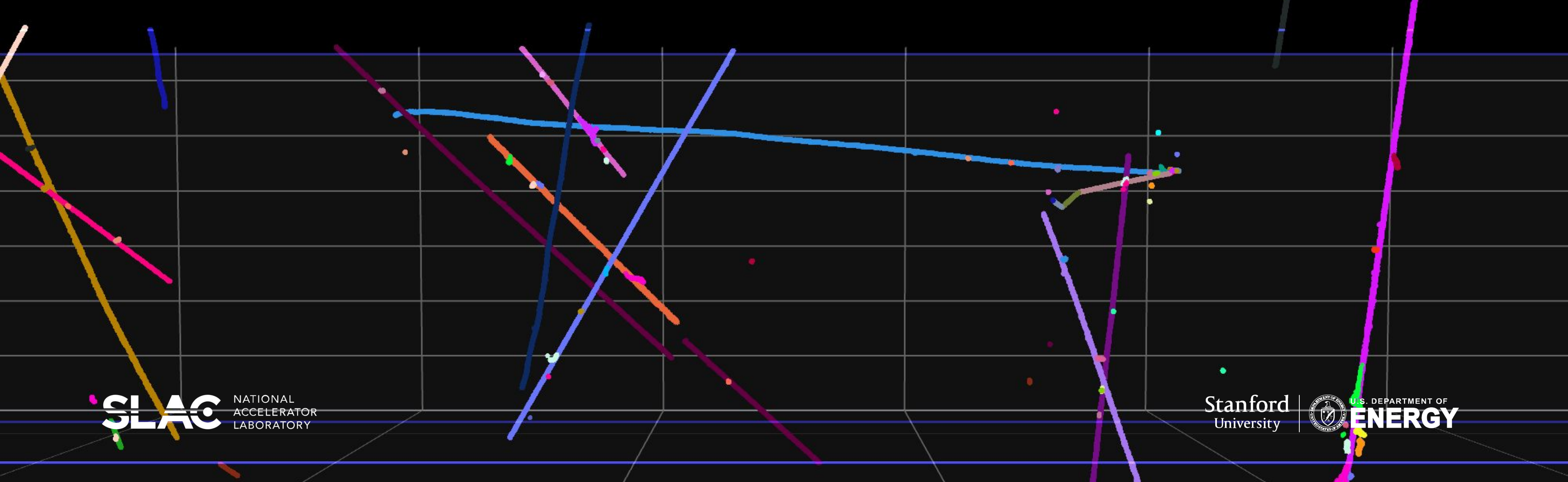
Scalable, End-to-end, ML-Based Reconstruction Chain in LArTPCs

ICARUS Machine Learning Workshop, CSU

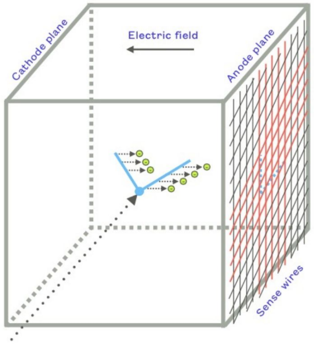
François Drielsma, Laura Dominé, Yeon-Jae Jwa, Dae Heun Koh, Kazu Terao **(SLAC)**

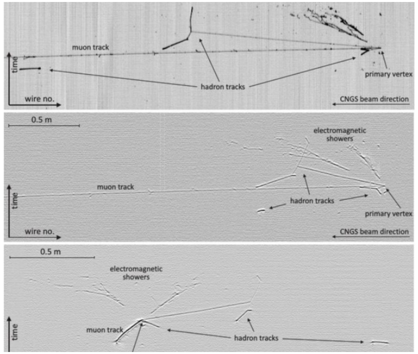


Liquid Argon Time-Projection Chambers

The modern Particle Imaging Detector

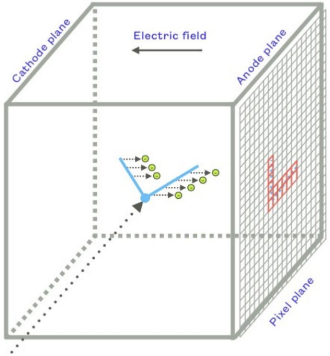
Wire TPC (2D)

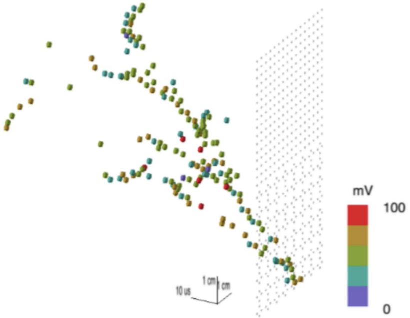




ICARUS, arXiv:1210.5089

Pixel TPC (3D)





LArPIX, arXiv:1808.02969

LArTPC are at the center stage of **beam ν physics** in the US

Short Baseline Neutrino program

- µBooNE, **ICARUS**, **SBND**

DUNE long-baseline experiment

- **Wire**: DUNE FD
- **Pixel**: **DUNE ND-LAr**

Advantages:

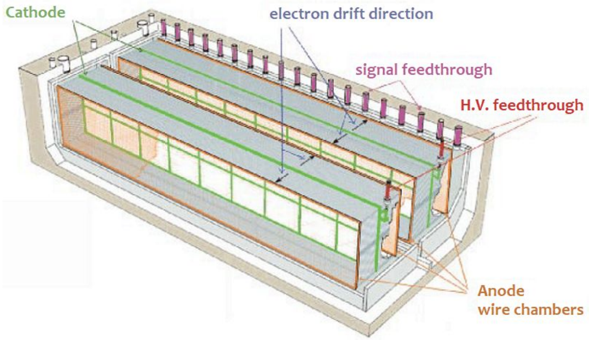
- **Detailed:** O(1) mm resolution, precise calorimetry
- **Scalable**: Up to tens of kt

Liquid Argon Time-Projection Chambers

Case study: Detector

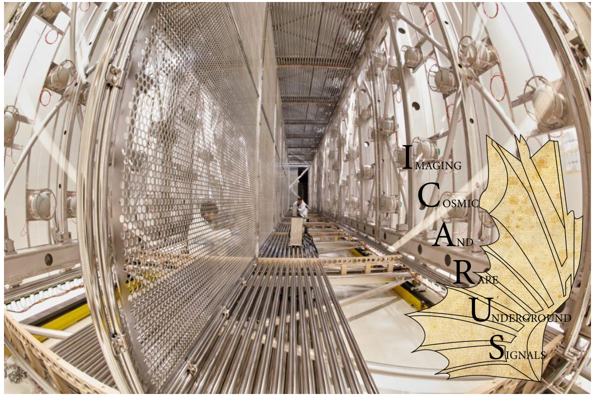
The **largest LArTPC** in operation is **ICARUS**

- **Surface**-level detector
- **500 t** fiducial mass (2 cryos, 4 TPCs)
- **Physics**: sterile neutrinos (MiniBooNE / Neutrino-4), cross sections, BSM



Event rates

- BNB beam: ~ 0.03 Hz neutrinos
- NuMI off-axis: ~ 0.015 Hz neutrinos
- In-time cosmic activity: ~ 0.25 Hz



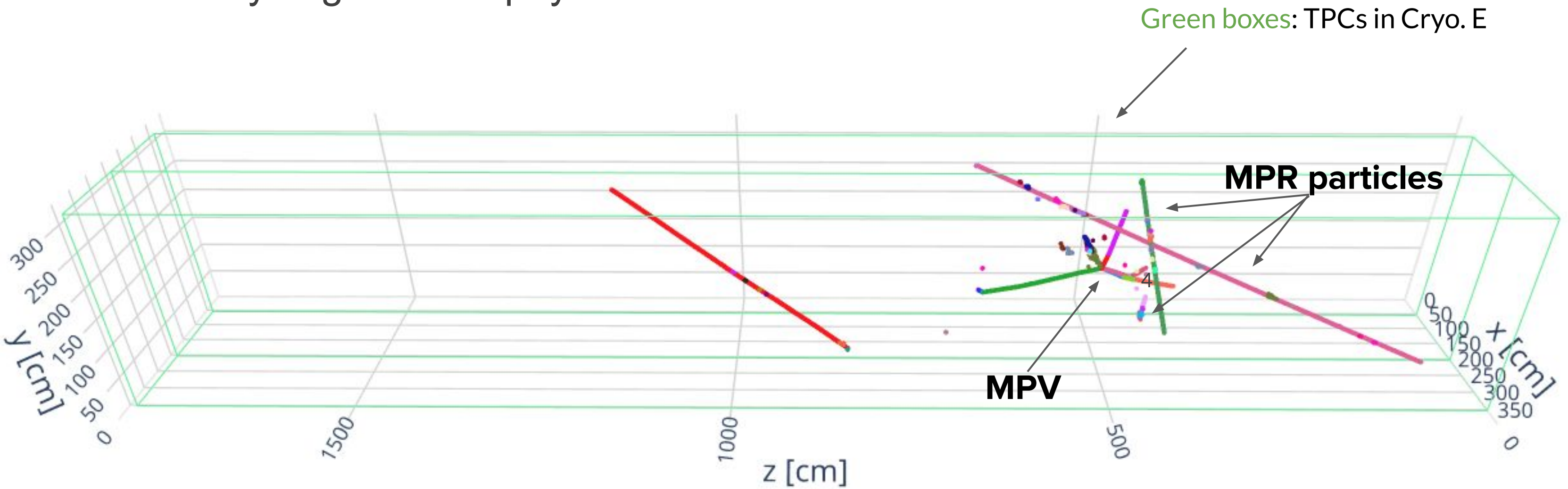
Low-rate neutrino experiment with a **significant cosmic background**

Liquid Argon Time-Projection Chambers

Case study: Datasets

Generic **simulated** dataset used for **optimization** and **testing**:

- Isotropic mix of 1 set of particles sharing a vertex + 5-9 localized single particles
    - **Covers phase-space** of neutrino interactions + cosmics, but…
    - … stays agnostic to physics → unbiased



Liquid Argon Time-Projection Chambers

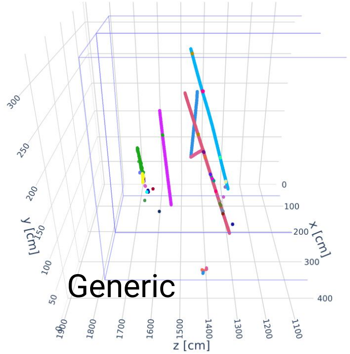
Case study: Datasets

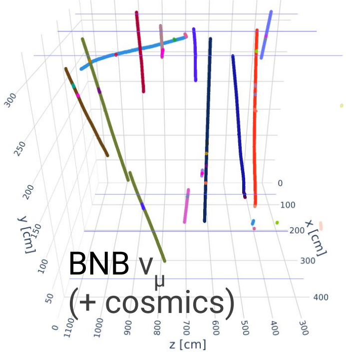
Generic **simulated** dataset used for **optimization** and **testing**:

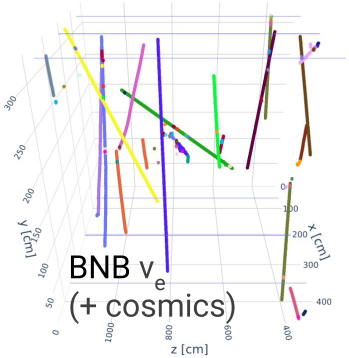
- Isotropic mix of 1 set of particles sharing a vertex + 5-9 localized single particles
  - **Covers phase-space** of neutrino interactions + cosmics, but…
  - … stays agnostic to physics → unbiased

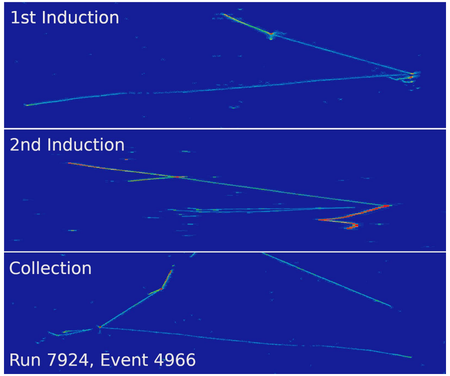
Specific datasets used for **validation**:

- Simulated **BNB ν_μ** and **BNB ν_e** + **hand-scanned data events** (C. Farnese et al.)





