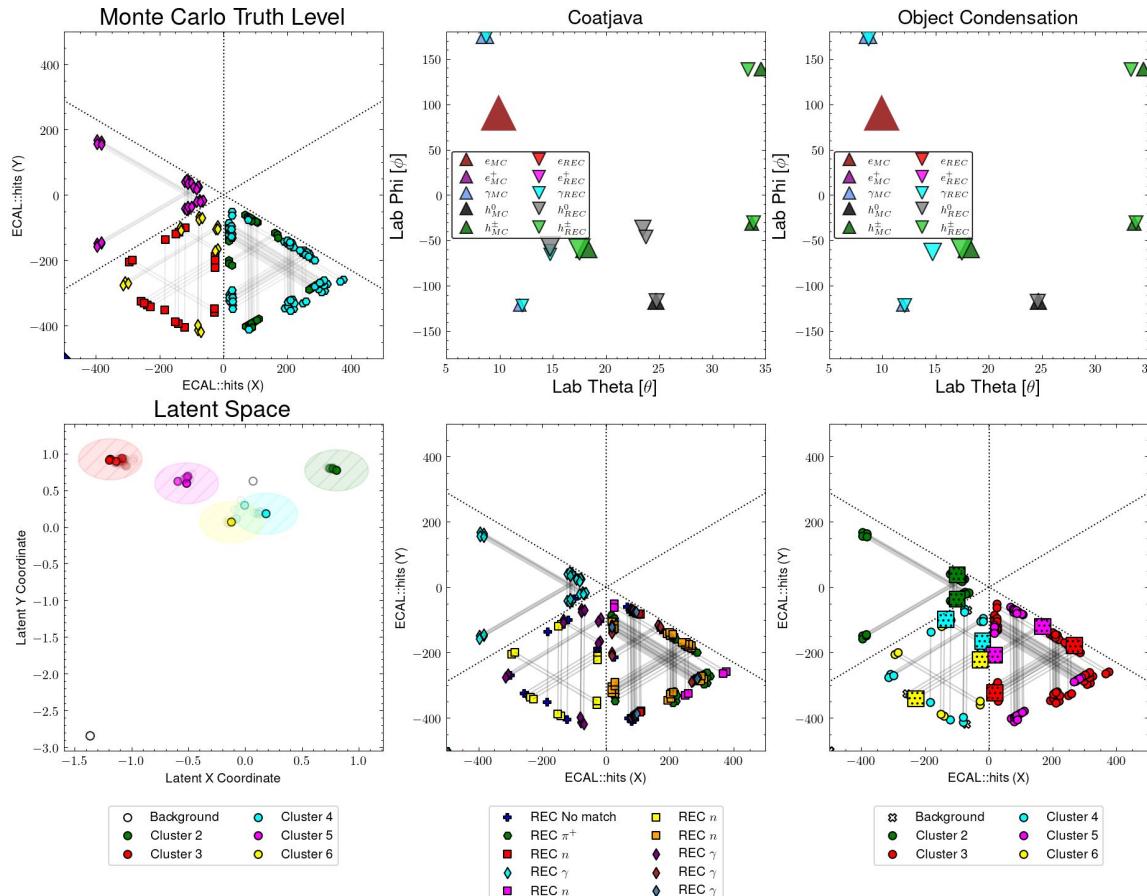
Obj. Cond.=Could use more development in calculating the centroid location after clustering

○ Background
● Cluster 2

● REC No match
● REC e^-

● REC π^-
● REC π^+

Training Results (Event E)



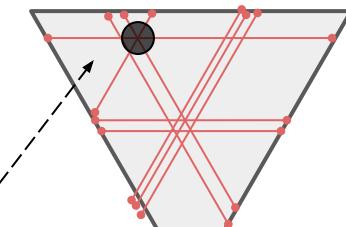
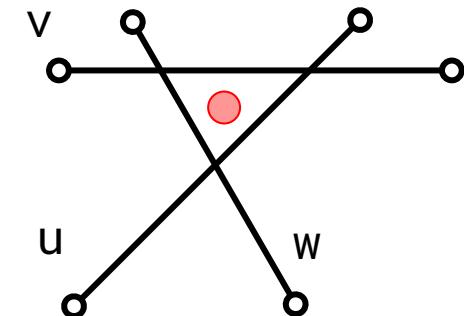
Coatjava (middle column) finds a swamp of neutrals in Sector 6

Object Condensation (right column) correctly identifies only two unique clusters in Sector 6.

Obj. Cond.=Can handle multiple particles in a sector and still predict their centroid effectively

Creating new ECAL::clusters bank ([python](#))

1. Loop over PCAL, ECIN, and ECOUT strips
2. For each strip (j) belonging to cluster (k)
 - a. Find its most energetic ($\sum_j E_j$) 3-way intersection. A 3-way intersection is determined by the average (x,y,z) of closest approach for **uv**, **vw**, **uw** strips. E_j is energy corrected to account for attenuation!
3. For each cluster (k) containing (N) 3-way intersections
 - a. Only consider 3-way intersections in the sector with a 50%+ majority
 - b. Calculate the z-score z_i for each 3-way intersection (x,y,z)
 - c. Report the centroid's (x,y,z) as the weighted sum of the 3-way intersections, where $w_i = (1+z^2)^{-1}$ to lessen the impact of *distantly separately 3-way's*
4. Purposefully assign ECAL::cluster status of PCAL, ECIN, ECOUT to a uniquely identifiable status value to force REC::Calorimeter to recognize them as a group
5. Generate new ECAL::clusters, ECAL::calib (empty) and ECAL::moments (empty)



Creating new ECAL::clusters bank ([python](#))

index :	2	3	1	0
pindex :	1	1	2	3
detector :	7	7	7	7
sector :	3	3	2	5
layer :	1	4	1	7
energy :	0.1823	0.0090	0.3472	0.2481
time :	149.9131	151.2900	149.1106	155.7128
path :	764.8085	801.8519	744.9232	770.0773
chi2 :	0.0000	0.0000	0.0000	0.0000
x :	-37.5027	-29.0460	93.4869	-120.4040
y :	70.6440	58.9809	78.6126	-239.4463
z :	752.1381	794.9993	735.1639	722.0366
hx :	-39.3423	-38.3846	93.4869	-120.4040
hy :	68.0255	66.5349	78.6126	-239.4463
hz :	753.1024	790.1033	735.1639	722.0366
lu :	0.0000	0.0000	0.0000	0.0000
lv :	0.0000	0.0000	0.0000	0.0000
lw :	0.0000	0.0000	0.0000	0.0000
du :	0.0000	0.0000	0.0000	0.0000
dv :	0.0000	0.0000	0.0000	0.0000
dw :	0.0000	0.0000	0.0000	0.0000
m2u :	0.0000	0.0000	0.0000	0.0000
m2v :	0.0000	0.0000	0.0000	0.0000
m2w :	0.0000	0.0000	0.0000	0.0000
m3u :	0.0000	0.0000	0.0000	0.0000
m3v :	0.0000	0.0000	0.0000	0.0000
mcw :	0.0000	0.0000	0.0000	0.0000
status :	2	2	1	0

Purposefully assign ECAL::cluster status of PCAL, ECIN, ECOUT to a uniquely identifiable status value to force REC::Calorimeter to recognize them as a group

Implementation added in personal coatjava fork in order to force all clusters with identical status to be assigned to the same REC::Particle

See *processNeutralTracks_OC* in [EventBuilder.java](#)

```
// Group DetectorResponses by their status
for (DetectorResponse response : allResponses) {
    int status = response.getStatus();
    groupedResponses.computeIfAbsent(status, k -> new ArrayList<>()).add(response);
}
```

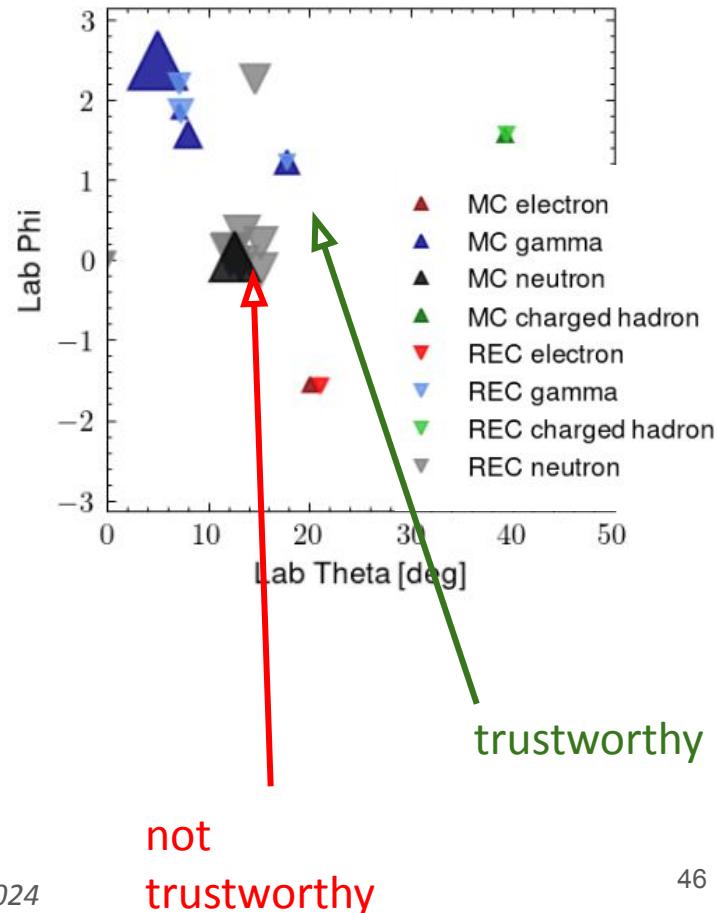
Benchmarks – Coatjava vs. Obj Condensation

Next we will discuss three of the ways I directly compared the two clustering methods

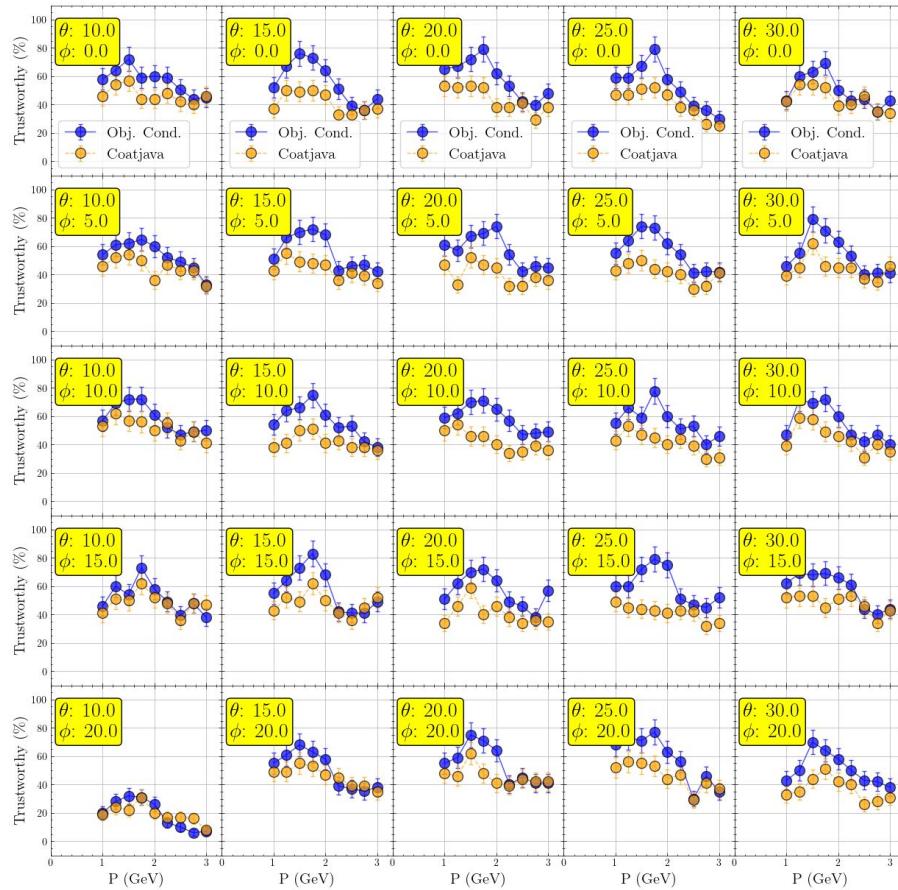
1. (P, θ, ϕ) binned neutron gun events
2. Incoherent J/Psi production off deuterium ($e+n \rightarrow e'+J/\Psi + n'$)
3. clasdis Monte Carlo SIDIS events

We classify REC::Particle's as trustworthy if ...

- A. There is an MC::Particle within ...
 $\delta\theta < 4$ [deg] and $\delta\varphi < 8$ [deg]
- B. The matched MC::Particle has the same pid as the REC::Particle
- C. There are no other REC::Particles that also satisfy this requirement for that MC::Particle



Particle Gun Benchmark



- 1000 **e+n** events in each bin
- Columns: $\theta=10,15,20,25,30$ [deg]
- Rows: $\phi=0,5,10,15,20$ [deg]
- Data points: $1 < P < 3$ [GeV]

Observations:

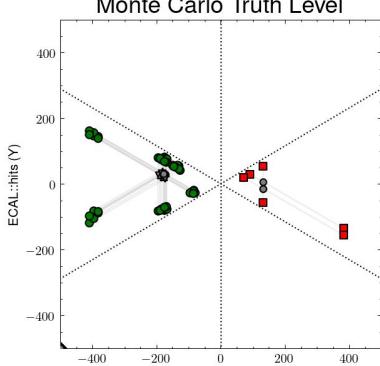
Average improvement of **Object Condensation** vs. **Coatjava** of 20-40%

Can be further improved because of the 3-way intersection issue... (see next slide)

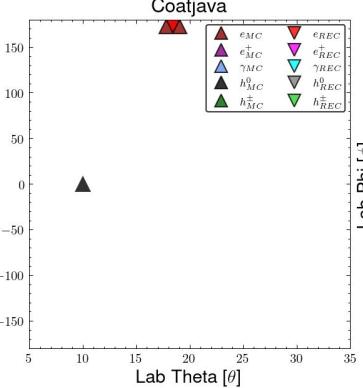
Should extend study to smaller momentum

Particle Gun Benchmark

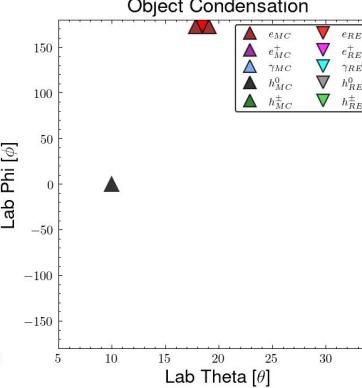
Monte Carlo Truth Level



Coatjava



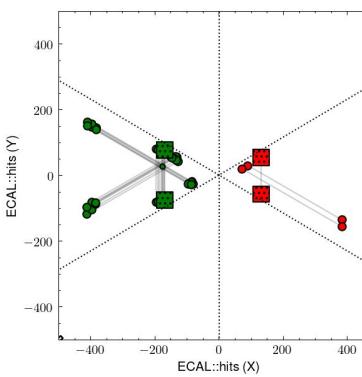
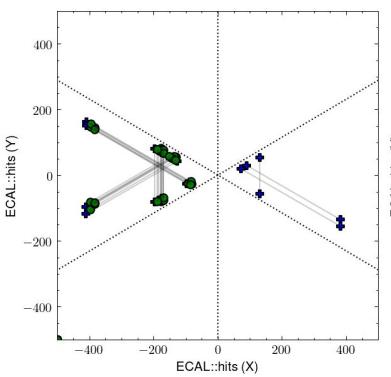
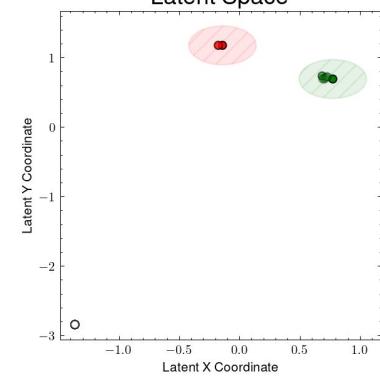
Object Condensation



Here, the electron and neutron are thrown and leave hits in sectors 4 and 1, respectively...

Despite **Object Condensation** clustering the neutron hits as an object (bottom right red cluster), since there is no 3-way intersection, we do not assign it as a REC::Particle later

Latent Space



○ Background
● Cluster 3
● Cluster 2

● REC No match ● REC e⁻

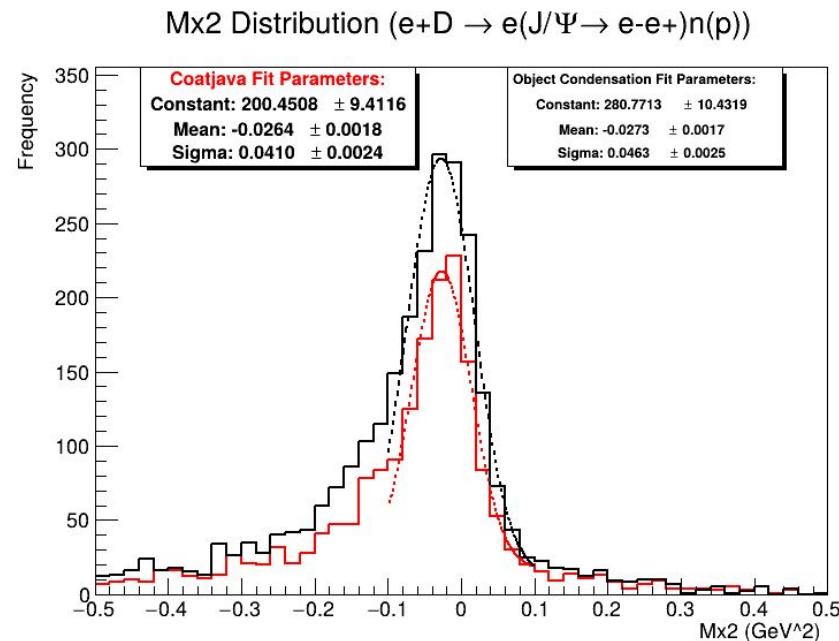
○ Background
● Cluster 2

Future: Train Obj. Cond. to predict neutral particle Px, Py, Pz and bypass needing ECAL cluster

Incoherent J/Psi Production Benchmark

- 1M spherically generated $e+n \rightarrow e' + (J/\psi \rightarrow e^+e^-) + n$ events (w/ Fermi motion)
- For comparison, we make simple cuts
 - $N_{\text{electrons}} = 2$
 - $N_{\text{positrons}} = 1$
 - $N_{\text{neutrons}} = 1$
- We see that Object Condensation provides a roughly 40% increase

Richard Tyson is completing a more thorough comparison using his analysis pipeline to come to a more accurate conclusion

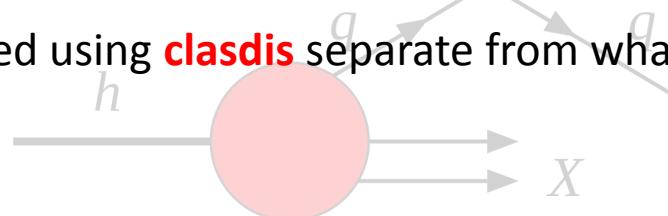


SIDIS Monte Carlo Benchmark

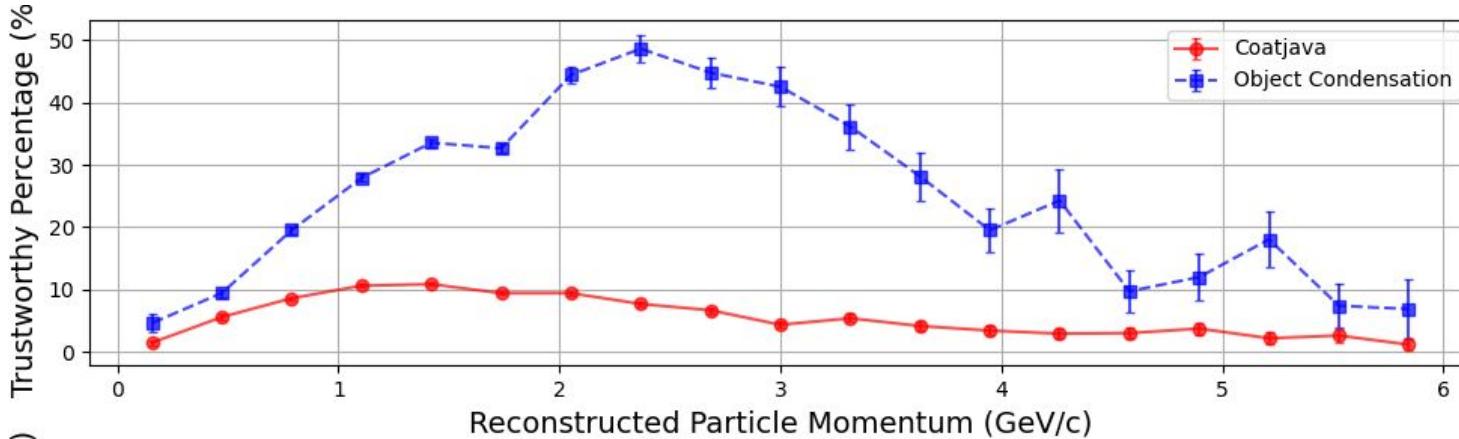
- Provides the most complex comparison... ex: multiple particles/types per sector
- One major advantage *currently* for **Coatjava** is that one strip can belong to two particles/clusters in the same layer...still working how to adjust for this in Obj. Cond.
- Typically, photons leave less hits in the calorimeter per cluster than neutrons which means Obj. Cond. has a disadvantage in finding photons

In the next slides, we compare the **Momentum** and **Theta** dependence of trustworthy neutrons/photons between Coatjava and Object Condensation...

1M e+p DIS events simulated using **clasdis** separate from what was used during training

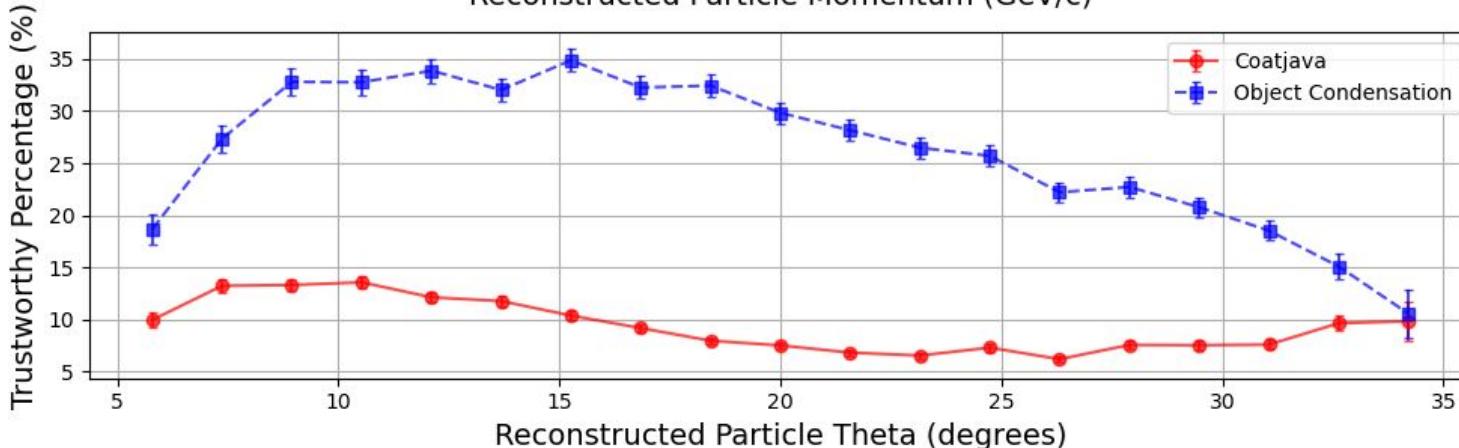


Trustworthy REC::Neutron Percentage



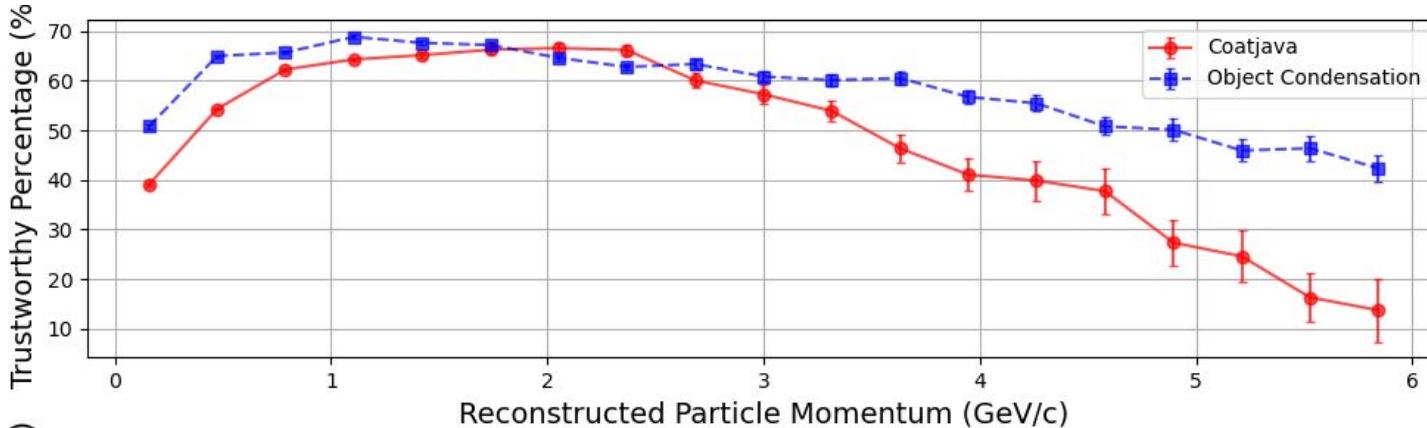
Coatjava
87383 REC neutrons
- 14164 w/ match
- 7708 trustworthy

Object Condensation
29858 REC neutrons
- 8187 w/ match
- 8187 trustworthy



3.1x increase in the
trustworthiness of a
REC::Neutron
(8.82% → 27.4%)

Trustworthy REC::Photon Percentage



Coatjava

179039 REC photons

- 100245 w/ match

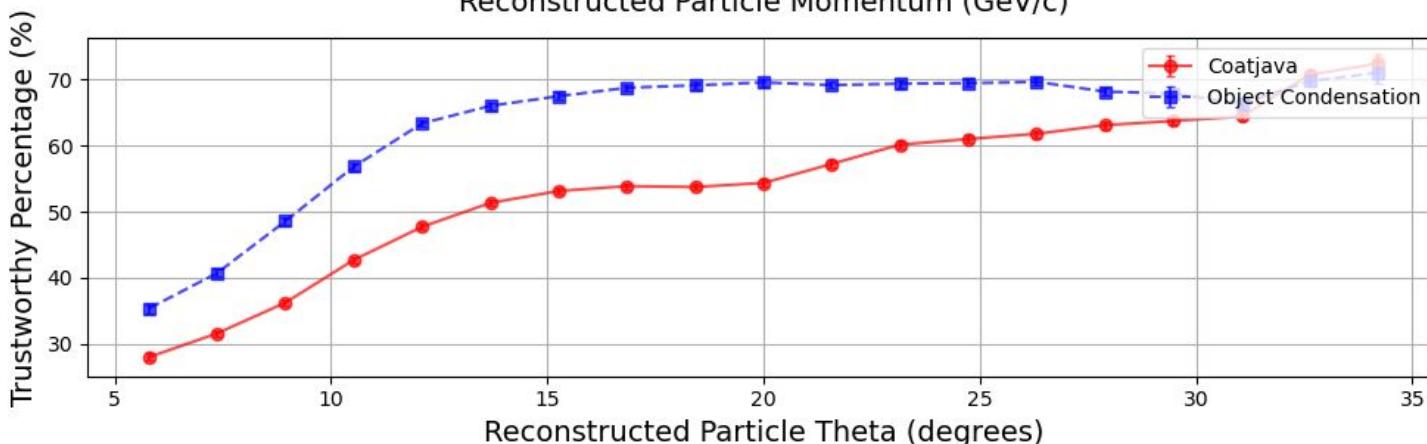
- 92233 trustworthy

Object Condensation

131555 REC photons

- 82817 w/ match

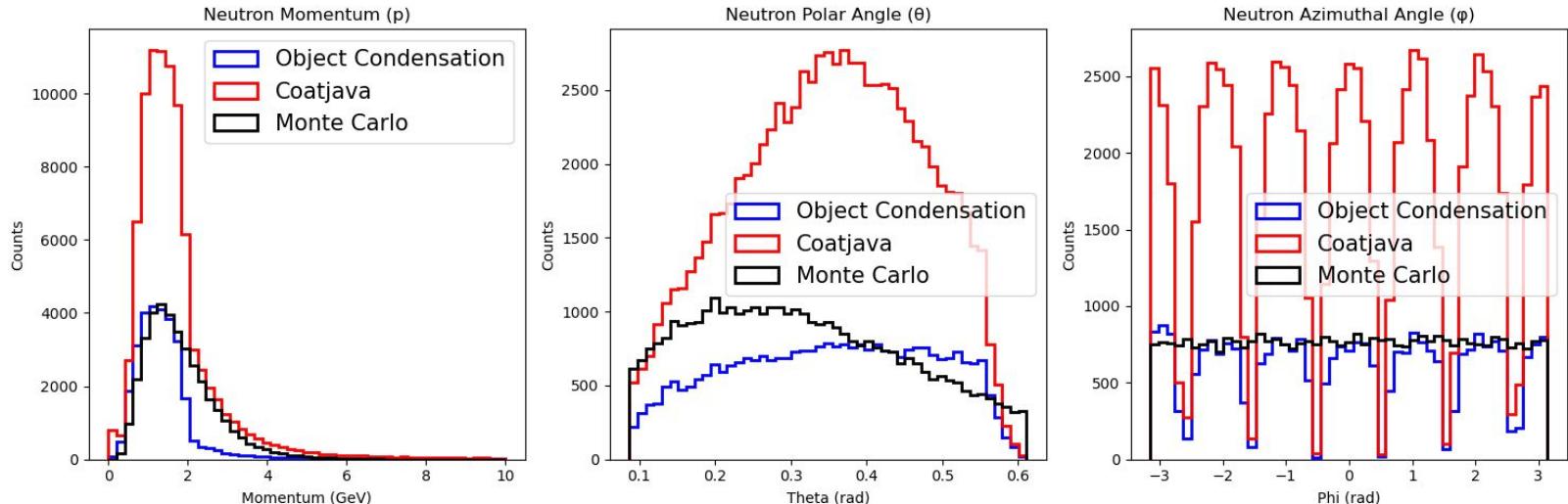
- 82798 trustworthy



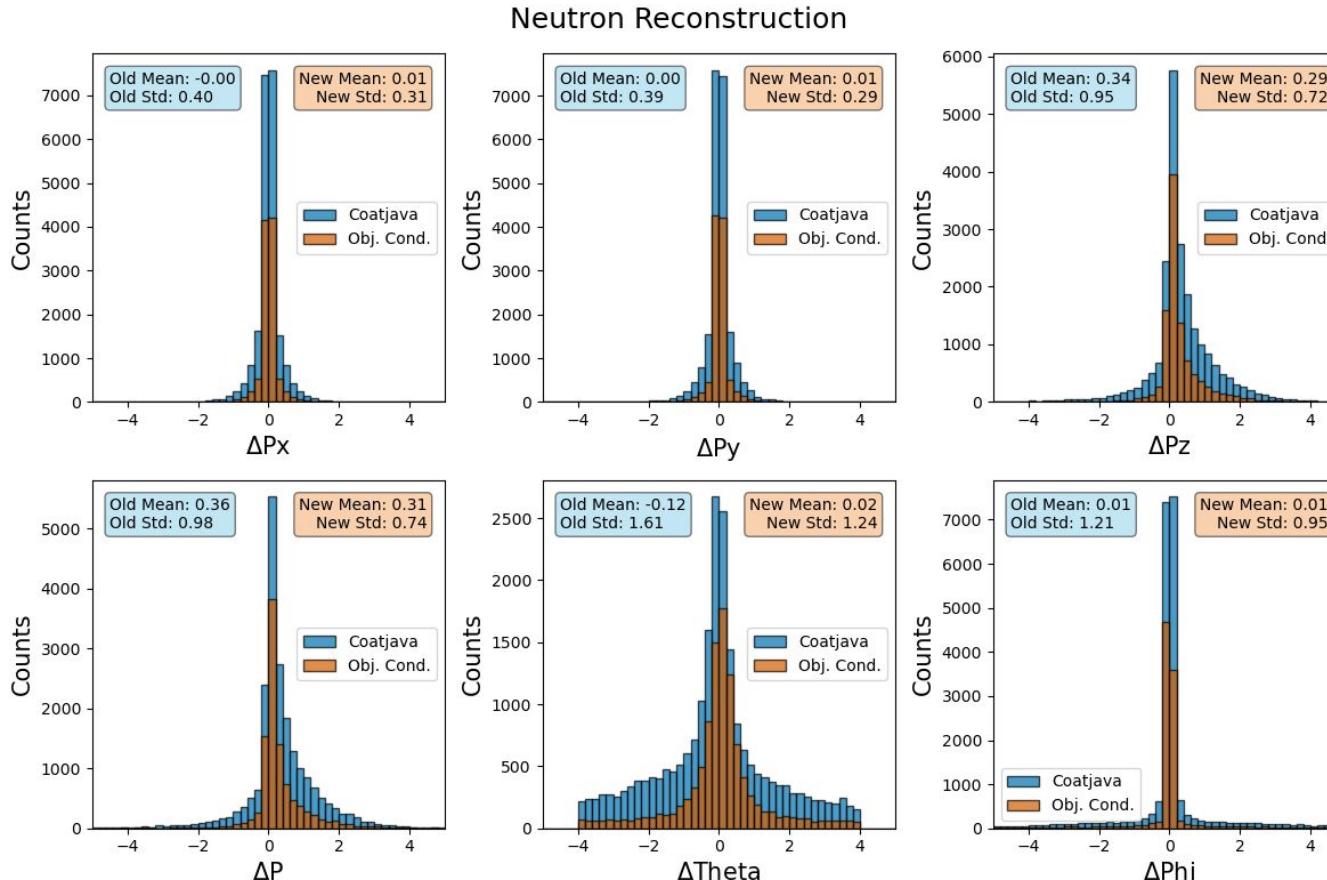
11% less trustworthy photons, but higher trustworthy %-age overall