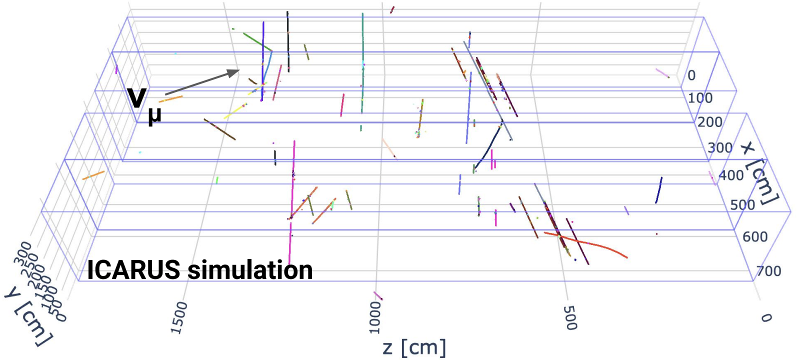


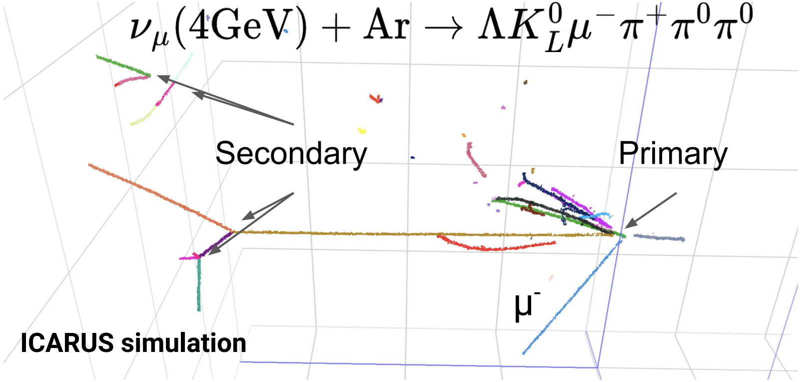


Reconstruction in LArTPCs

Challenges with LAr

Dense medium → Slow	High Z material → Messy
Electron drift velocity O(1) mm/μs	Argon has a large nucleus (Z=18)
• **Long** (O(1) ms) readout **window** • Need **light association** for timing	• **Complicated** nuclear physics • **Secondary** interactions





ICARUS ML Working Group

Group Composition

ML Working group in ICARUS lead by **F. Drielsma** and **K. Terao**

- **Do physics with ML-based LArTPC reco. chain in ICARUS**
- **LArTPC experts**: T. Usher (SLAC), M. Mooney (CSU)
- **ML experts**: L. Dominé, D.H. Koh, Y-J. Jwa (SLAC)
- **Analysts**: A. Mogan, D. Carber, J. Dyer, L. Kashur, J. Mueller (CSU, ICARUS), B. Carlson (UF, SBND)

F. Drielsma

K. Terao

T. Usher
Signal Proc.

L. Dominé
Muon decay

Y-J. Jwa
Reco.

J. Dyer
H->ee

J. Mueller
BNB ν_μ CCQE

M. Mooney
Data/sim

D. H. Koh
ν_e LEE

D. Carber
NuMI ν_e

L. Kashur
BNB ν_μ CC-π^0

A. Mogan
Cosmic rejection

Tomographic Reconstruction

Approach

The ML-based reconstruction chain is simpler in 3D space

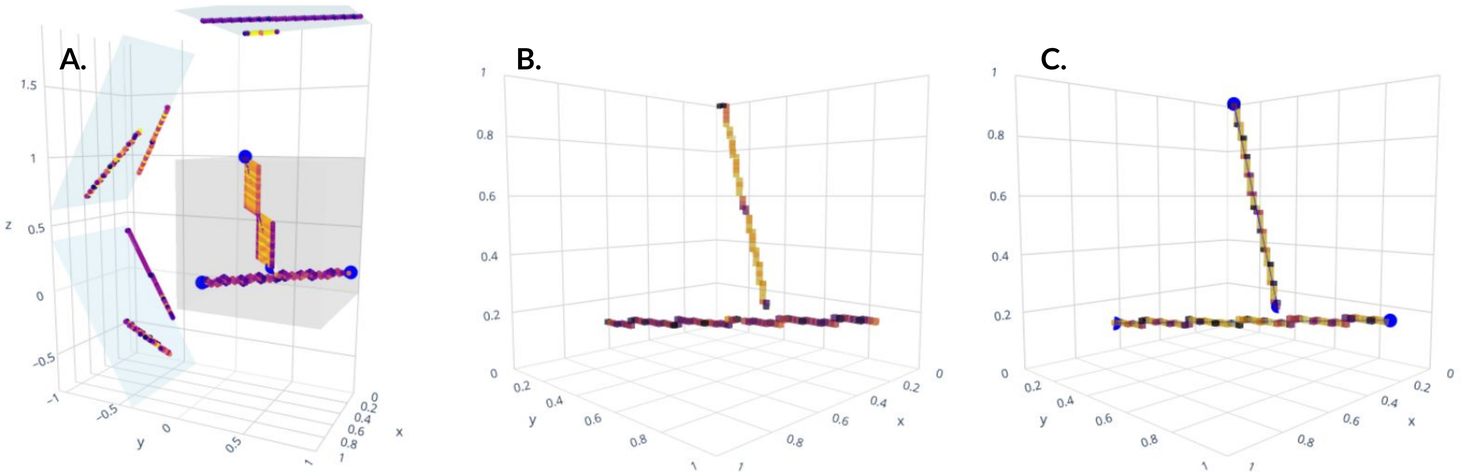
Tomographic reco. is necessary to use the ML-based chain on Wire LArTPCs

Tomographic Reconstruction

Approach

The tomographic reconstruction is broken down into three steps

A. Cluster3D (T. Usher) finds valid combinations of hits across 2 or 3 planes
B. A CNN identifies and removes artifacts of the reconstruction (deghosting)
C. Hit charge is redistributed to remaining space points



Tomographic Reconstruction

Cluster3D (T. Usher)

In the case of wire LArTPCs, we have a set of 2D hits in each of 3 projections:

- $\{h_{p,i} = (t_{p,i}, w_{p,i})\}_{i\in[0,n_p-1]}, \quad p = 0, 1, 2$
  - $t_{p,i}$ is measured along a common axis, $\mathbf{e}_t = \mathbf{e}_x$
  - $w_{p,i}$ is measured along $\mathbf{e}_p = \lambda\mathbf{e}_y + \kappa\mathbf{e}_z$

Cluster3D is a traditional algorithm which combines hits are compatible:

- Find pairs of hits, $(h_{p,i}, h_{q,j})$, which is compatible in time: $|t_{p,i} - t_{q,j}| <$ □
- Form a doublet candidate space point, $\mathbf{x}_{ij}$, where the two wires intersect
- If a hit in the third plane, $h_{r,k}$, is compatible with $\mathbf{x}_{ij}$, form a triplet $\mathbf{x}_{ijk}$

Tomographic Reconstruction

Cluster3D (T. Usher)

Two pieces of information from Cluster3D currently used by the reconstruction

Charge Q

- For doublets, $Q = Q_i + Q_j$
- For triplets, find where the WFs overlap in time and integrate the charge on the collection plane

Quality χ^2

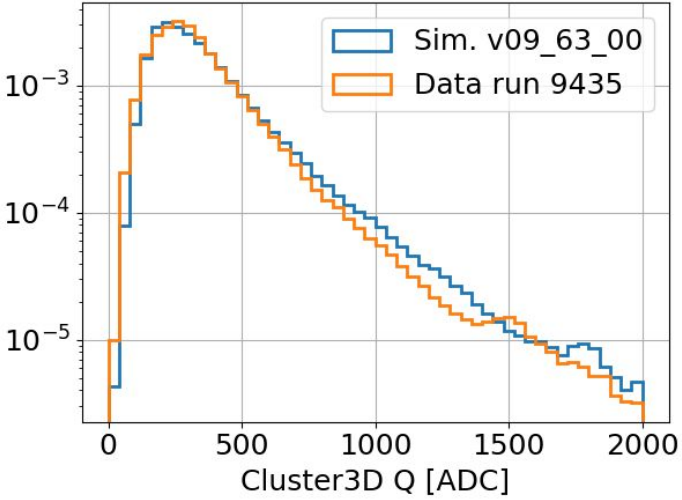
- For doublets, $\chi^2 = (t_i - t_{ij})^2 / \sigma_i^2 + (t_j - t_{ij})^2 / \sigma_j$ with t_{ij} the weighted time average
- For triplets, t_{ij} is checked against the third plane as $\chi^2 = (t_k - t_{ij})^2 / (\sigma_i^2 + \sigma_j^2 + \sigma_k^2)$