Neutron Kinematics

- Yields for **Object Condensation** and **Monte Carlo** seem to more closely match for $p < 2 \text{ GeV}$ which corresponds to $\beta < 0.9$
- In **Coatjava** there is an if-statement that assigns $\beta > 0.9$ particles to neutrons if it has no PCAL cluster (hence why there are $p > 2 \text{ GeV}$ neutrons)
- Since **Object Condensation** finds clusters more effectively, this if-statement fails



Neutron Kinematic Resolutions

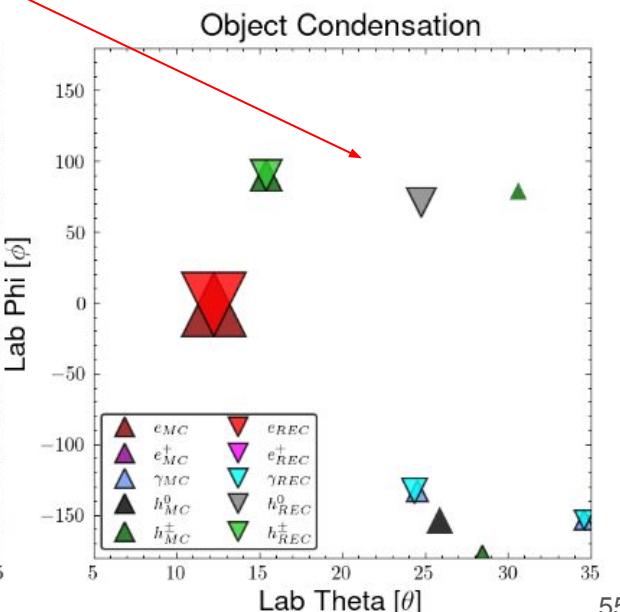
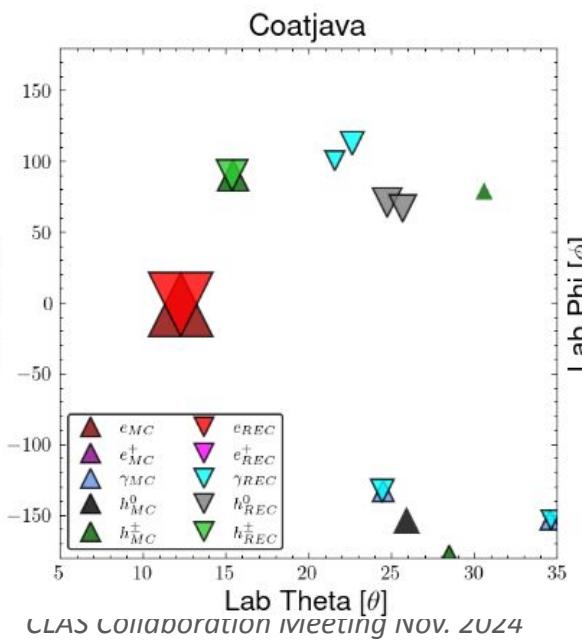
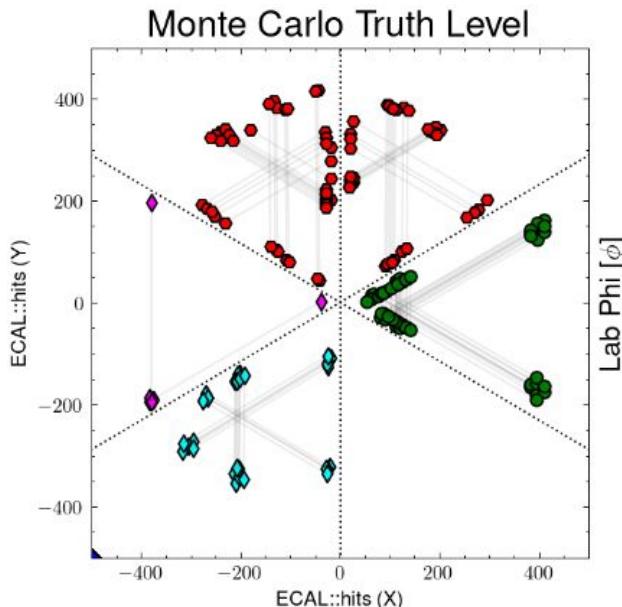


Plots shown for all REC neutrons with a Monte Carlo match

We see the resolution is improved and the mean is essentially the same between **Coatjava** and **Object Condensation**

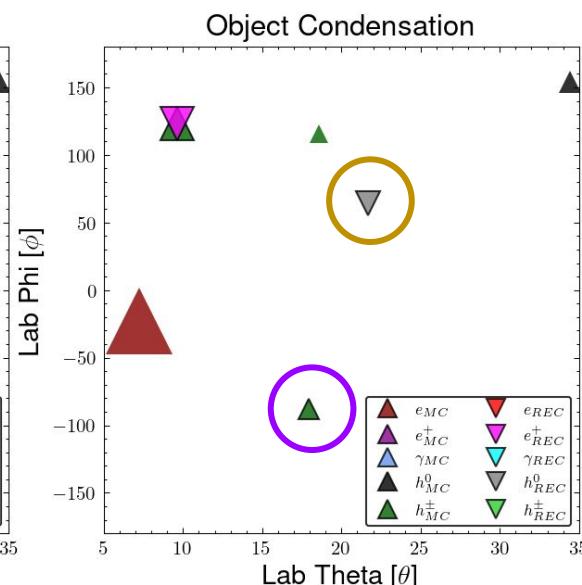
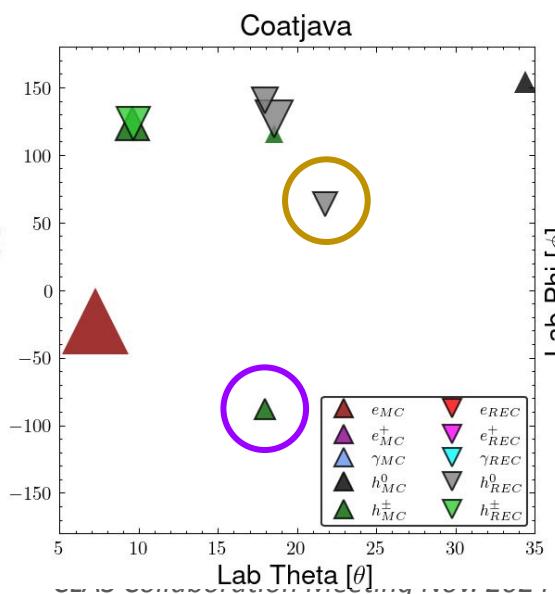
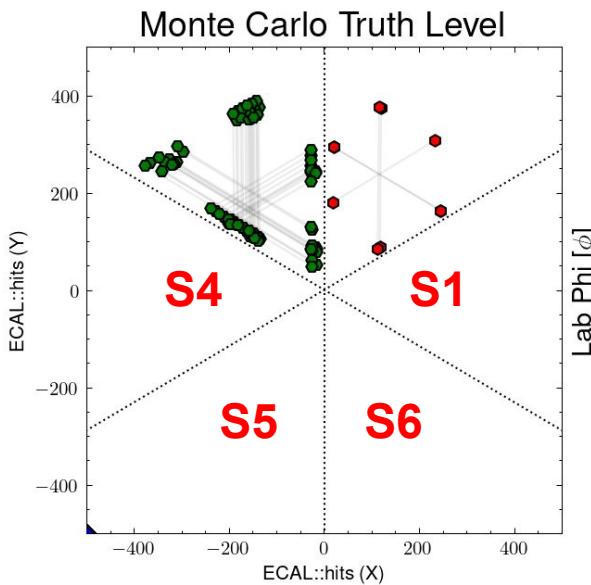
Multi-sector hits

- 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 ($\varphi \sim 60 - 100$ [deg])
 - Both Coatjava and Object Condensation find a stray neutral

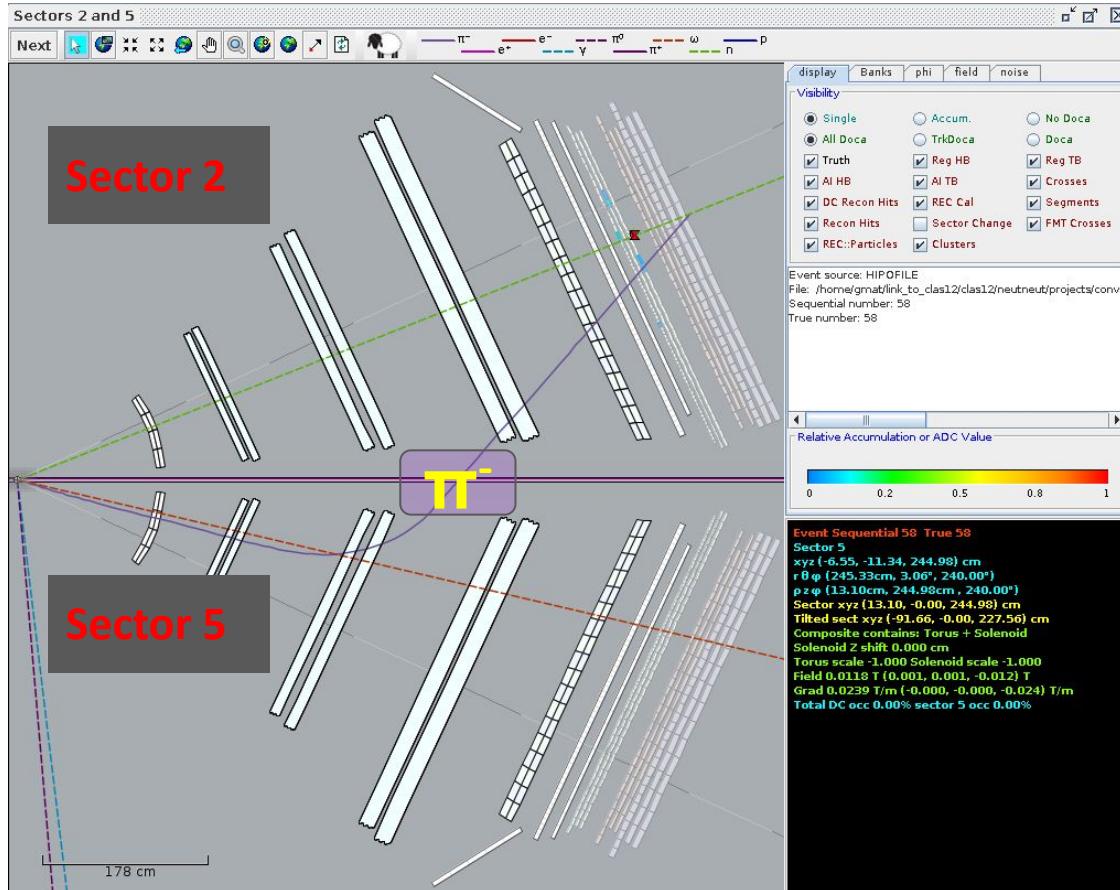


Intersector Tracks

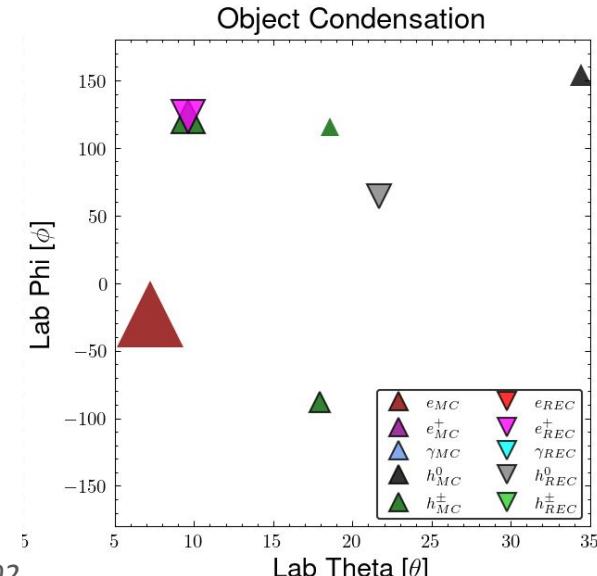
- In this other example, a **Pi- Generated in Sector 5** crosses into sector 2. This pion leaves hits in **Sector 2 which is registered as a Neutron**
 - It *makes sense* that Object Condensation would see the **Sector 2** 3-way intersection as a viable cluster



Intersector Tracks

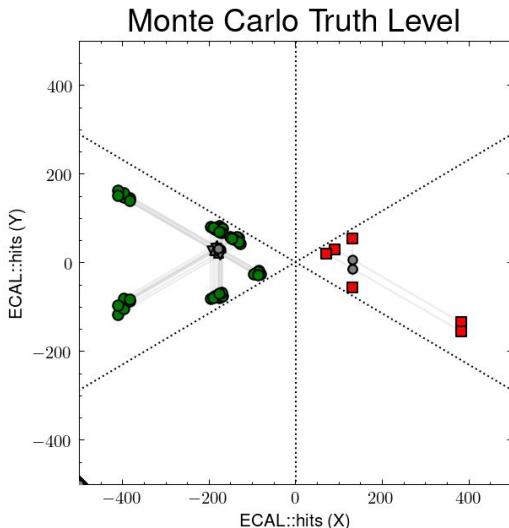


Track might actually leave hits in all 3 DC's (albeit different sectors)
Does the track algorithm account for this in anyway?



Areas to Improve

- The latent space clusters in Object Condensation have the ability to learn features
 - This can be used during training to predict **PID, Cluster X,Y,Z** perhaps more efficiently than just using 3-way intersections
 - This would also allow for non-traditional clusters to be reconstructed, such as 2-way intersections



- On the left is a strip plot from our neutron gun events
 - Since there is no 3-way intersection here, neither Obj. Condensation nor Coatjava will reconstruct a particle
- Can predicting cluster X,Y,Z help resolve this problem?*
- Also, what can we do to add/improve our definition of noise?*

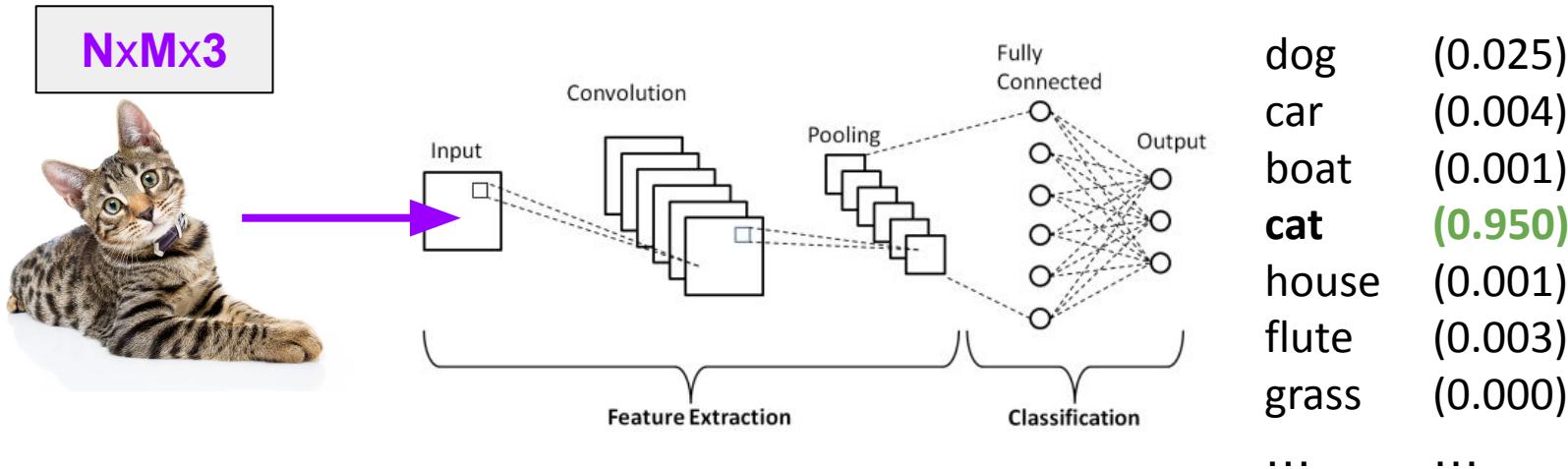
Conclusion/Future

- We identified that an AI trained on REC::Calorimeter/REC::Particle would not alleviate the neutral particle clustering effectively → Turn to the source, train an AI-assisted clustering algorithm
- Coatjava was forked to attach additional truth information to the ECAL::hits bank for training purposes
- A feature extractor utilizing GravNet blocks was used to accumulate abstract nearest neighbor information
- Object Condensation was used to optimize the feature extractor, encouraging it to form clusters in a 2-dimensional latent space that represent Monte Carlo particles
- The ECAL::hits that fall into these clustered regions were processed to calculate a new ECAL::clusters bank
- The ECAL::clusters bank is fed back into the Coatjava pipeline to form a new REC::Particle bank
- We see **3 times improvement** in the trustworthiness of REC::Particle neutrons without sacrificing yields
- Streamlining of collaborator usage/testing of my training/coatjava fork
- Add PID, Px, Py prediction capabilities of neutrals to training
- Begin hyperparameter search to optimize network
- With collaborator approval, consider publishing (might be first AI-assisted calorimeter clustering tested on Monte Carlo in a full reconstruction pipeline) Looking into using it for EIC KLM 2nd detector clustering

TO DO

Extra Slides

Task: Image Classification

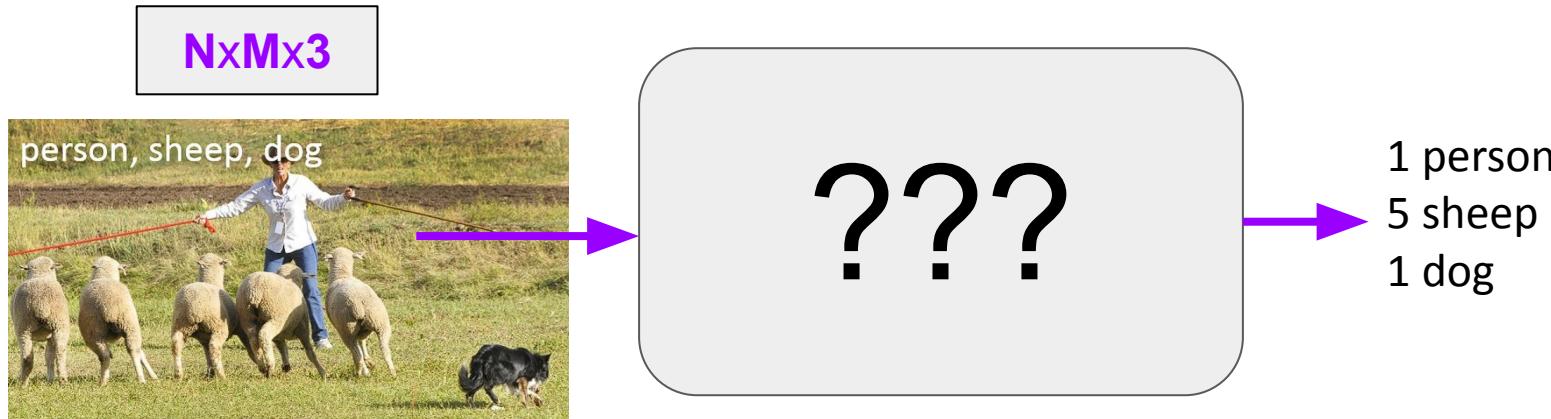


Given... An isolated 'grid' of inputs

Output... A list of prediction scores for each trained category

★**Training**★ is straightforward. ImageNet has ~14 million labeled images with more than 22,000 categories.

Image within Image Classification



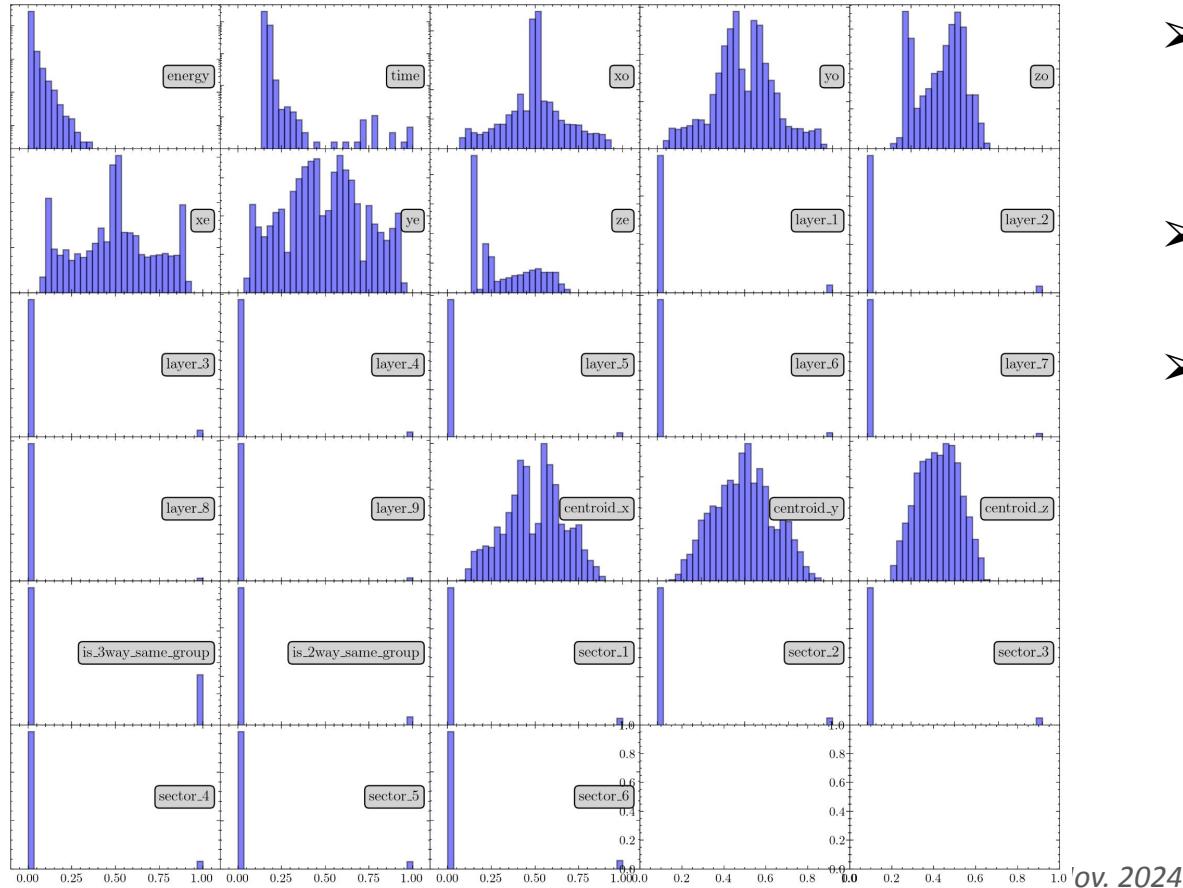
Given... An isolated 'grid' of inputs

Output... A potentially arbitrary number of objects, each classified

★Training is more difficult!★

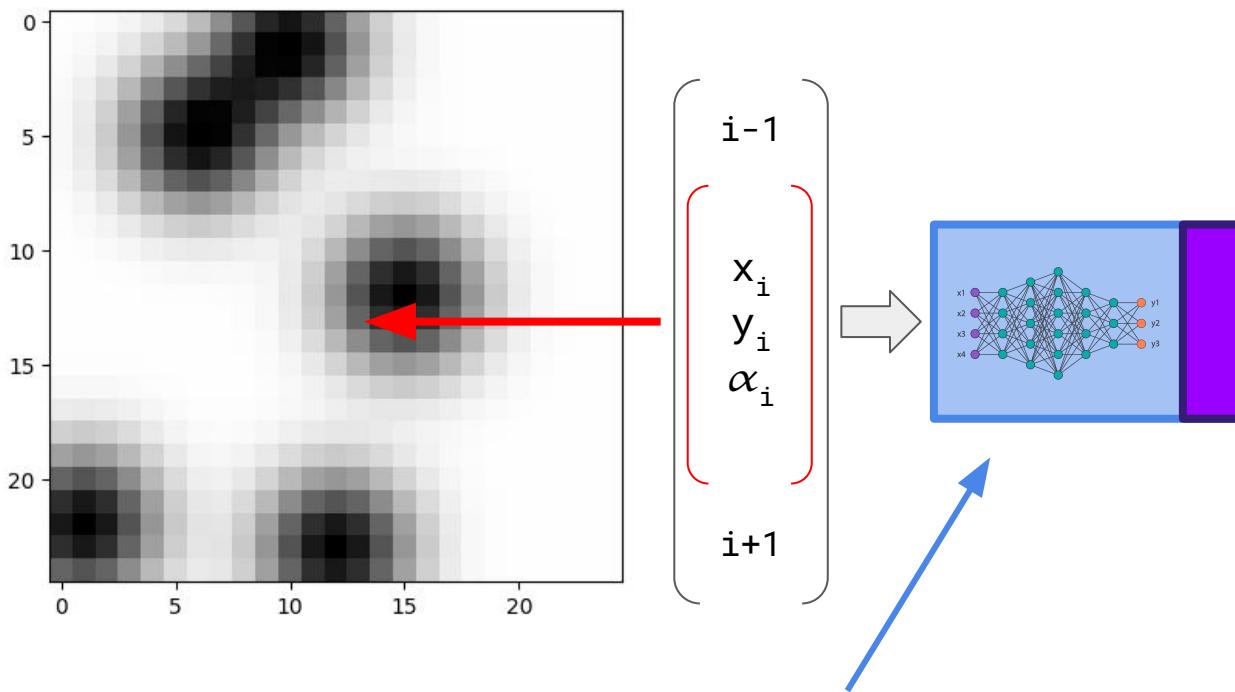
- Cannot easily train for datasets with all possible category combinations
- How would one deal with situations where objects *overlap*?
- The ★Approach★ must be changed (can't do simple CNN)

Machine Learning Input Features



- Shown are the *per strip* input features (normalized to 1)
- Energy & Time are log-scaled
- The one-hot encode for the strip's sector *is not used* because it too strongly correlates with being a unique particle, leading to a quick local minimum during training

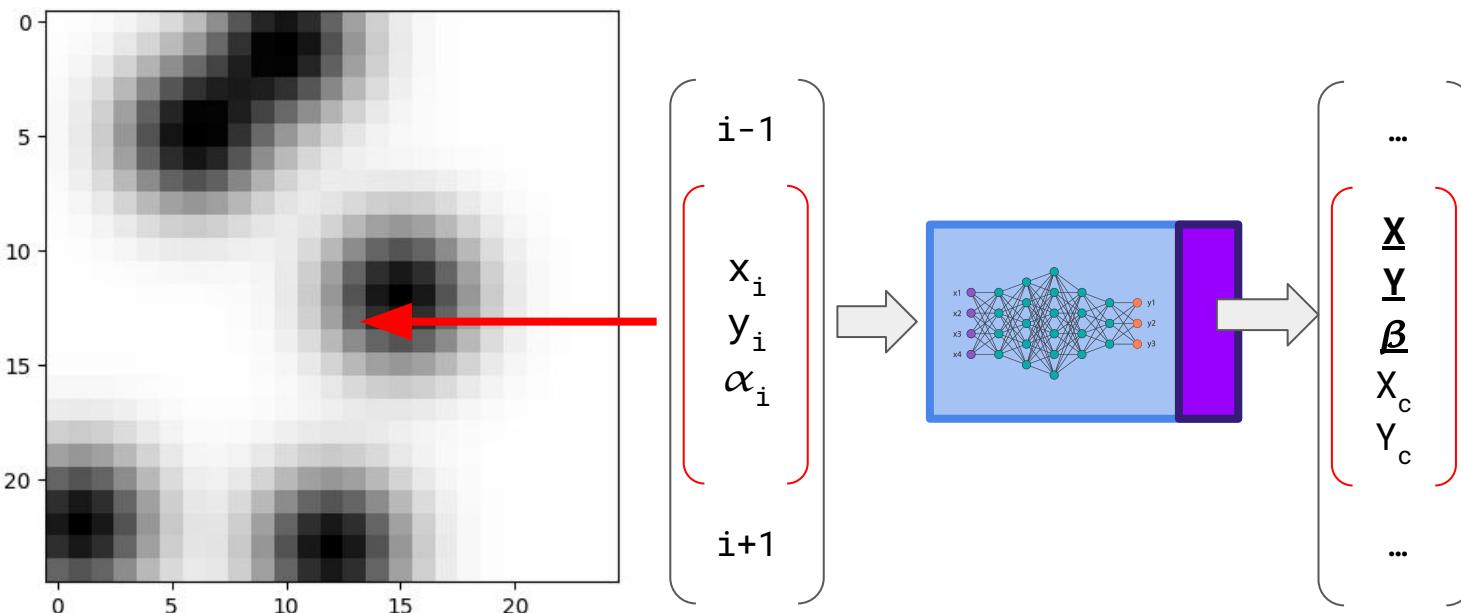
Object Condensation Basics



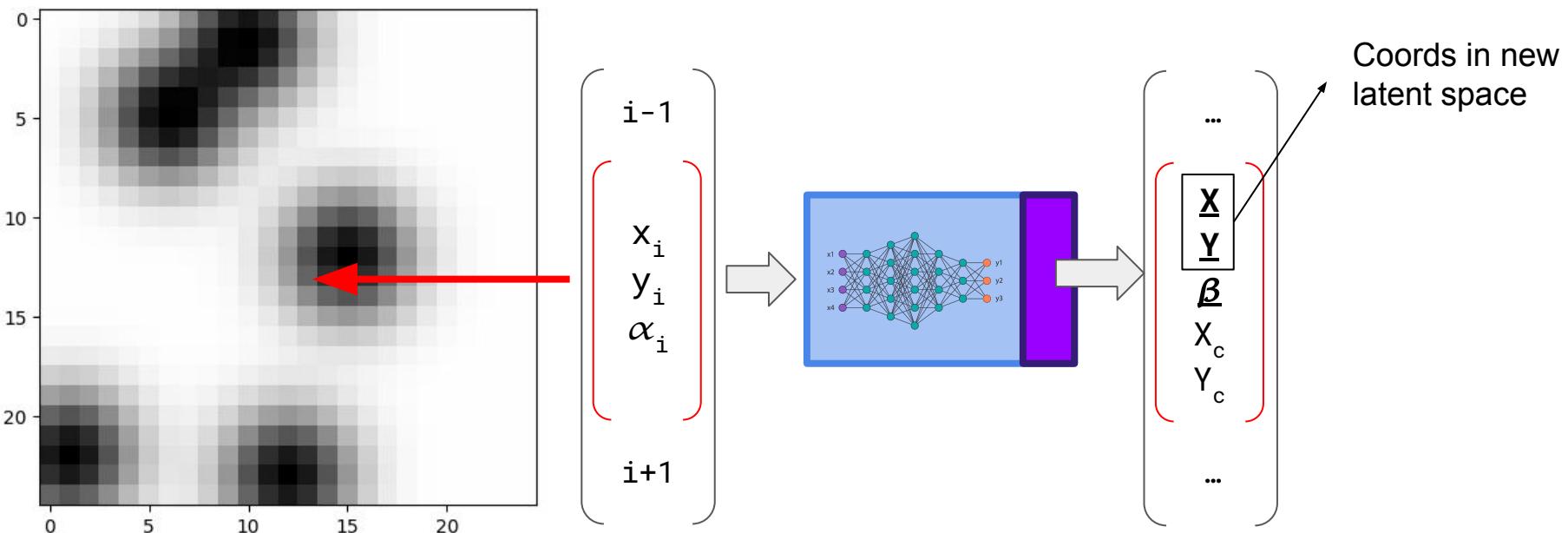
I want my ML model to tell me how many clusters, and their centroids (x_c , y_c)

*Lets see what a **well-trained** model does, then discuss how we even **train** it to perform the task at hand (clustering!)*