Tomographic Reconstruction

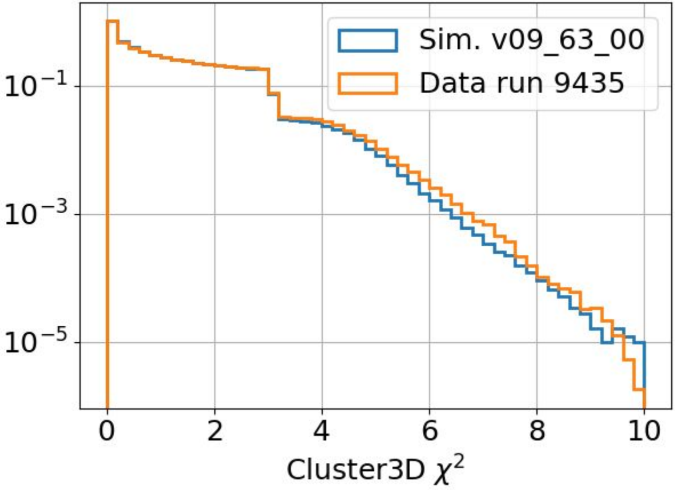
Cluster3D (T. Usher)

Two pieces of information from Cluster3D currently used by the reconstruction

Charge Q



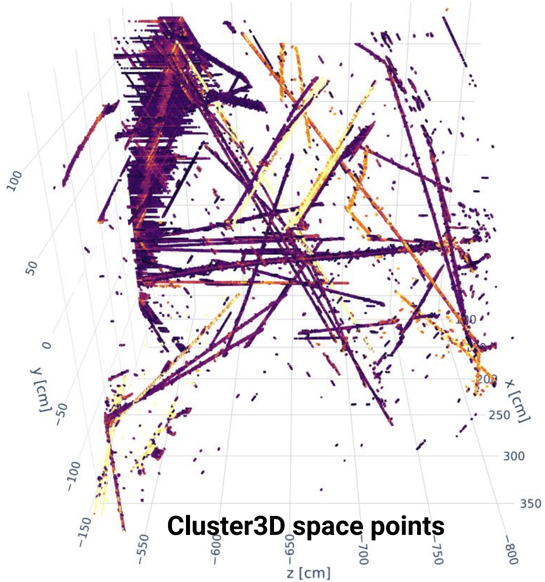
Quality χ^2



Tomographic Reconstruction

Cluster3D (T. Usher)

At this point in time, it's **hard to discern** what's going on…



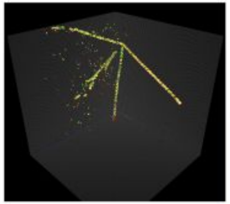
Cluster3D space points

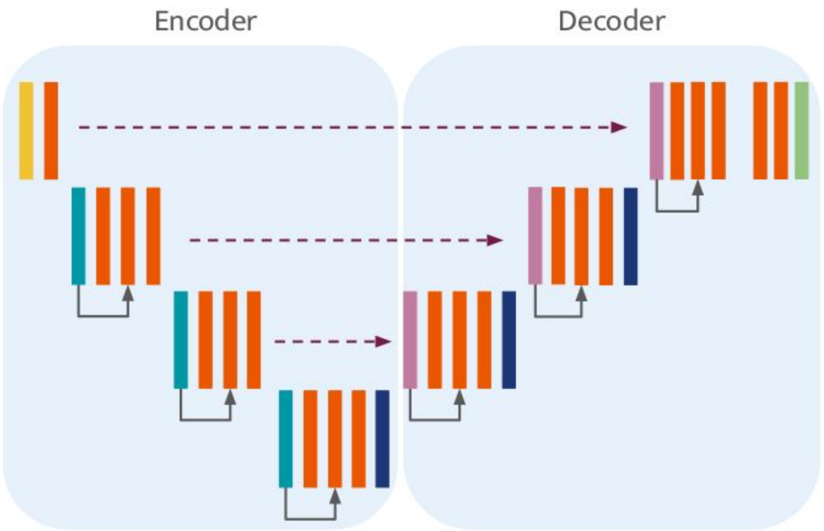
Semantic Segmentation

Backbone (L. Dominé)

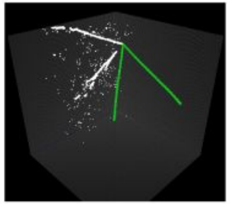
UResNet (UNet + ResNet + Sparse Conv.) as the **backbone feature extractor**

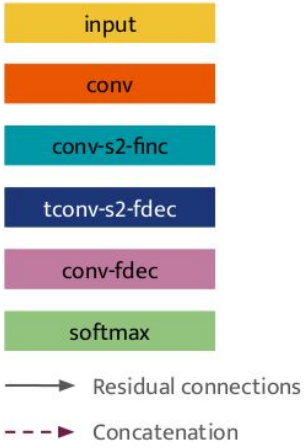
Input





Output





Paper: PhysRevD.102.012005

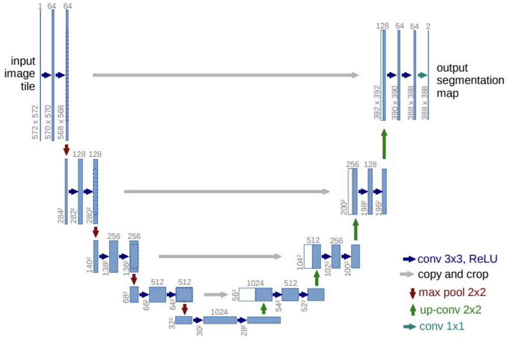
Semantic Segmentation

Backbone (L. Dominé)

UResNet (UNet + ResNet + Sparse Conv.) as the **backbone feature extractor**

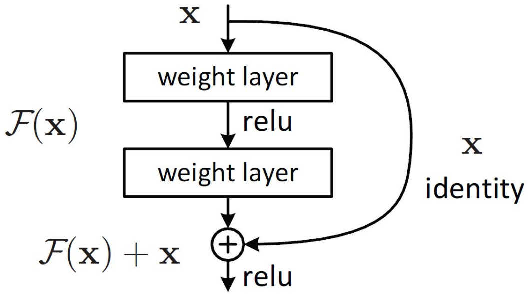
UNet

- Downsizing -> expand receptive field
- Skip connections -> preserve resolution



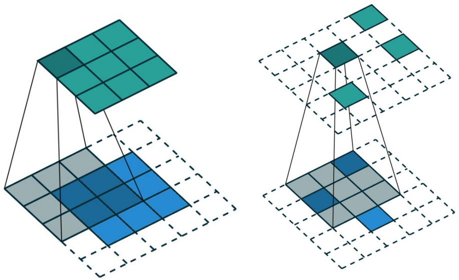
ResNet

- Identity bypass + convolution -> learns residual transform
- Speeds up learning, enables deeper networks



Sparse Convolutions

- Only applies convolutions on active pixels
- Saves memory, execution speed (dramatically)



Semantic Segmentation

UResNet architecture (L. Dominé)

Input to the network:

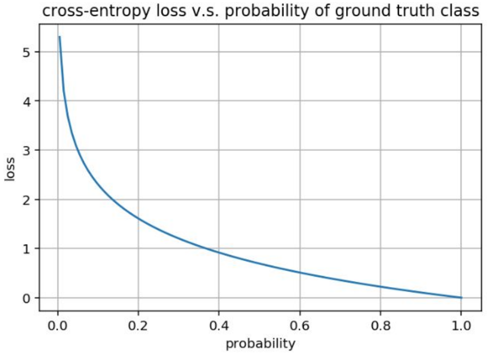
- Voxel set: rasterized Cluster3D space points (3x3x3 mm^3)
- Features (Nx2): space point charge and quality

Output:

- Features (NxC): one per target class and per voxel, $f_{i,\,c}$
  - 2 numbers for ghost labeling, 5 numbers for semantic segmentation

Loss:

- Cross-entropy loss:
  - Normalize output with Softmax: $\mathbf{s}_i = \exp(-f_{i,\,c})/\sum_c \exp(-f_{i,\,c})$
  - Loss formula: $L = -N^{-1}\sum_i \sum_c t_{i,c} \ln(s_{i,c})$ with
    $\mathbf{t}_i$ the one-hot encoded label vector, e.g. (0, 1, 0, 0, 0)



Semantic Segmentation

UResNet architecture (L. Dominé)

Now, how does one optimize an architecture like this?

- Things like UNet depth, input number of features, are **hyperparameters**
- Must scan to identify optimal values (currently using F32D5)

Filters	8	16	32
Depth 6	98.94%	99.16%	99.23%
Depth 5	98.86%	99.07%	99.06%
Depth 4	98.74%	99.00%	99.07%

TABLE III. Comparison of the non-zero accuracy at inference time on the test set of 3D 512px images for sparse U-ResNet, for different depths and initial number of filters.

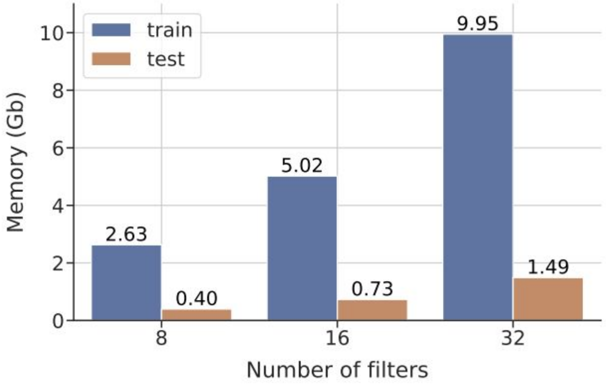


FIG. 10. Memory usage of sparse U-ResNet with depth 6, 3D 512px images and a varying number of initial filters.

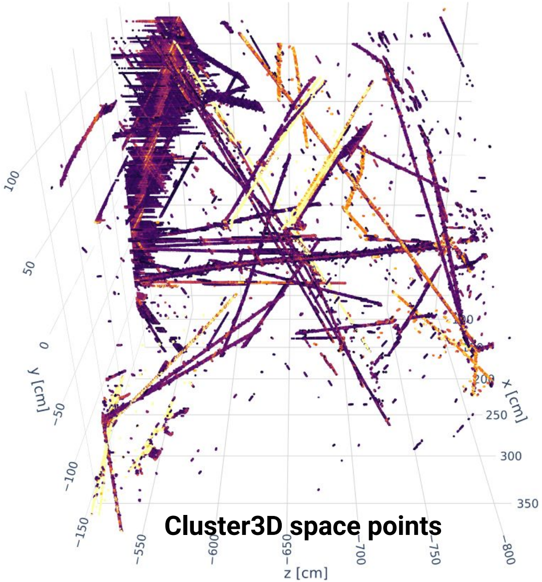
Paper: PhysRevD.102.012005

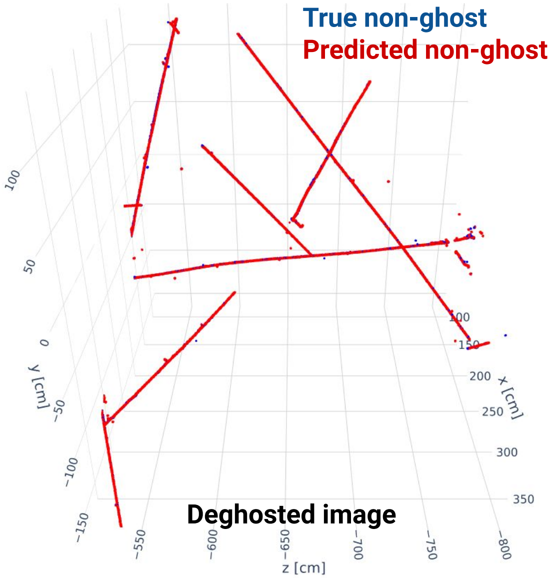
Tomographic Reconstruction

Ghost busting

Now armed with **UResNet**, let's **bust ghosts**

- Classify each voxel into two categories: **ghost** and **non-ghost**



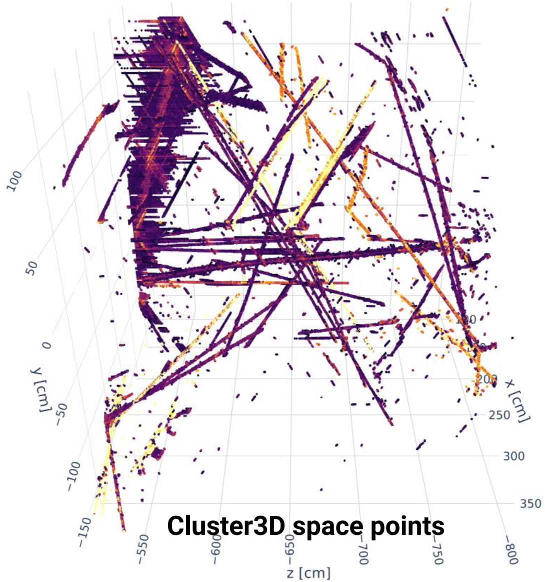


Tomographic Reconstruction

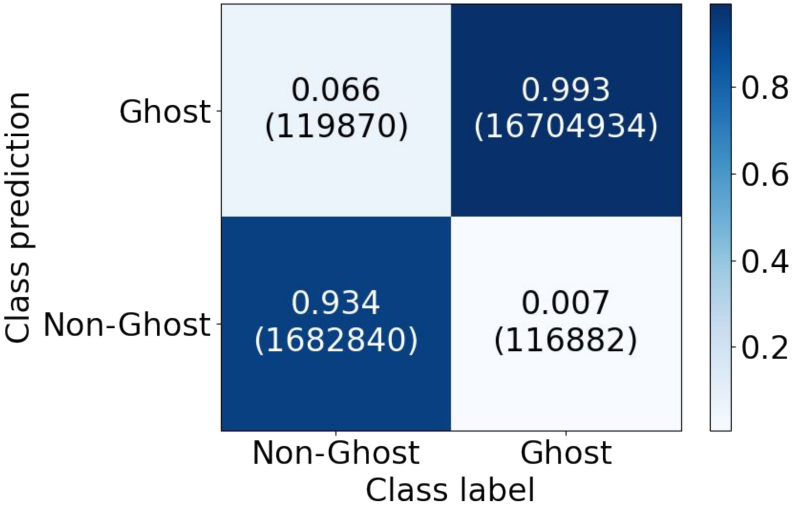
Ghost busting

Now armed with **UResNet**, let's **bust ghosts**

- Classify each voxel into two categories: **ghost** and **non-ghost**

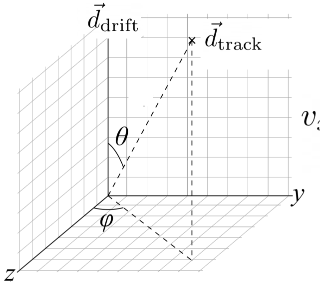


BNB ν_μ only



Tomographic Reconstruction

Track completeness

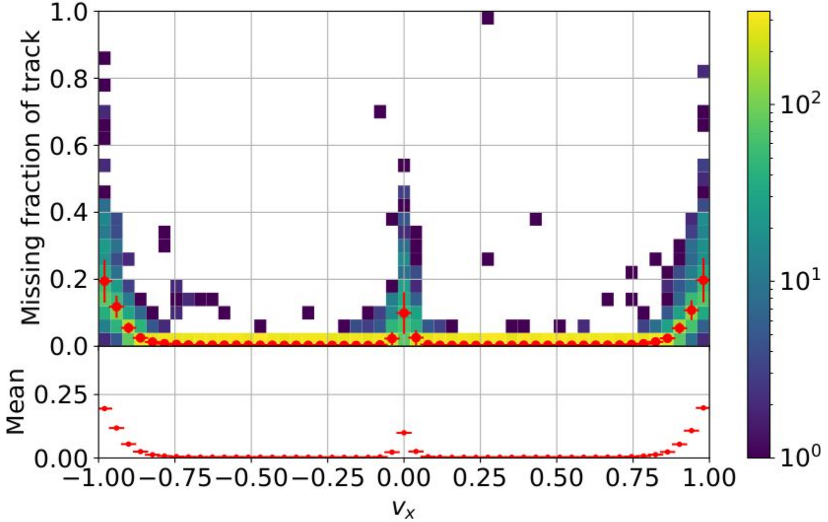


$$v_x = \vec{d}_{\text{track}} \cdot \vec{d}_{\text{drift}}$$
$$= \cos\theta$$

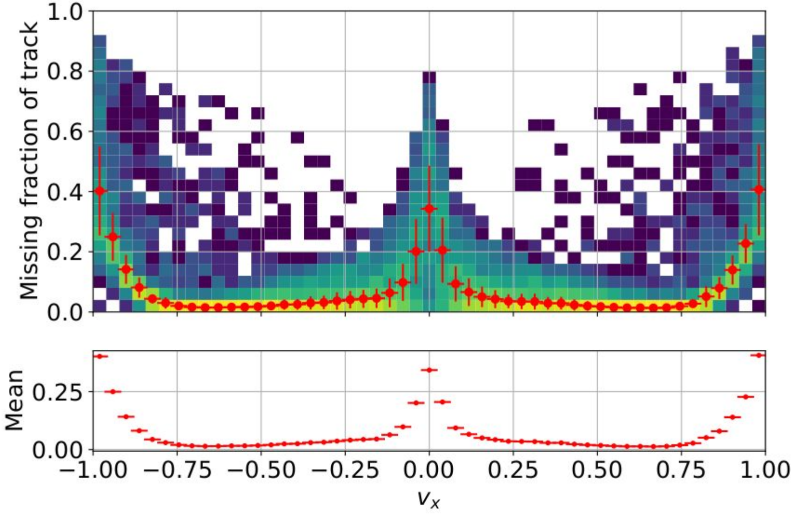
Definition: (total length of gaps)/(length of track)

- **Excellent track completeness with doublets**
- Justifies the computational cost

With doublets



Without doublets

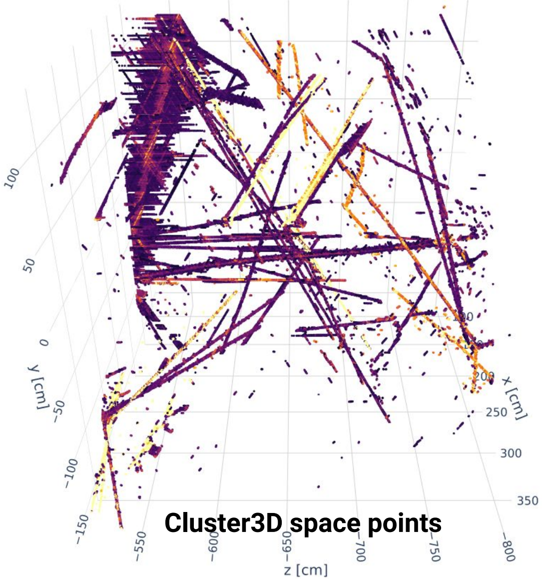


Tomographic Reconstruction

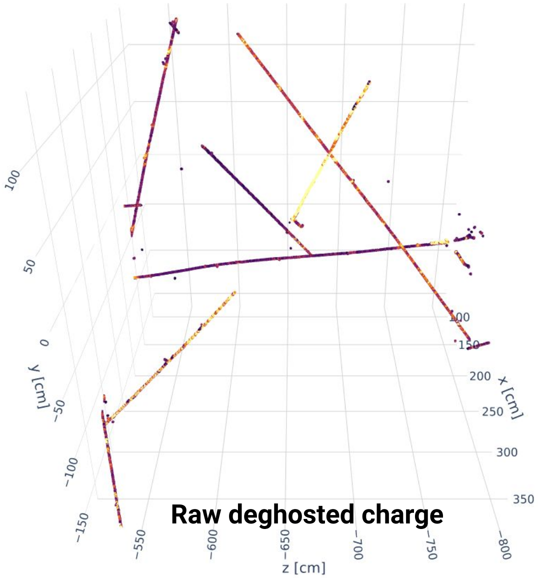
Charge conservation

What does the deghosted voxel charge look like?

- Raw charge is **very angle dependant**, because of **varying hit multiplicity**



Cluster3D space points



Raw deghosted charge

Tomographic Reconstruction

Charge rescaling

Charge redistribution scheme:

1. Keep track of hit composition for each Cluster3D SP
2. Record charge and hit composition of selected SP for each voxel
   - 2(+1) unique hit identifiers and 2(+1) hit integrated charge values (one per plane)
3. Apply deghosting algorithm on image
   - Use charge and χ^2 of selected space point as features
4. Count the number of times, $n_{p,i}$, each hit, $h_{p,i}$, is used in the deghosted voxels
5. Recompute the corrected charge of all voxels, which account for hit multiplicity

$$Q_\tau = \frac{1}{n_{hits}} \sum_{p \in u,v,w} Q_{p,\tau_p} / n_{p,\tau_p}$$

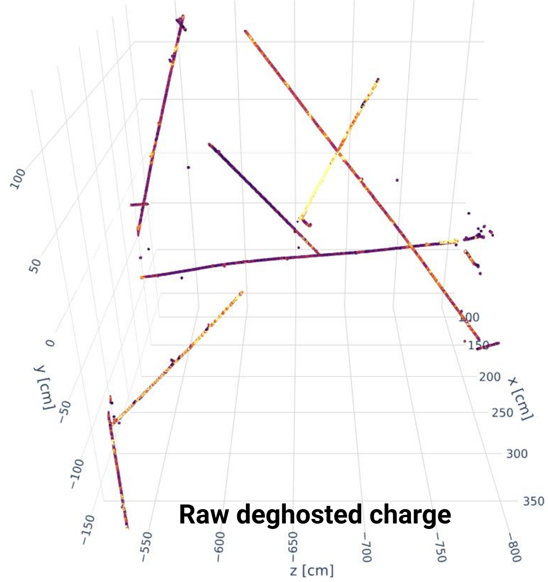
with Q_τ the charge associated with doublet $\tau = (i, j)$ or triplet $\tau = (i, j, k)$.