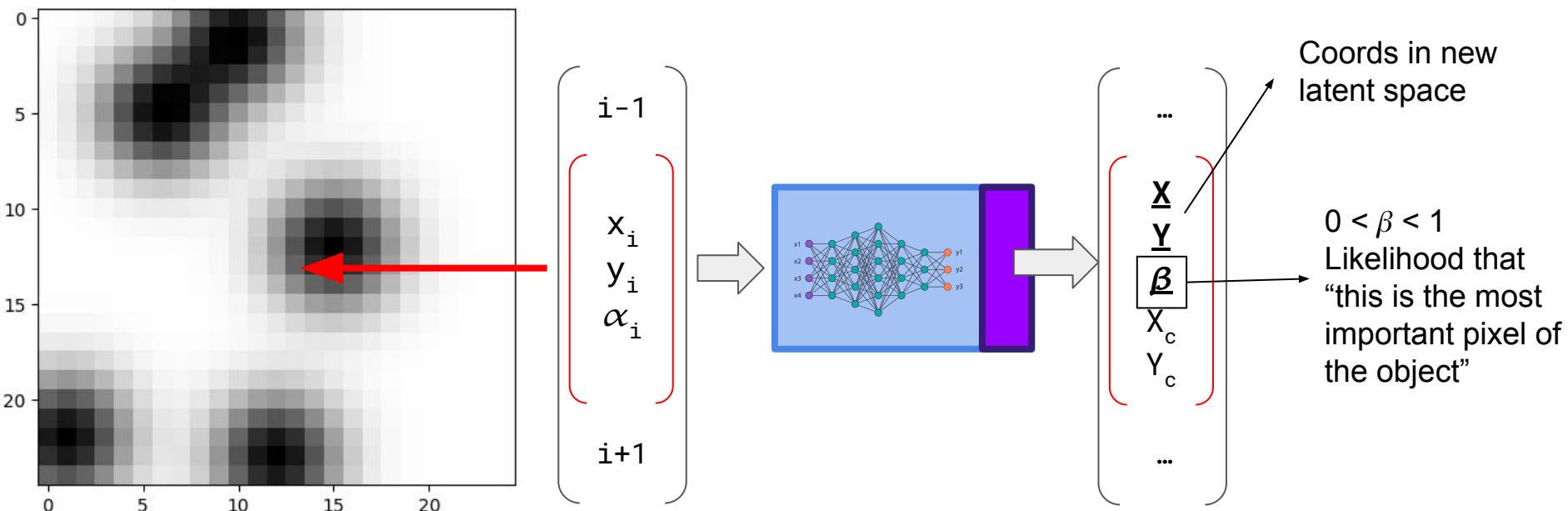
Object Condensation Basics



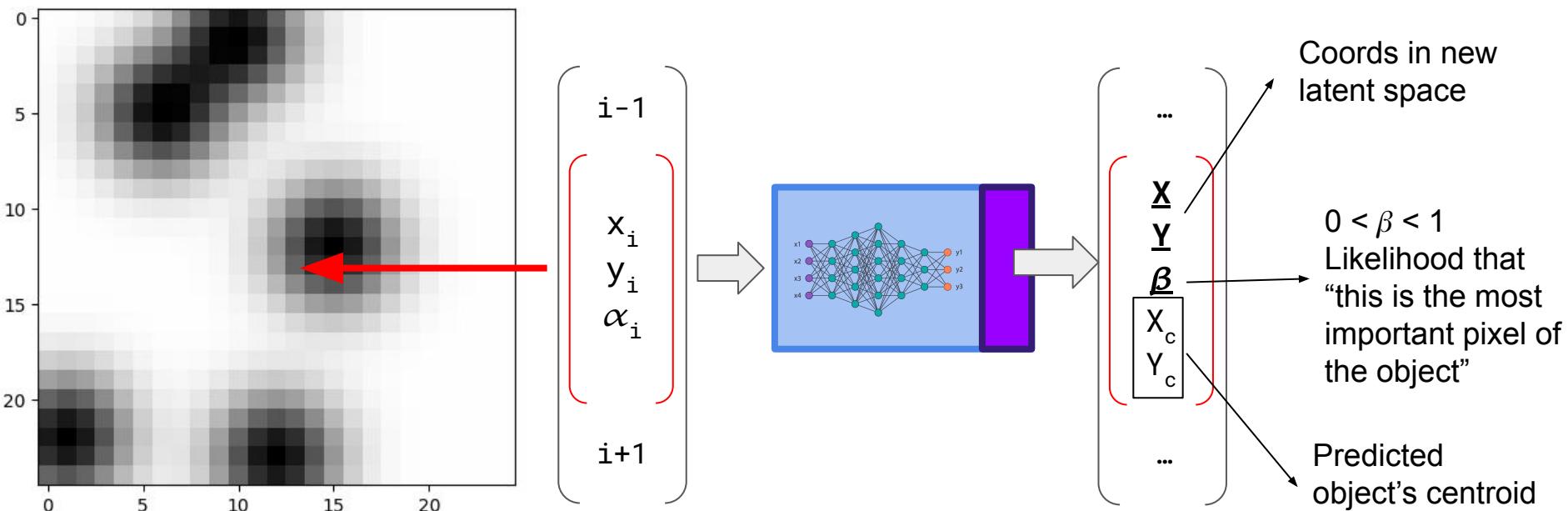
Object Condensation Basics



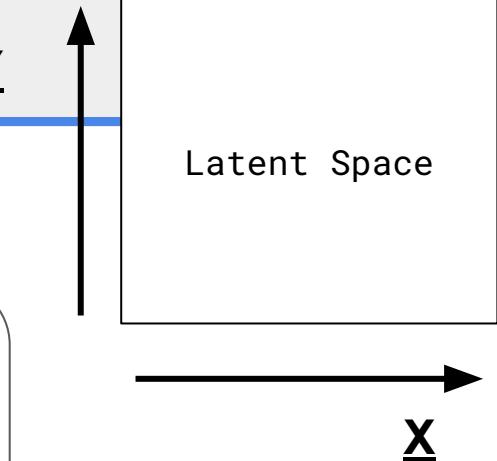
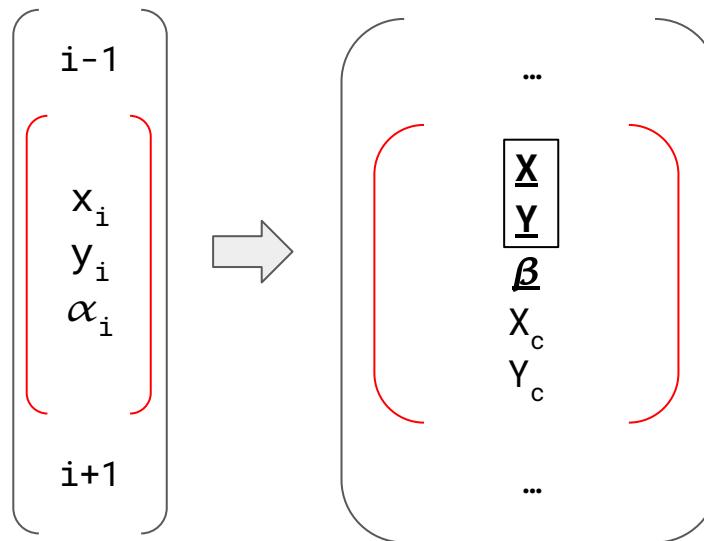
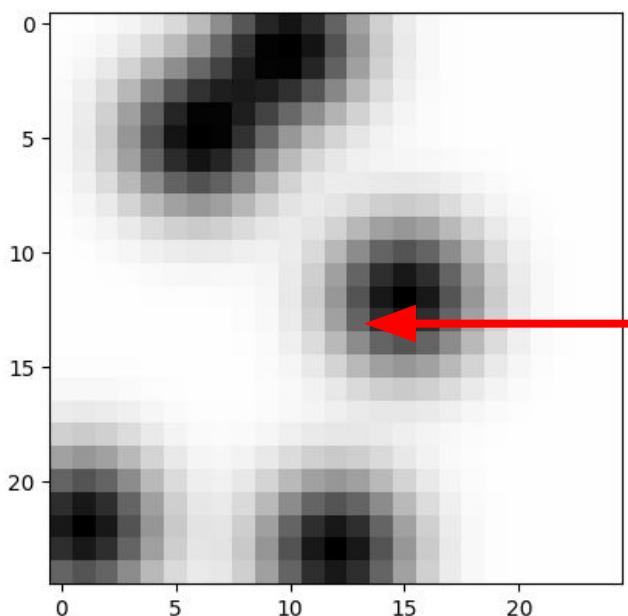
Object Condensation Basics



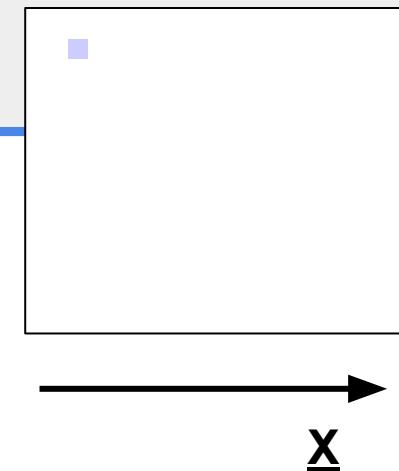
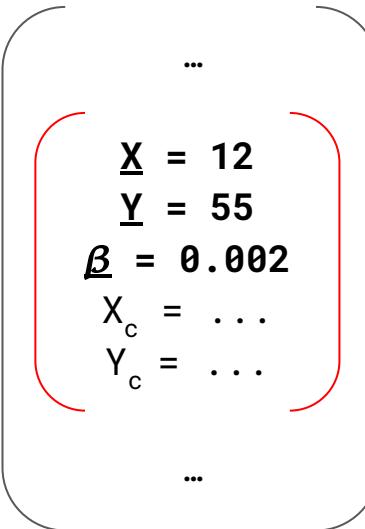
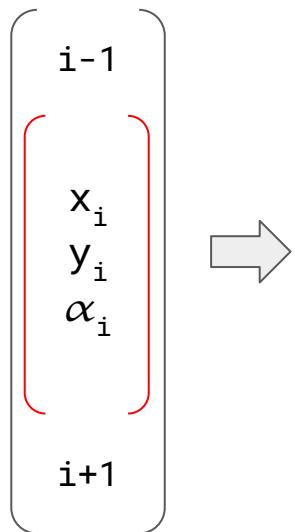
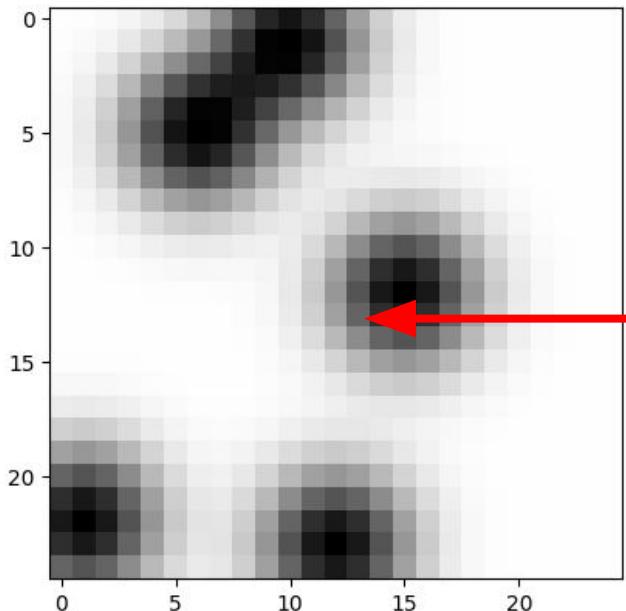
Object Condensation Basics



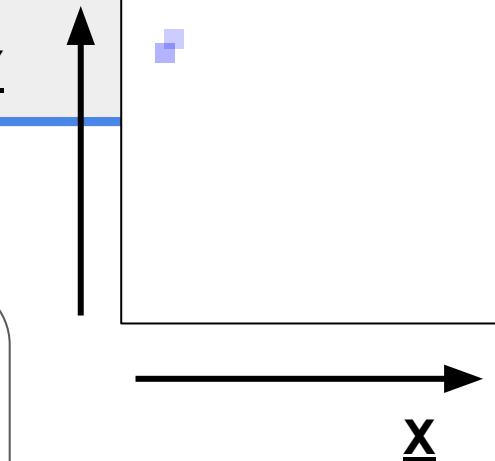
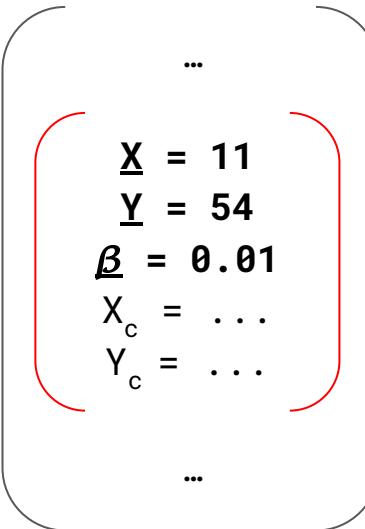
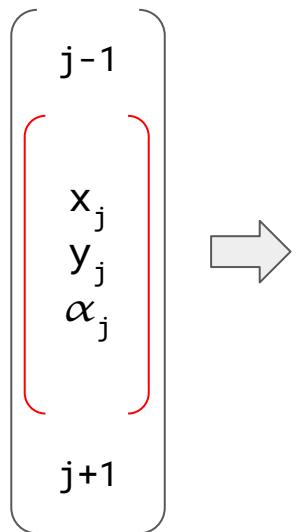
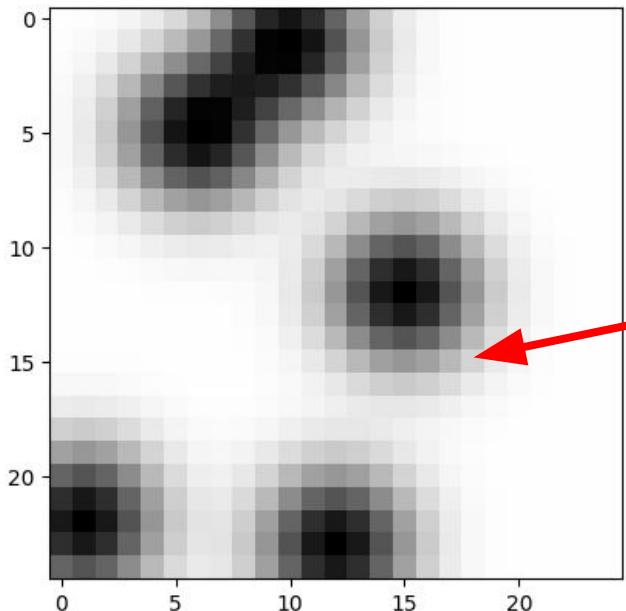
Object Condensation Basics



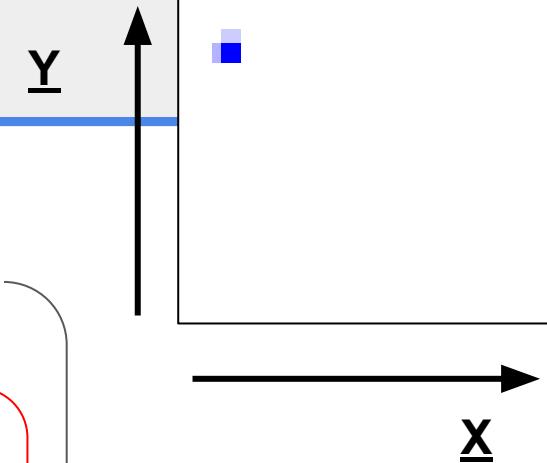
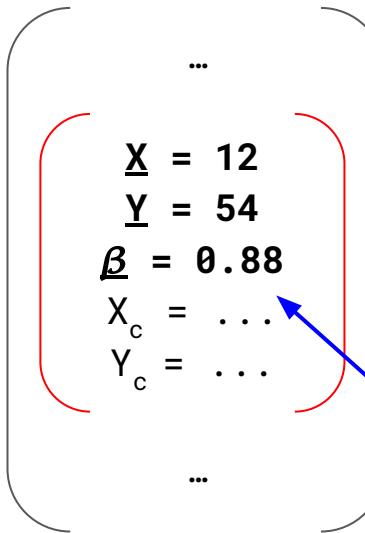
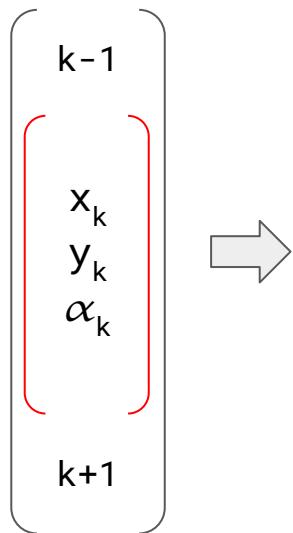
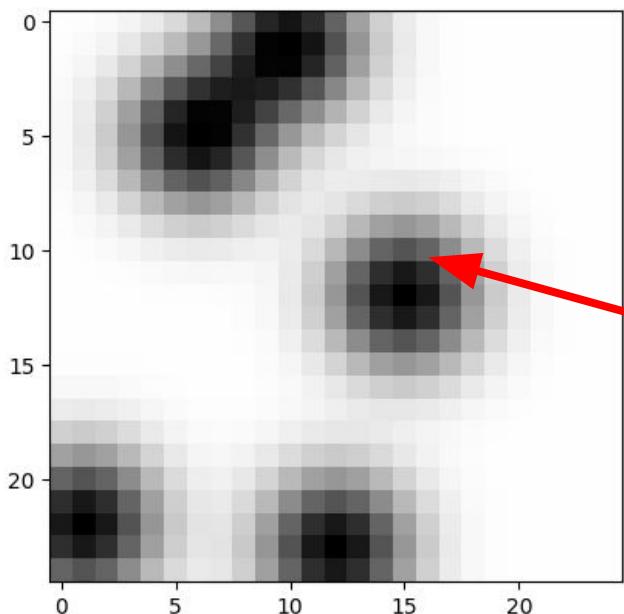
Object Condensation Basics



Object Condensation Basics

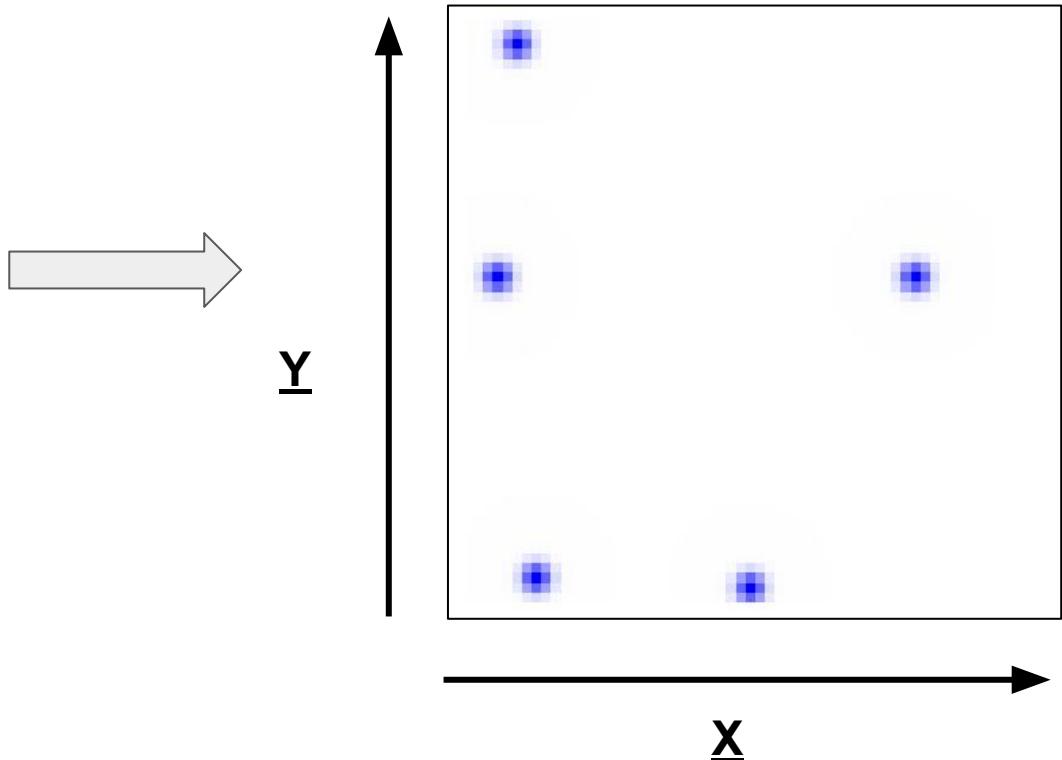
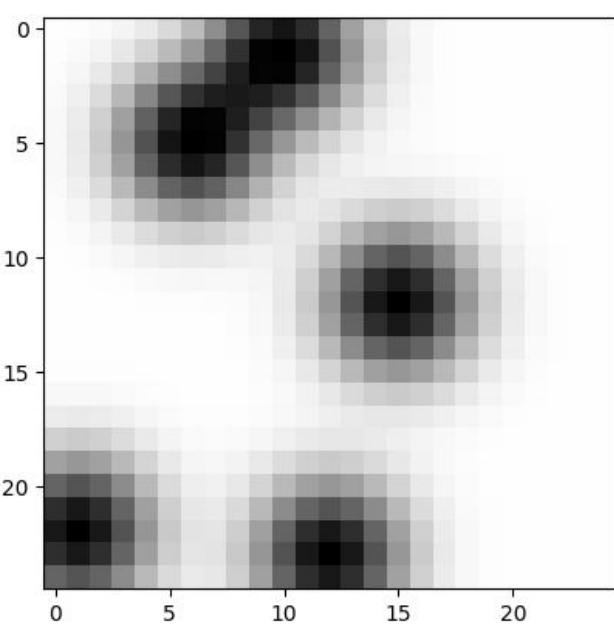


Object Condensation Basics



High β implies the model thinks this point is very important!

Object Condensation Basics

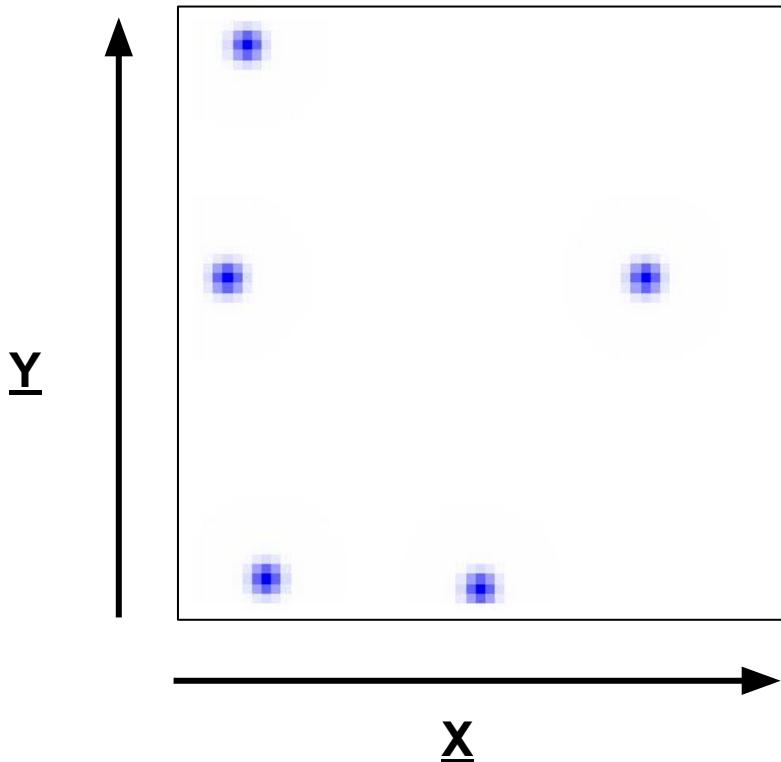


Input: Set of $[x, y, \alpha]$ (625x3)

Output: Set of $[\underline{X}, \underline{Y}, \beta, x_c, y_c]$ (625x5)

Model is trained to make 1 bright β per object

Object Condensation Basics



Solution becomes much simpler to picture...

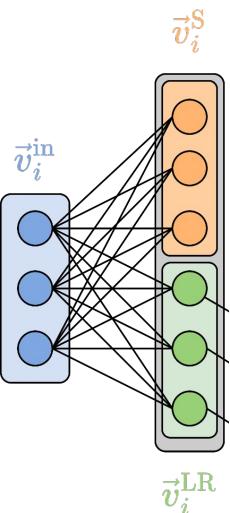
... threshold away dim pixels ($\beta < 0.8$) ...

... count the # pixels remaining ...

... read off their predicted x_c and y_c ...

GravNet Layer

v_i^{in} → Strip i's Input
vector to GravNet

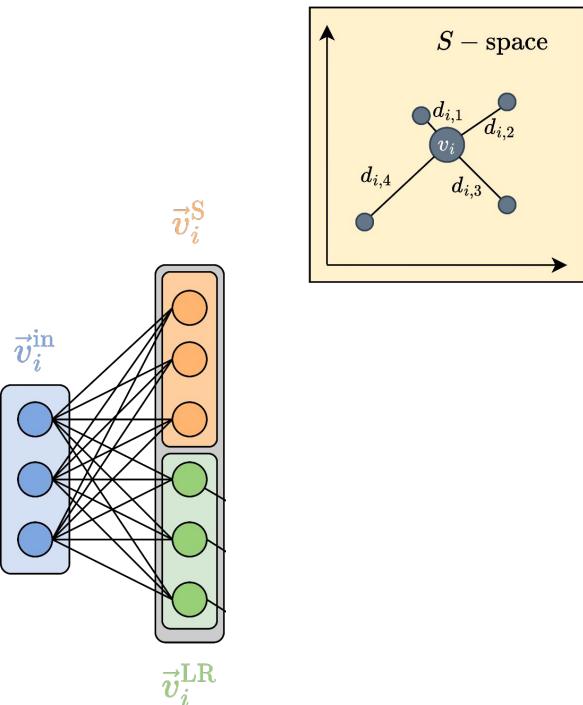


Hyperparameters
S-dims, # Learned Features

Procedure (for each strip)

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

GravNet Layer



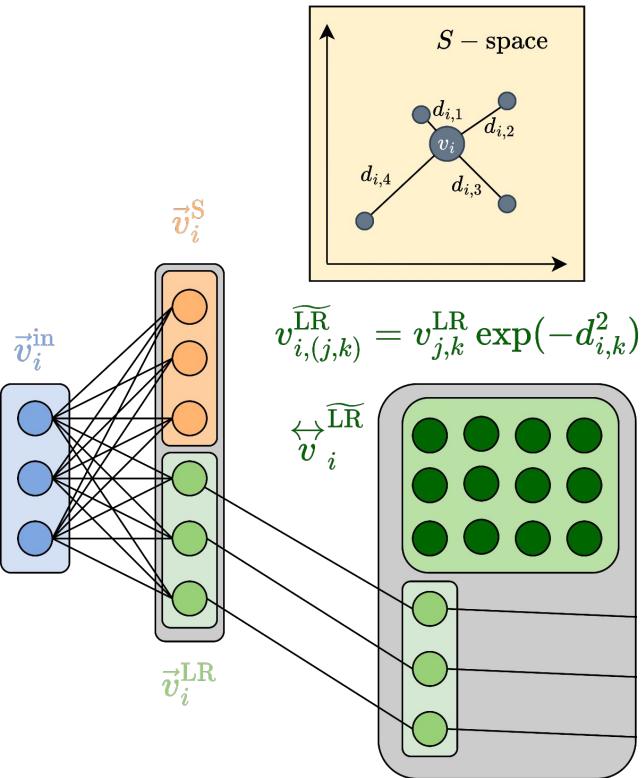
Hyperparameters

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

Procedure (for each strip)

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

GravNet Layer

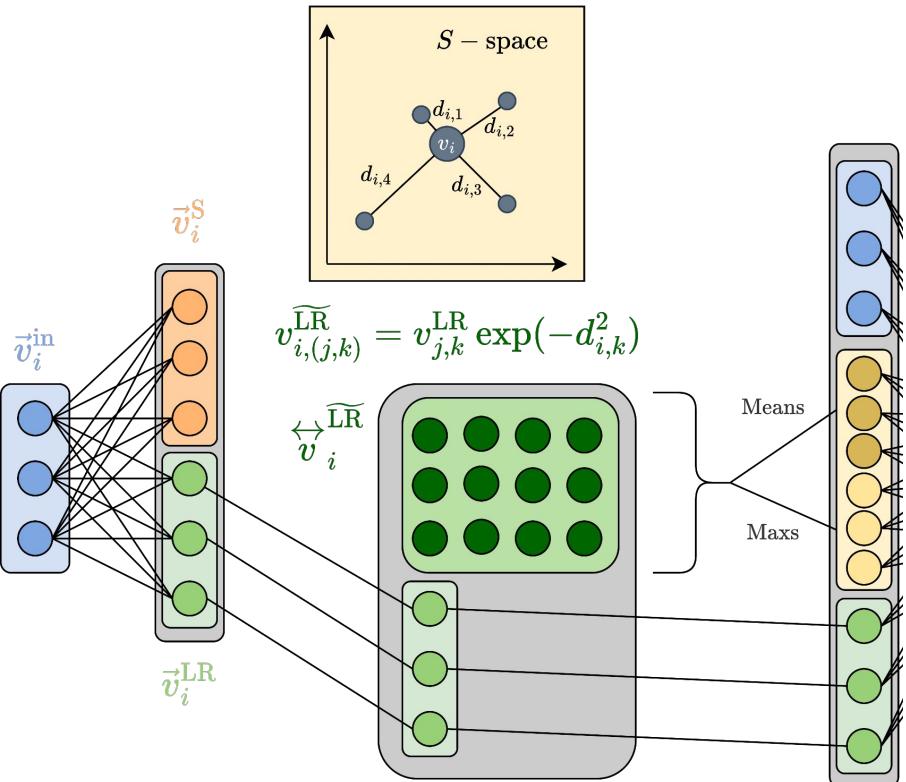


Hyperparameters
S-dims, # Learned Features, # S-Neighbors

Procedure (for each strip)

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

GravNet Layer

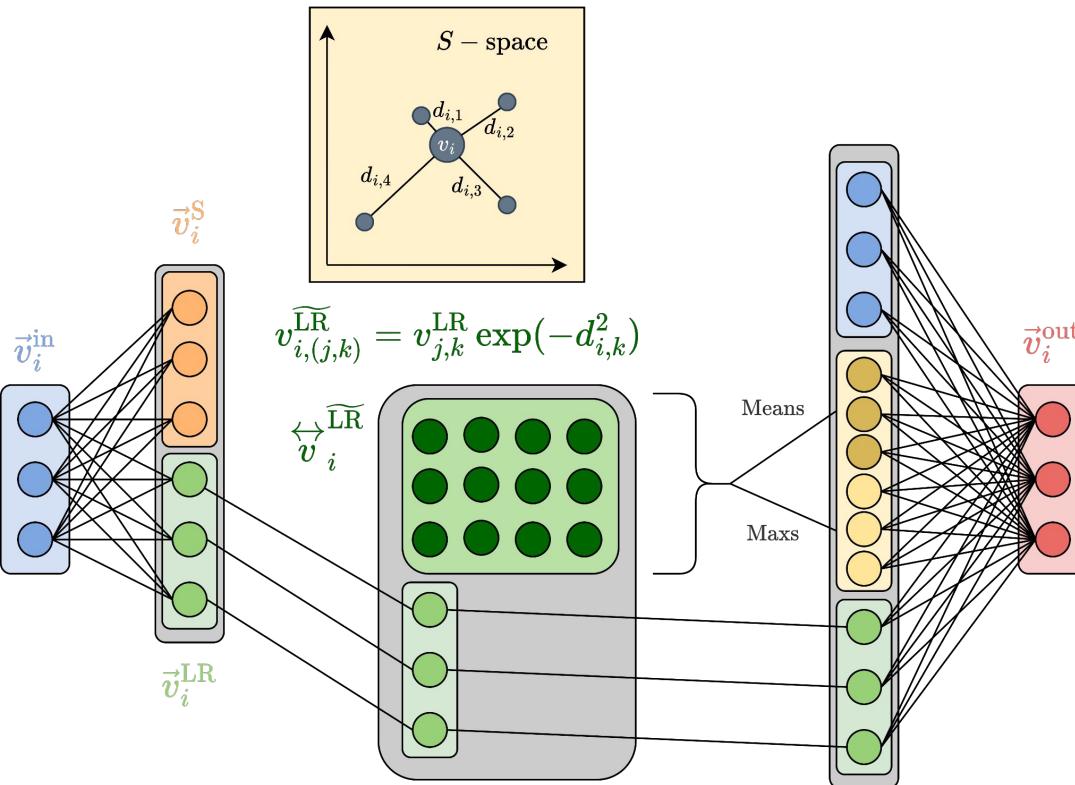


Hyperparameters
 # S-dims, # Learned Features, # S-Neighbors

Procedure (for each strip)

1. A DNN produces a set of coordinates in **S-space** and hidden features \vec{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 \vec{v}^{in} , \vec{v}^{LR} and the mean(+max) of \vec{v}^{LR}