Tomographic Reconstruction

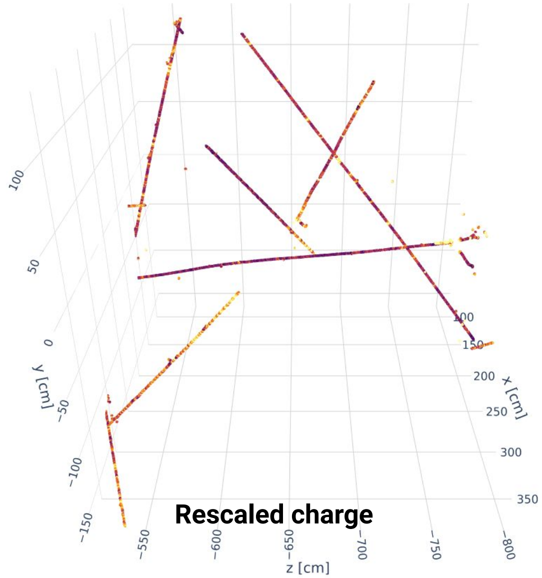
Ghost busting

Now armed with **UResNet**, let's

- Raw charge is very angle dependant, because if hit multiplicity!



Raw deghosted charge



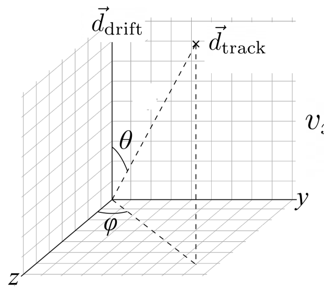
Rescaled charge

Tomographic Reconstruction

Charge conservation

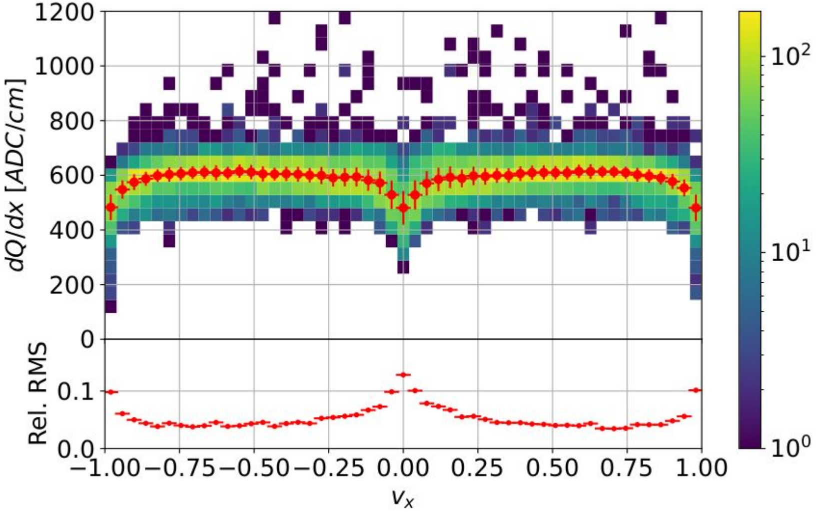
Definition: (total track charge)/(length of track)

- **dQ/dx affected by remaining inefficiencies**
- Unmatched hits -> loss of charge

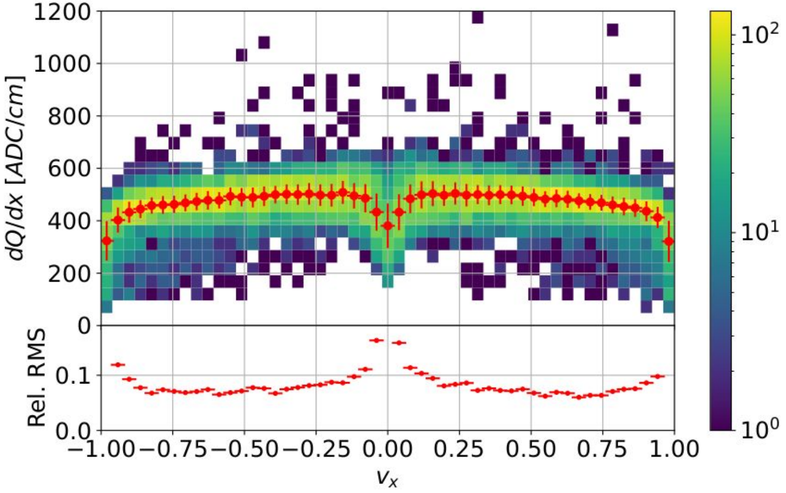


$$v_x = \vec{d}_{\mathrm{track}} \cdot \vec{d}_{\mathrm{drift}} = \cos\theta$$

With doublets



Without doublets

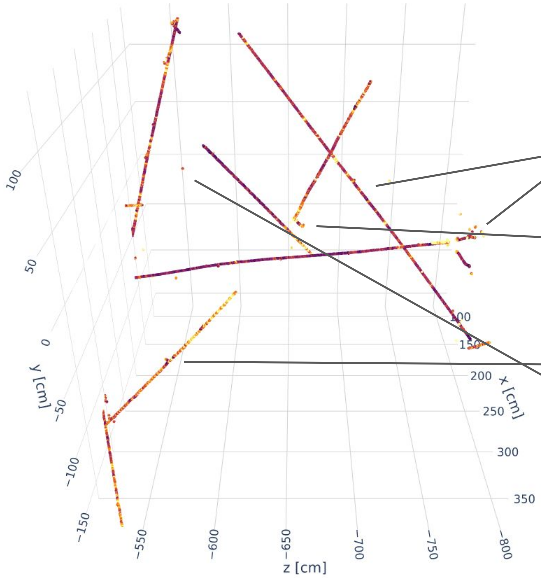


Semantic segmentation

Particle topologies

Now that we only have legitimate space points left:

- Very clear topological differences in leftover voxels (UResNet!)



Primary **showers** have sparse and broad fragments

Tracks are rectilinear and continuous

Michel electrons live next to a muon Bragg peak

Delta rays are attached to side of tracks

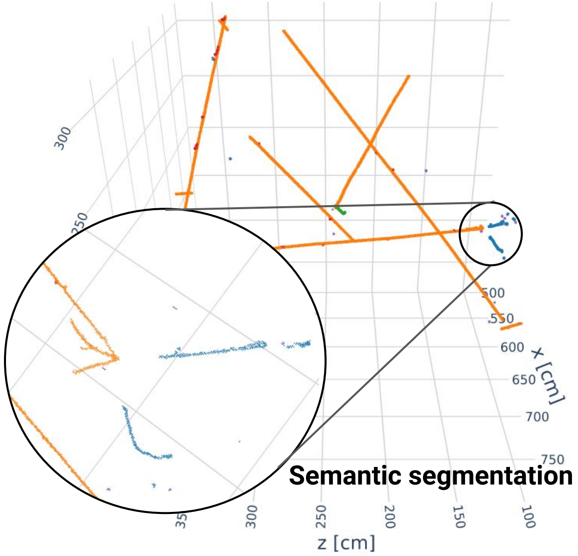
Low energy deposits are amorphous and isolated

Semantic Segmentation

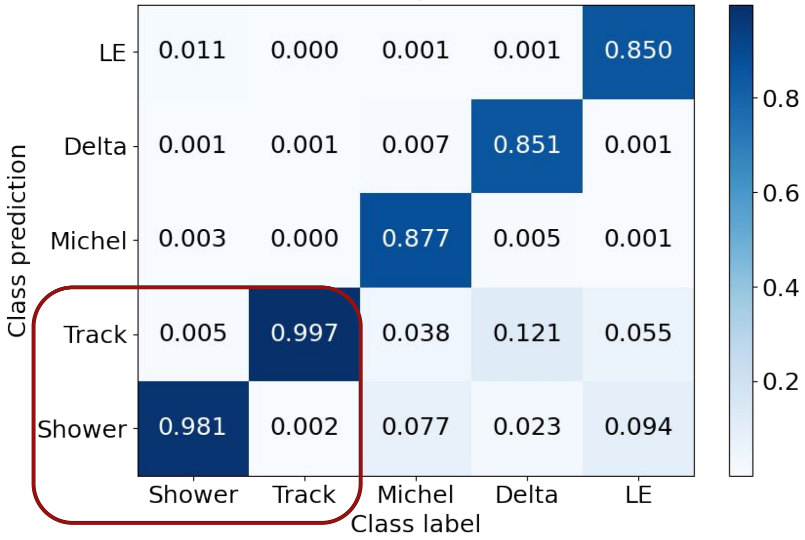
Performance

Separate **topologically different** types of activity

- **Showers**, **Tracks**, **Michel electrons**, **delta rays**, **low energy blips**



BNB ν_μ only



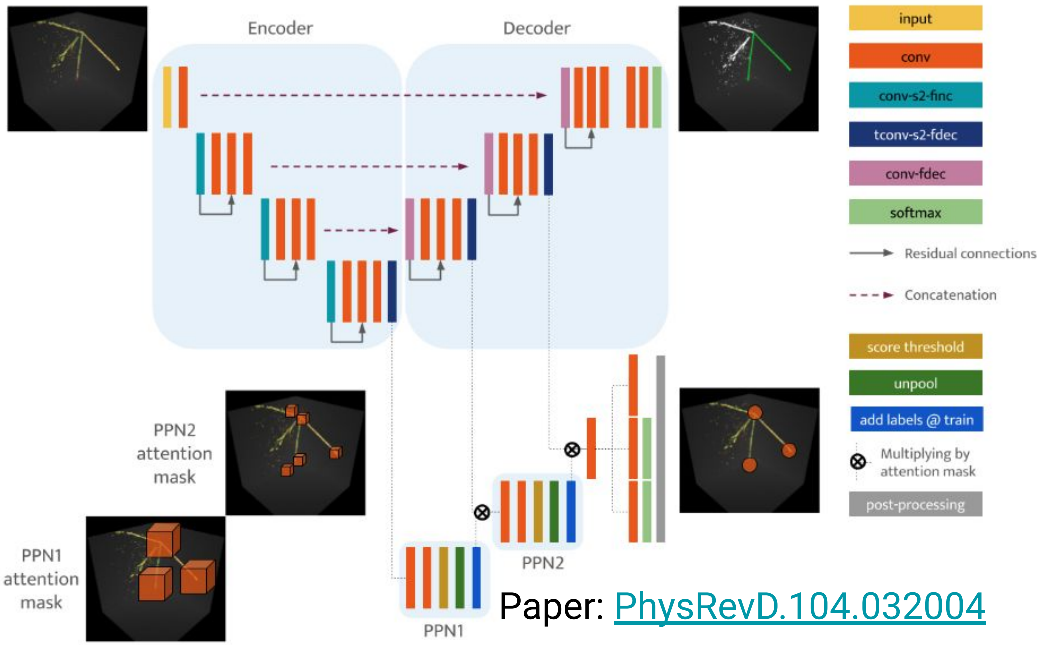
Paper: PhysRevD.102.012005

Point Proposal Network (PPN)

Architecture (L. Dominé)

The Point Proposal Network (PPN) identifies **points of interest** using decoder features:

- Three CCN layers to progressively narrow ROI
- Last layer reconstructs:
  - Relative position to voxel center of active voxel
  - Point type
- Post-processing aggregates nearby points



Paper: PhysRevD.104.032004

Point Proposal Network (PPN)

Architecture (L. Dominé)

Input to the network:

- Voxel set: deghosted and rescaled voxels
- Features (Nx1): rescaled charge in each voxel

Output:

- PPN1 (N”x2): scores for positive/negative on each voxel at depth D1
- PPN2 (N’x2): scores for positive/negative on each voxel at depth D2
- PPN3 (Nx10): scores for
    - Positive/negative (2): Is a pixel within 5 voxels of a point of interest at the original res.?
    - Class (5): Which type of particle is a point associated with?
    - Position (3): How far is the voxel center from the point of interest?
    - Start/end point (2): Is this voxel at the start or the end of a particle trajectory?

Point Proposal Network (PPN)

Architecture (L. Dominé)

Losses:

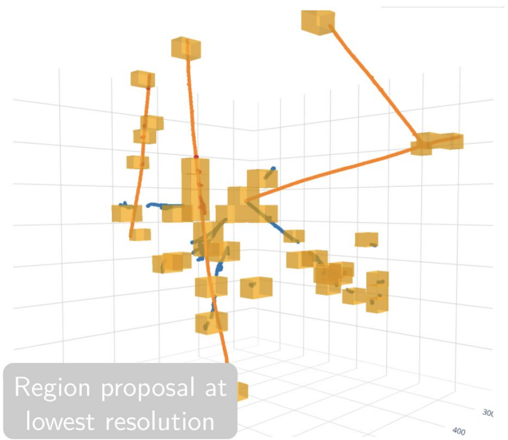
- Cross-entropy loss on classification tasks (same as segmentation)
  - Positive/negative classification (one per PPN layer, so three of them)
  - Start/end classification (only on positive points)
  - Point type classification (only on positive points)
- L2 loss on position regression
  - Loss only applied for positive voxels
  - $\mathcal{L} = \frac{1}{N} \sum_i \sum_j \min_j ||\mathbf{v}_i + \mathbf{q}_i - \mathbf{p}_j||$
    - $\mathbf{v}_i$ is the center of the ith voxel
    - $\mathbf{q}_i$ is is the predicted displacement for the ith voxel
    - $\mathbf{p}_j$ is the position of the jth label point of interest
- The total loss is the **sum of all 6 losses**

Point Proposal Network (PPN)

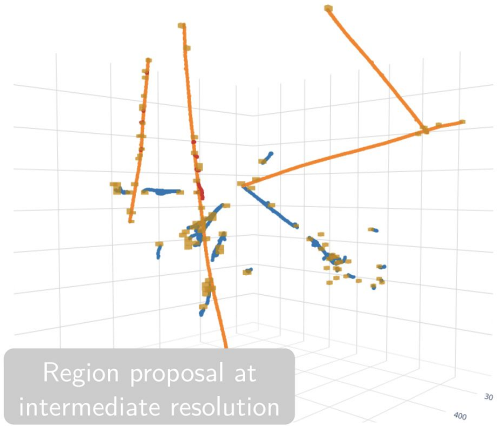
Masks

Let’s take a look at what the masks look like at each layer (other image)

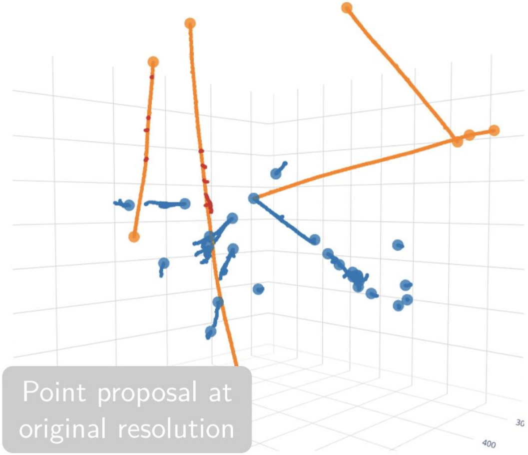
- Regions of interest already **identified at lower resolutions**
- **Resolution improved** at each layer



Region proposal at lowest resolution



Region proposal at intermediate resolution

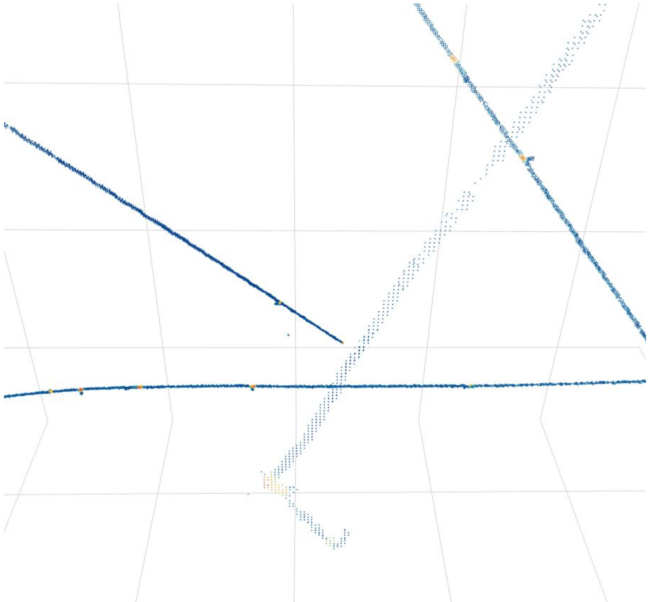


Point proposal at original resolution

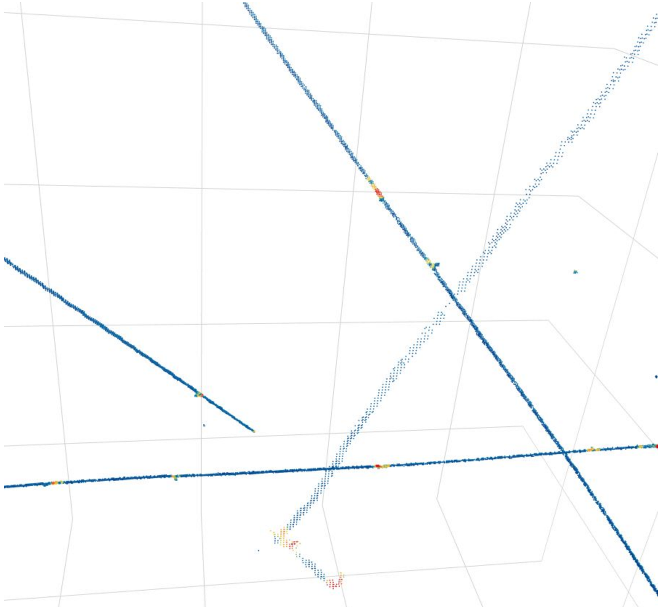
Point Proposal Network (PPN)

Scores and distances

Let’s take a look at what the score and distance predictions look like



PPN scores (0->1)



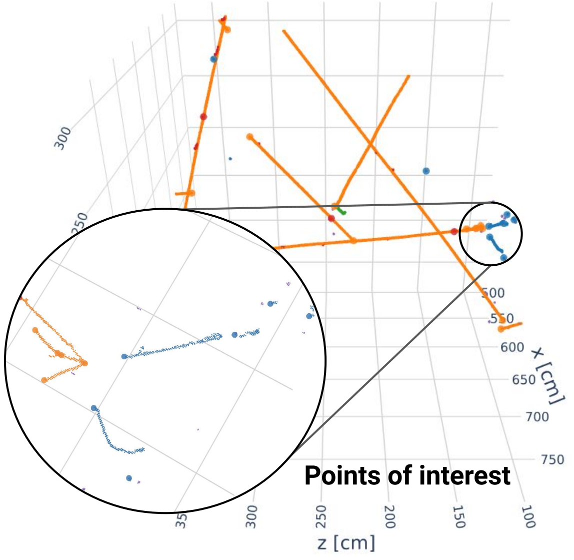
PPN distances (0->5)

Point Proposal Network (PPN)

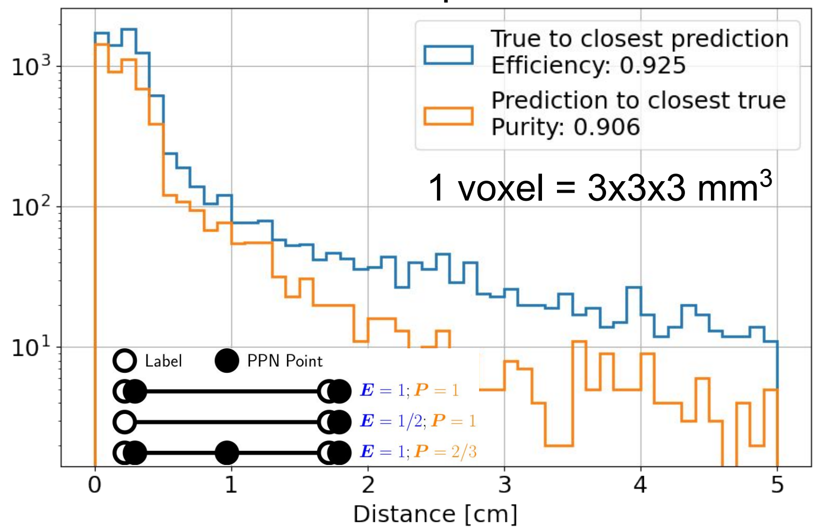
Points of interest

Narrow down a region proposal all the way to a **point**

- Predict **masks** at different scales with UResNet, predict **position** in voxel



BNB ν_μ only



Paper: [PhysRevD.104.032004]

Clustering

Considerations

So far, each voxel has been **treated individually**. What about **clustering**?

- We don't know how many cluster an image has
- It is no longer a simple classification/regression task for each voxel

Clustering

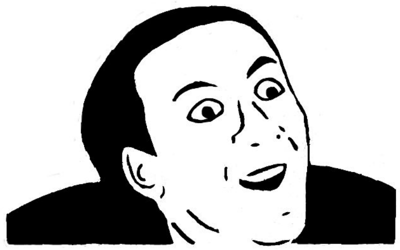
Considerations

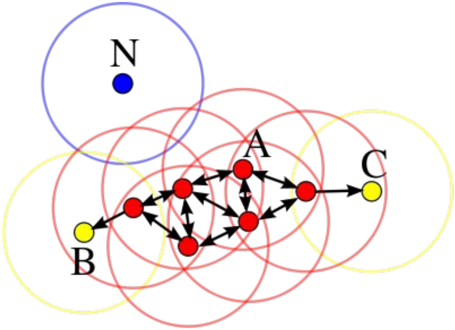
So far, each voxel has been **treated individually**. What about **clustering**?