GravNet Layer



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

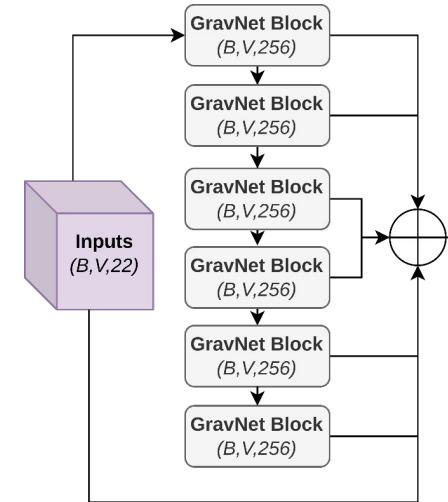
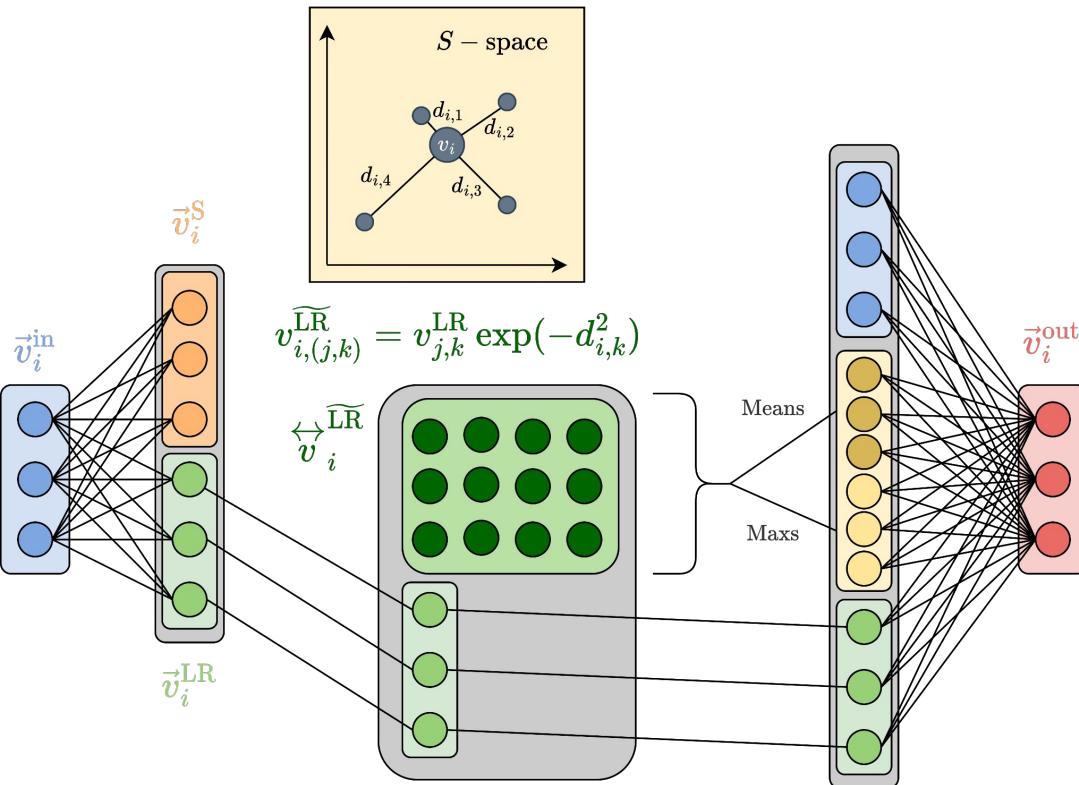
Procedure (for each strip)

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 \mathbf{v}^{in} , \mathbf{v}^{LR} and the mean(+)max of \mathbf{v}^{LR}
5. DNN the final result to a new output vector \mathbf{v}^{out}

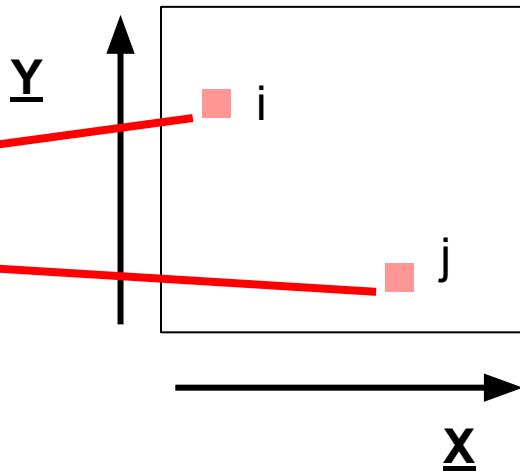
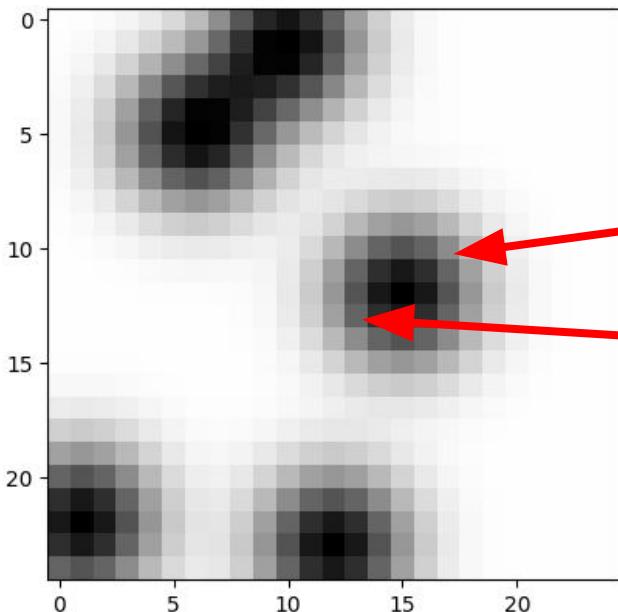
GravNet Layer

The 1st v^{in} to the GravNet blocks have
no neighbor info

Each output of the block provides
higher level nearest neighbor features



Demand A: Points group together

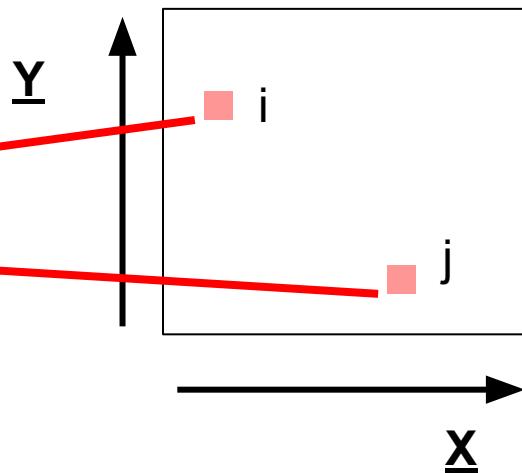
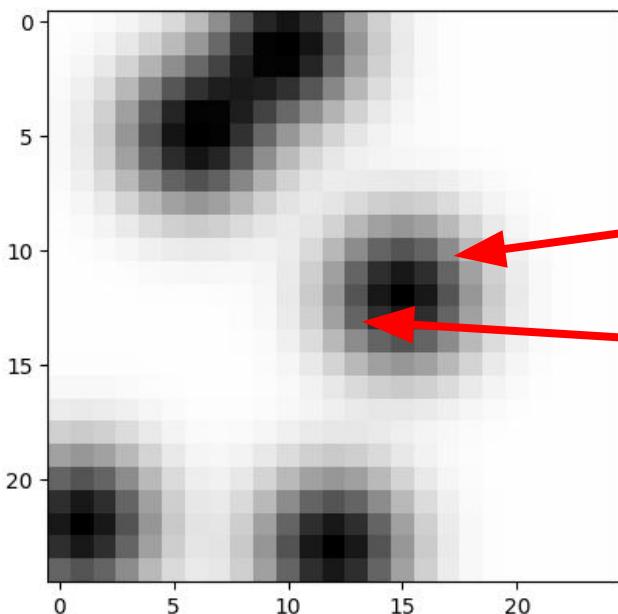


During training, we know these points (i,j) come from the same object...

...we want them to attract to one another in the latent space...

Before training, X & Y are random

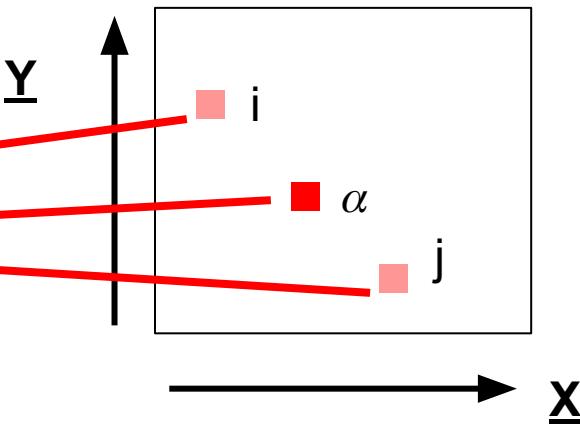
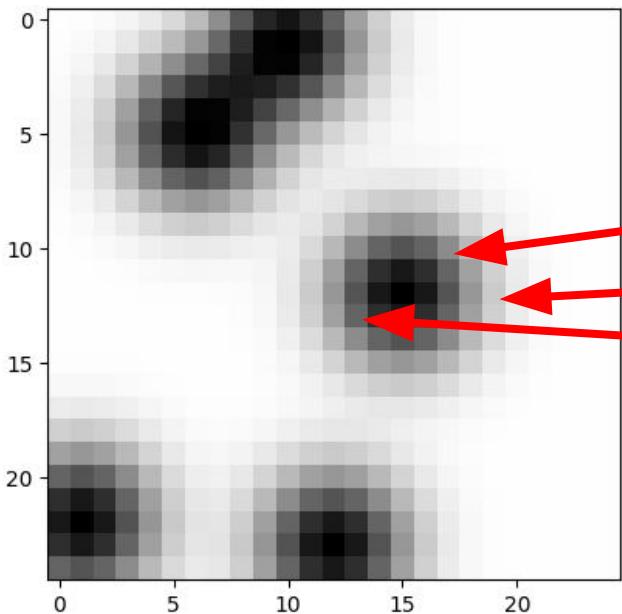
Demand A: Points group together



If we want to encourage our frontend model to make these $\underline{X}, \underline{Y}$ closer, have it minimize some **attractive potential**

How to define?

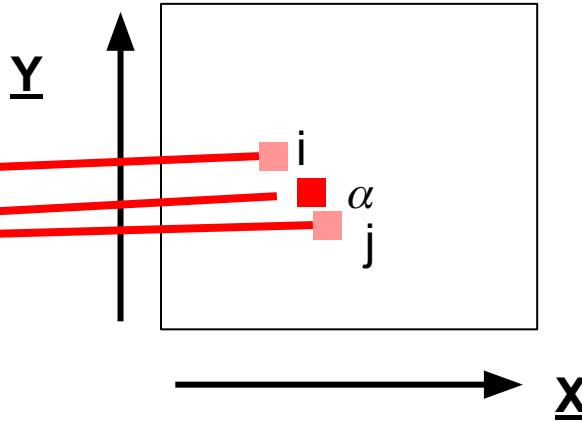
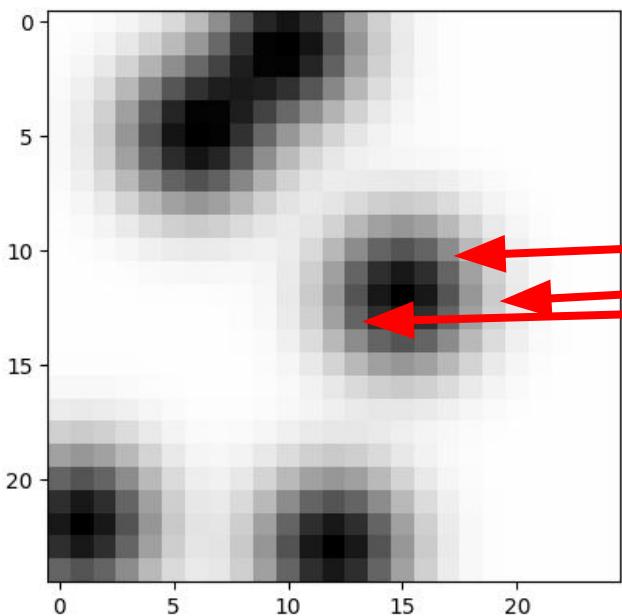
Demand A: Points group together



Punish the model if the pixels are “far away” from the *brightest* (highest β) pixel

Label that brightest pixel as α

Demand A: Points group together

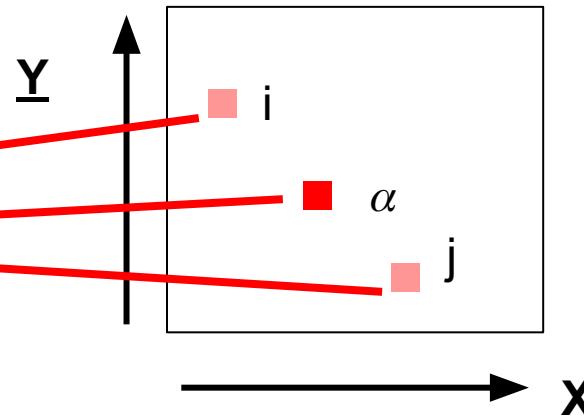
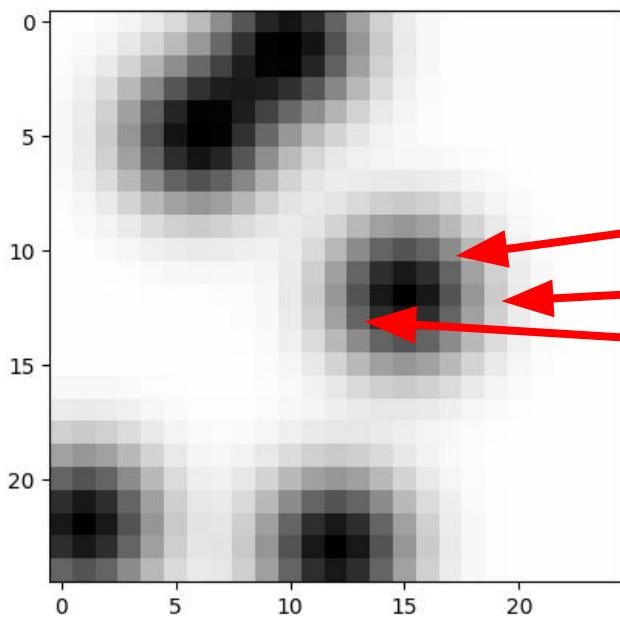


Less punishment !!!

Punish the model if the pixels are “far away” from the *brightest* (highest β) pixel

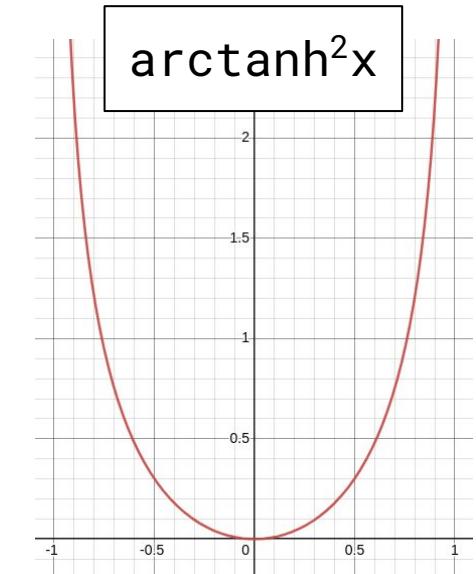
Label that brightest pixel as α

Demand A: Points group together



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

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



$i \rightarrow$ unique id for each pixel $[0, 1, \dots, 624]$

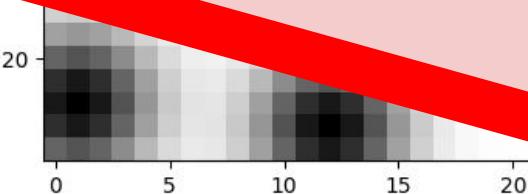
$\alpha \rightarrow$ per object k , id of pixel with highest charge