- We don't know how many cluster an image has
- It is no longer a simple classification/regression task for each voxel

"Why don't you use DBSCAN, dummy?"

- DBSCAN finds connected components, i.e. voxels that touch each other
  - It can give you individual **shower fragments** reliably (although don't forget S->ee!)
  - It can cluster **uninterrupted solitary tracks** reliably





Clustering

Considerations

So far, each voxel has been **treated individually**. What about **clustering**?

- We don't know how many cluster an image has
- It is no longer a simple classification/regression task for each voxel

"Why don't you use DBSCAN, dummy?"

- DBSCAN finds connected components, i.e. voxels that touch each other
  - It can give you individual **shower fragments** reliably (although don't forget S->ee!)
  - It can cluster **uninterrupted solitary tracks** reliably
- But…
  - What about tracks (or showers) coming from a common vertex?
  - What about tracks that have a break in them (cathode crosser, inefficiencies, etc.)?
  - How do you put shower fragments together?
  - (DBSCAN typically optimized for CPU, it ain't cheap to run…)

Clustering

Considerations

So far, each voxel has been **treated individually**. What about **clustering**?

- We don't know how many cluster an image has
- It is no longer a simple classification/regression task for each voxel

"Why don't you use DBSCAN, dummy?"

- DBSCAN finds connected components, i.e. voxels that touch each other
    - It can give you individual **shower fragments** reliably (although don't forget S->ee!)
    - It can cluster **uninterrupted solitery tracks** reliably
- But…
    - What about tracks (or showers) coming from a common vertex?
    - What about tracks that have a break in them (cathode crosser, inefficiencies, etc.)?
    - How do you put shower fragments together?
    - (DBSCAN typically optimized for CPU, it ain't cheap to run…)
- Let's address problem **1**, we'll deal with **2** and **3** later

Dense clustering problem

Aggregation problem

Dense clustering

Approach

You might say: “Just use PPN points to break touching instances”

- We tried, let’s just say “not great, not terrible”:
    - Tracks can touch away from PPN points (relatively rare in 3D, thank you tomographic reco.)
    - Small tracks will get killed in the process (anything < masking radius)
    - Highly colinear tracks don’t get broken up (detaching point > masking radius)
    - Does not disentangle colinear showers (no breaking point, many boundaries)

Dense clustering

Approach

You might say: “Just use PPN points to break touching instances”

- We tried, let’s just say “not great, not terrible”:
  - Tracks can touch away from PPN points (relatively rare in 3D, thank you tomographic reco.)
  - Small tracks will get killed in the process (anything < masking radius)
  - Highly colinear tracks don’t get broken up (detaching point > masking radius)
  - Does not disentangle colinear showers (no breaking point, many boundaries)

How else can we do it?

1. Need to **transform** input voxels to a **space where clusters are disconnected**
2. Ask the network to give you information about **location** and **size** of clusters
3. Cluster in the transformed space using either
   - Informed Gaussian mixture: **SPICE**
   - Graph edge selection and connected components: **Graph-SPICE** (smart DBSCAN!)

Dense clustering

Spatial transformation

Losses in the embedding (transformed) space:

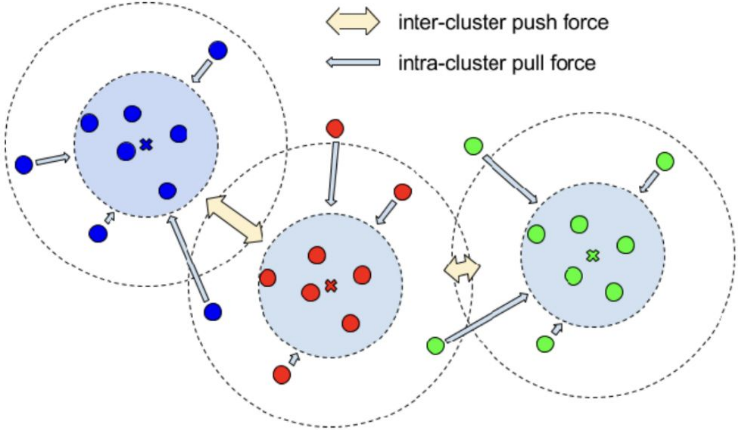
- Encourage points from the same cluster to stick together (L_{var})
- Encourage points from separate clusters to distance themselves (L_{dist})
- Regulalize to prevent distances exploding (L_{reg})

$$L = \alpha L_{var} + \beta L_{dist} + \gamma L_{reg},$$

$$L_{var} = \frac{1}{C}\sum_{c=1}^{C}\frac{1}{N_c}\sum_{i=1}^{N_c}[\max(0, \|\mu_c - x_i\| - \delta_v)]^2$$

$$L_{dist} = \frac{1}{C(C-1)}\sum_{\substack{c_A, c_B=1 \\ c_A \neq c_B}}^{C}[\max(0, 2\delta_d - \|\mu_{c_A} - \mu_{c_B}\|)]^2$$

$$L_{reg} = \frac{1}{C}\sum_{c=1}^{C}\|\mu_c\|$$

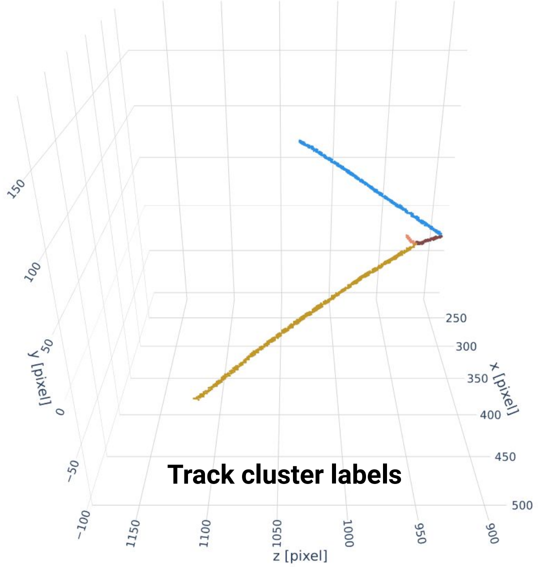


Dense clustering

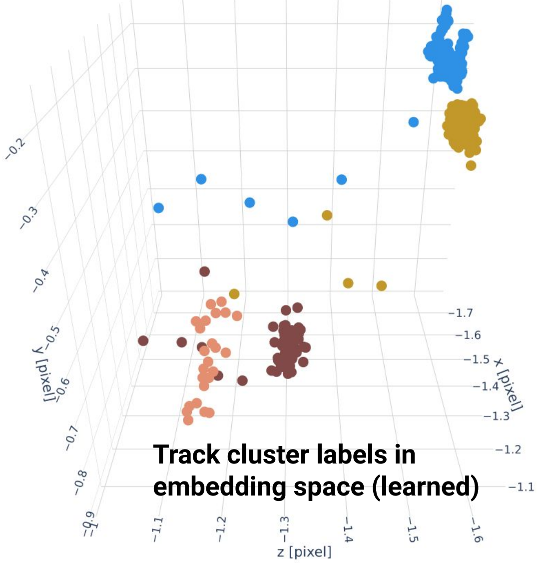
Spatial transformation

In practice, this makes touching tracks visually distinct!

- How do we put cluster them together ?



Track cluster labels



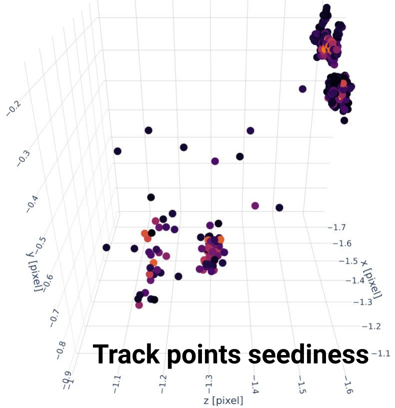
Track cluster labels in embedding space (learned)

SPICE

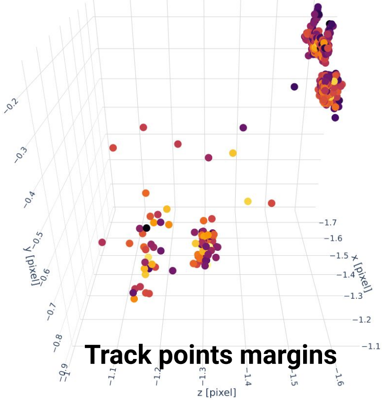
Gaussian mixture

One way is to use an informed **Gaussian mixture.** For each voxel:

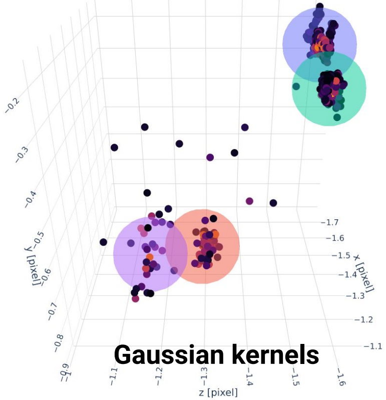
- Predict proximity to a cluster centroid (**seediness** score, s_i)
- Predict size (**margin**, σ_i) of cluster it belongs to
- Pick highest seediness point, merge points that satisfy $\exp(\frac{-(\mathbf{x}_j-\mathbf{x}_i)^2}{2\sigma_i^2}) > 0.5$
- Repeat until the next be seed is < 0.5