$k \rightarrow$ unique id of the object $[0, 1, 2, 3, 4]$

$x \rightarrow$ latent space coordinates

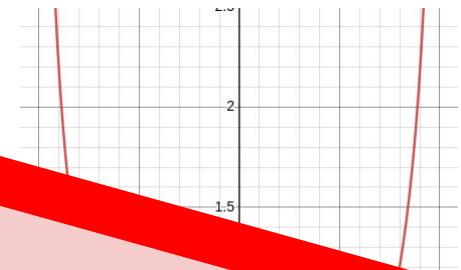
Demand A: Points group together

Attract an object's pixels towards its
brightest (highest β) member

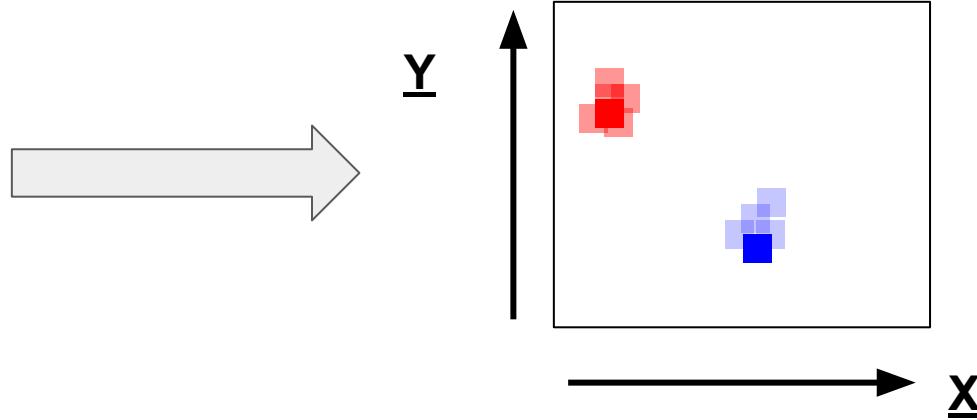
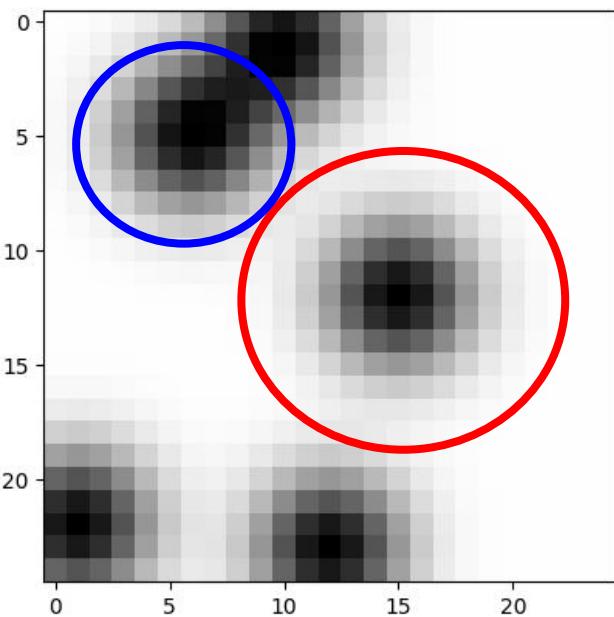


$$\check{V}_k(x) = \|x - x_\alpha\|^2 q_{\alpha k},$$

charge
 $k \rightarrow$ unique id or...
 $x \rightarrow$ latent space coordinate



Demand A: Points group together

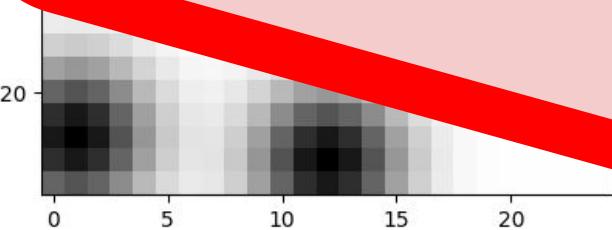


$$L_V = \frac{1}{N} \sum_{j=1}^N q_j \sum_{k=1}^K \left(M_{jk} \check{V}_k(x_j) \right)$$

In words... for each pixel (j) we calculate its potential w.r.t each object (k). If that pixel (j) is in object (k), punish (increase the **Loss**) if (j) is far away ... $M_{jk} = 1$ if (j) is in object (k), else 0

Demand A: Points group together

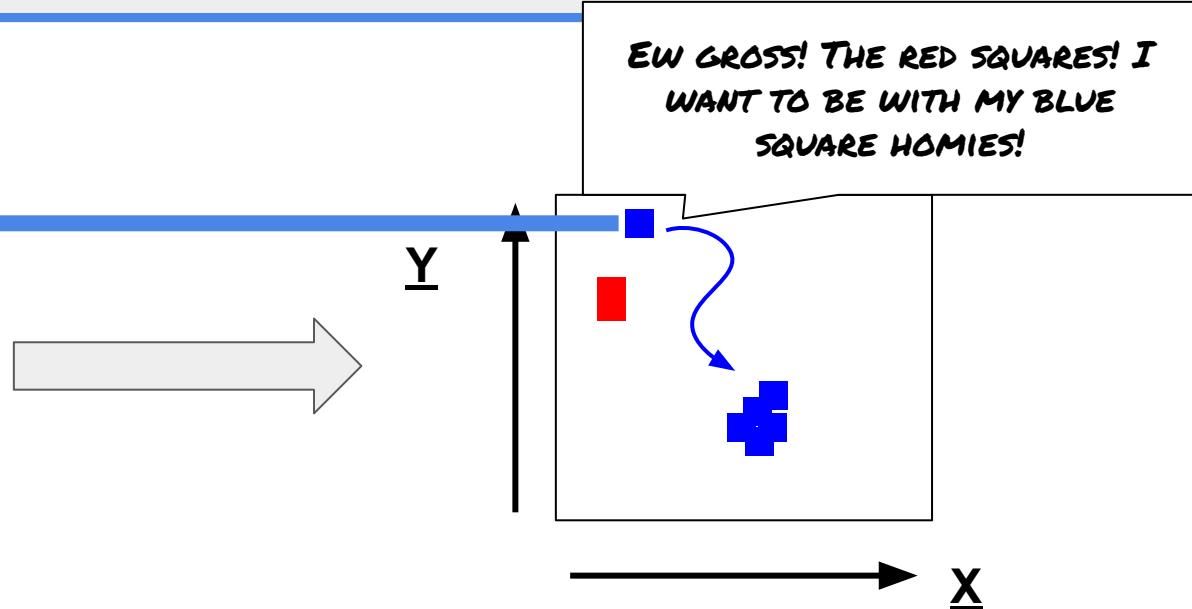
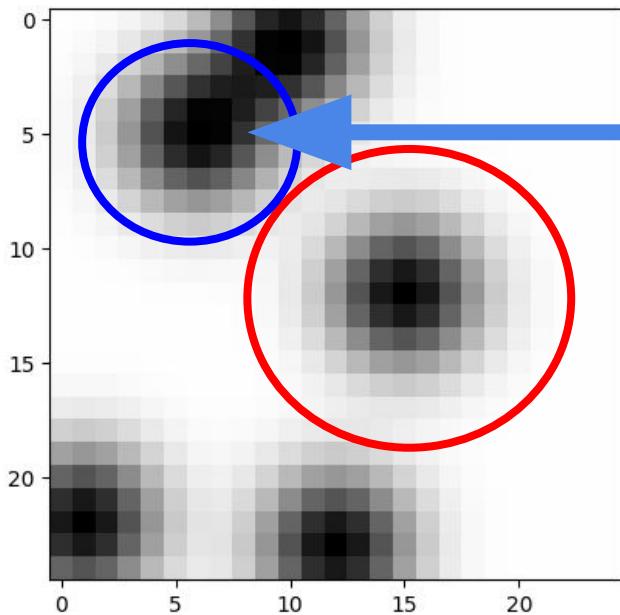
We must also “scare” away pixels from different objects so that they cluster elsewhere...



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

In words...
potential w.r.t each obj
object (k) , punish (increase it)
away ... $M_{jk} = 1$ if (j) is in object (k), etc.

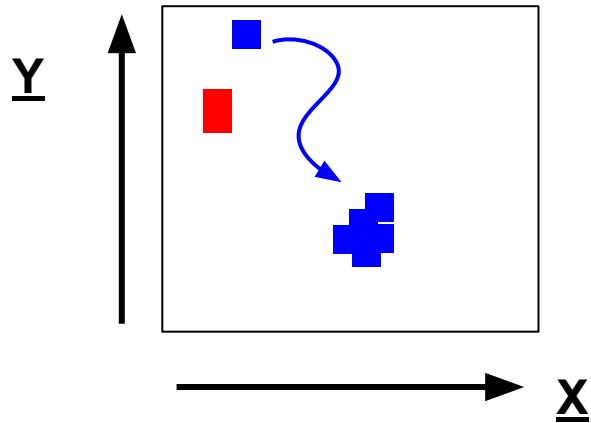
Demand B: Points group separately



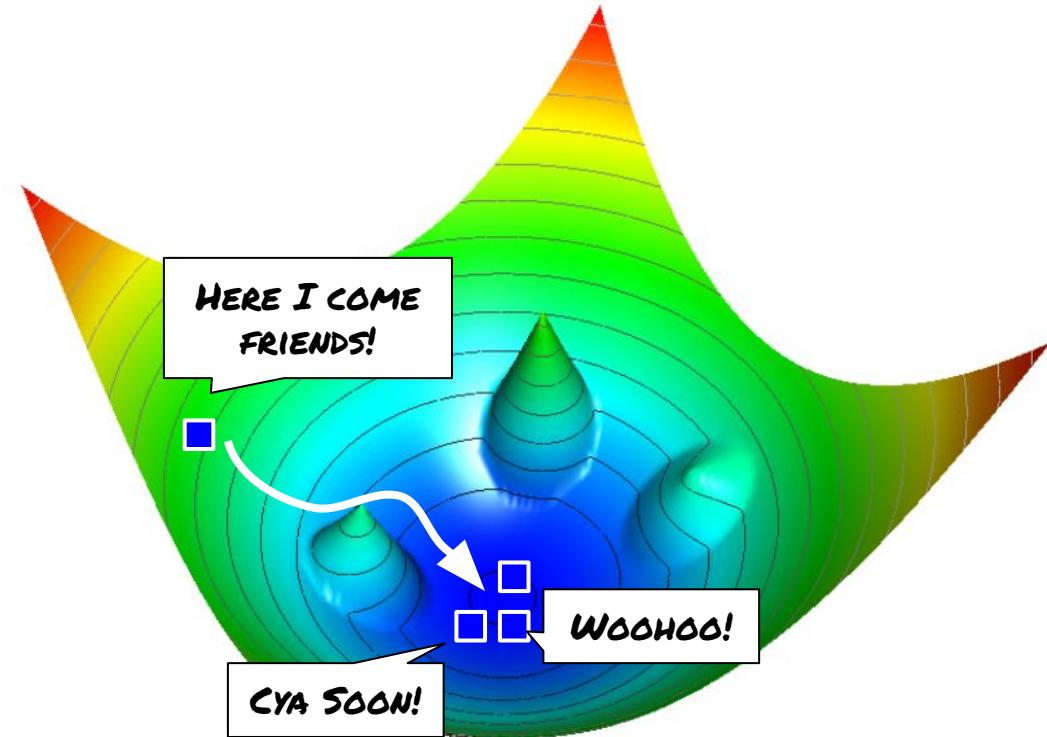
Punish the model if this repulsive term is large. Occurs when pixel (j) is near objects it is not affiliated with...

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

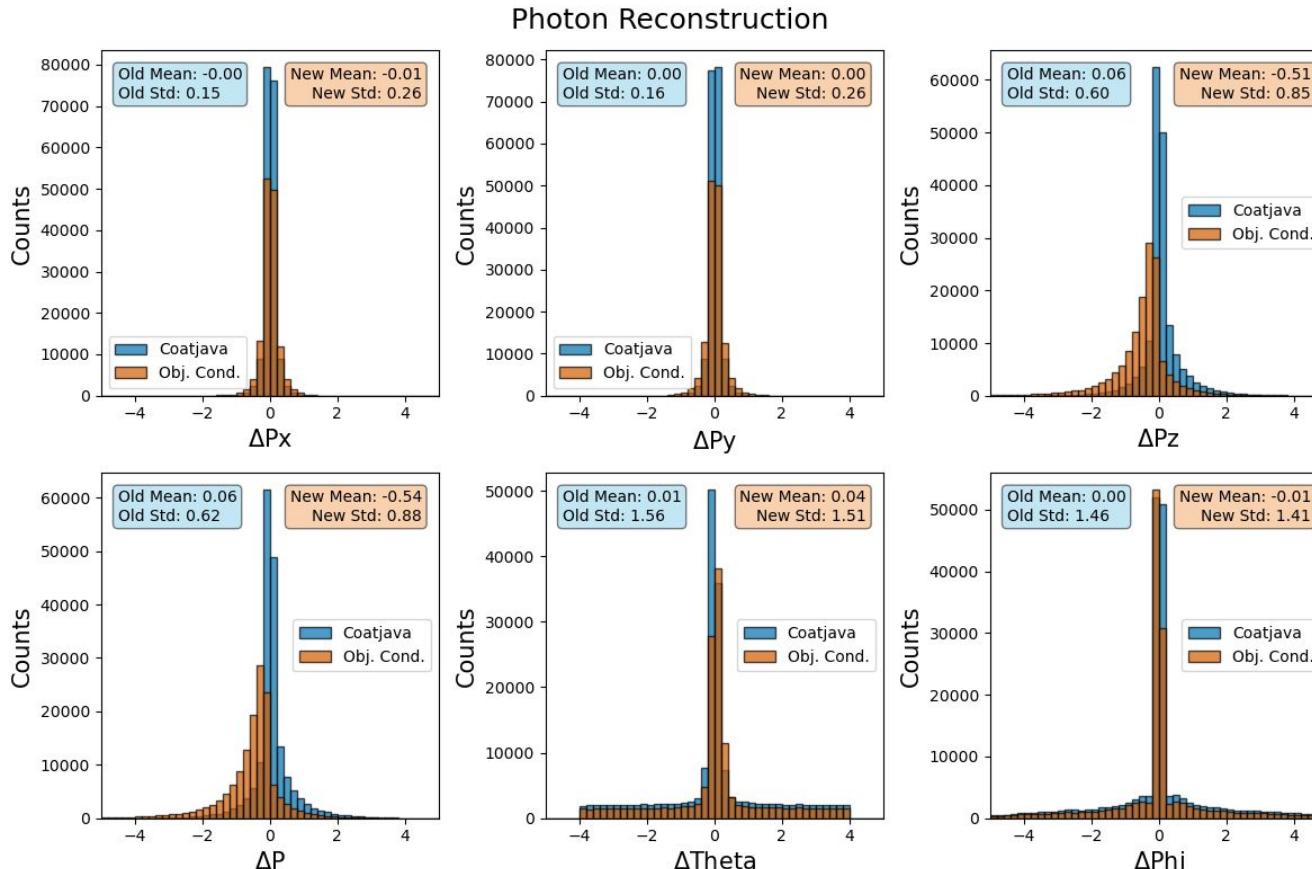
Demand B: Points group separately



(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)



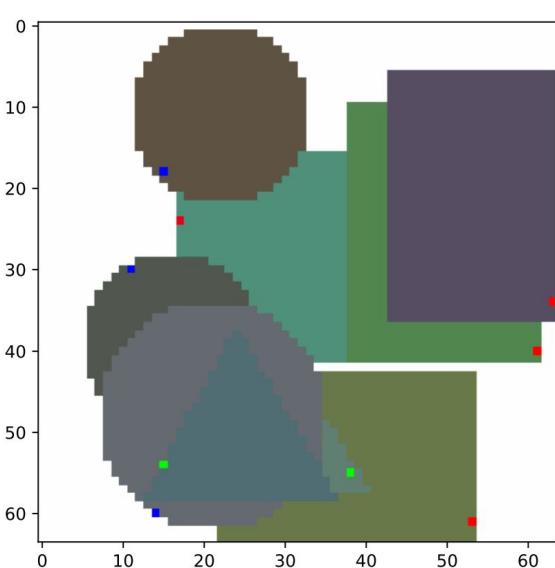
Photon Kinematic Resolutions



Plots shown for all REC photons with a Monte Carlo match

We see the resolution is good apart from the P_z and P which has a documented parallax fix in the [CLAS12 Electromagnetic Calorimeter](#) paper that I have not implemented (has to do with the cluster's centroid-z coordinate)

Interesting Example

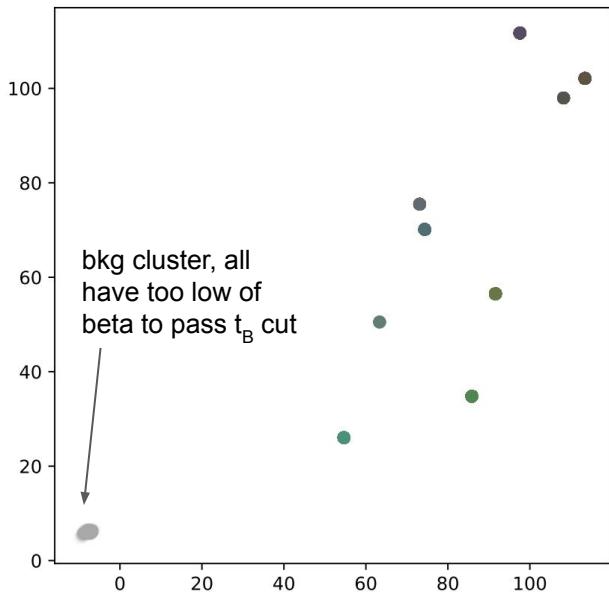


9 objects



Classifier

9 clusters



Input Space

Latent Space