Track points seediness



Track points margins

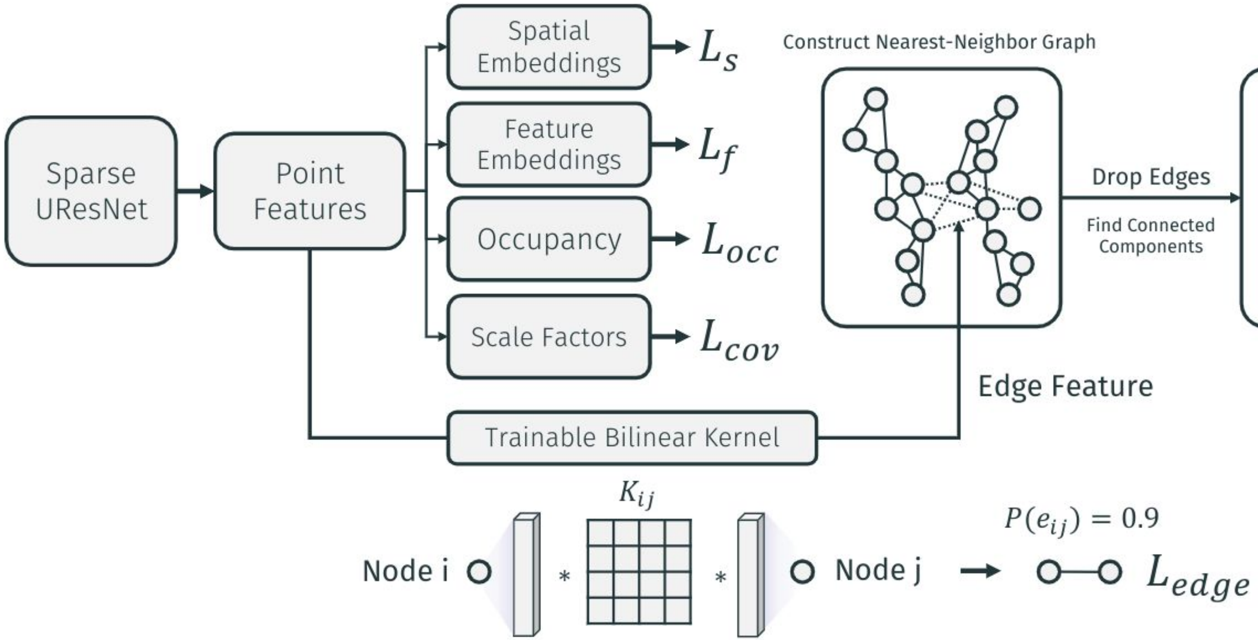


Gaussian kernels

Graph-SPICE

Connected components

Another way: **build a kNN graph** and to **find connected components**



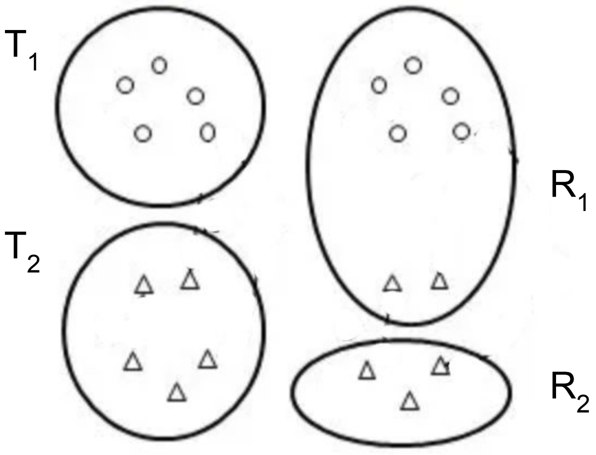
Clustering metrics

Definitions

Quantifying clustering accuracy is not trivial

- There is no single-voxel accuracy (cluster ID is not fixed)

Eff = (1+0.6)/2 = 0.8
Pur = (0.71+1)/2 = 0.85



Three metrics we use:

- **Efficiency**:
  - $\frac{1}{N_t}\sum_i^{N_t} \max_j \#(T_i \cap R_j)/\#T_i$
- **Purity**:
  - $\frac{1}{N_r}\sum_i^{N_r} \max_j \#(R_i \cap T_j)/\#R_i$

Clustering metrics

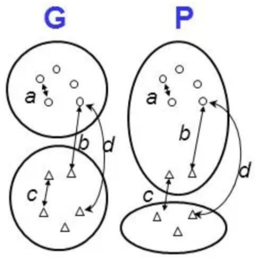
Definitions

Quantifying clustering accuracy is not trivial

- There is no single-voxel accuracy (cluster ID is not fixed)

Three metrics we use:

- **Efficiency**:
  - $\frac{1}{N_t}\sum_i^{N_t} \max_j \#(T_i \cap R_j)/\#T_i$
- **Purity**:
  - $\frac{1}{N_r}\sum_i^{N_r} \max_j \#(R_i \cap T_j)/\#R_i$
- **Adjusted Rand Index**



Agreement: a, d
Disagreement: b, c

$$RI(P,G) = \frac{a+d}{a+b+c+d}$$

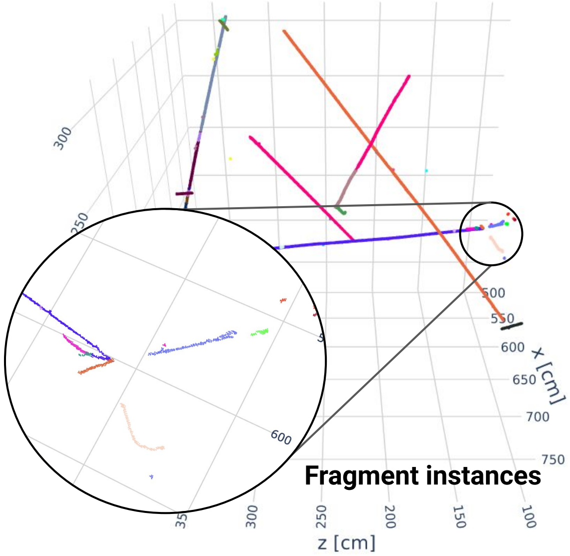
$$ARI = \frac{RI - E(RI)}{1 - E(RI)}$$

Graph-SPICE

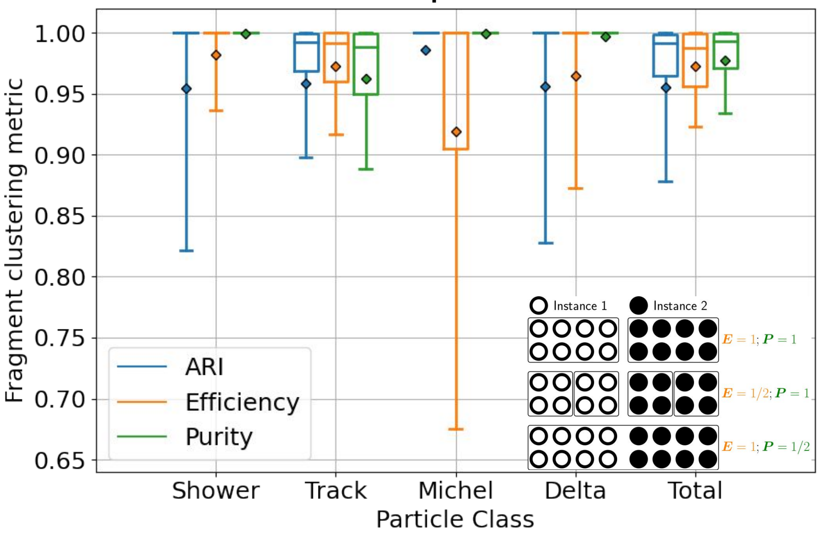
Performance

This approach works significantly better than PPN+DBSCAN

- Cluster track/shower **fragments** at this stage



BNB ν_μ only



Paper: arXiv:2007.03083

Aggregation

Considerations

Sweet, we solved problem 1, now we have **particle fragments**:

- Tracks are broken up where there's gaps (inefficiencies/dead material)
- Showers are broken up in fragments (e^+/e^- constituents)

How do we **aggregate** them them together?

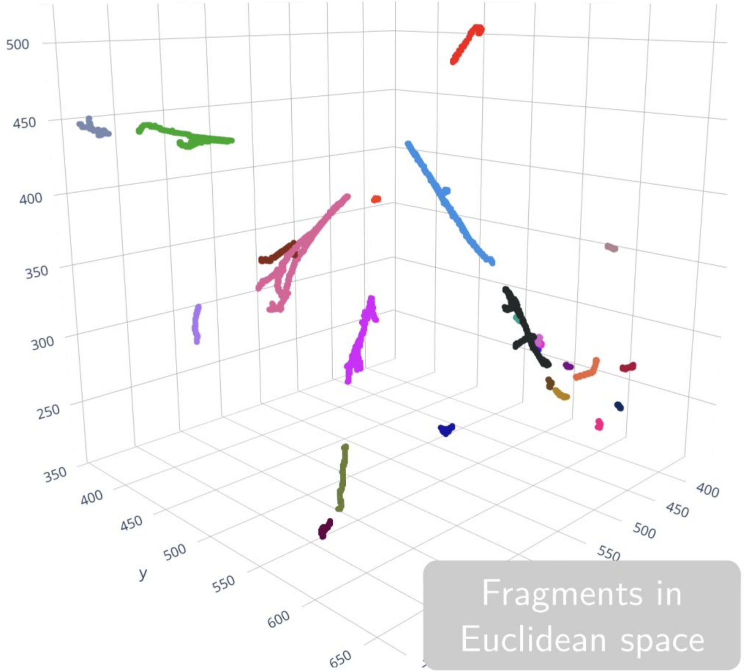
- Treat **each fragment as a whole**, encode it:
  - Use a **set of summary statistics**
- Find out **which fragments belong together,** which don't
  - This sounds a lot like graph edge classification problem
- **Along the way**: find out information about individual fragments
  - This sounds a lot like a graph node classification problem

Aggregation

Node encoding

Input:

- Particle fragments



Aggregation

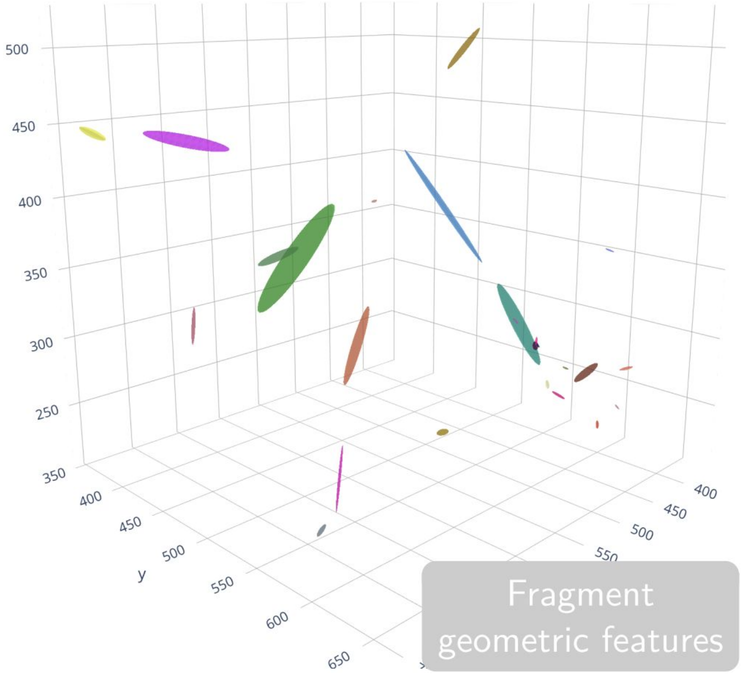
Node encoding

Input:

- Particle fragments

Node features:

- Particle centroid
- Covariance matrix, PCA
- End points (PPN), directions, dQ/dx



Aggregation

Node encoding

Input:

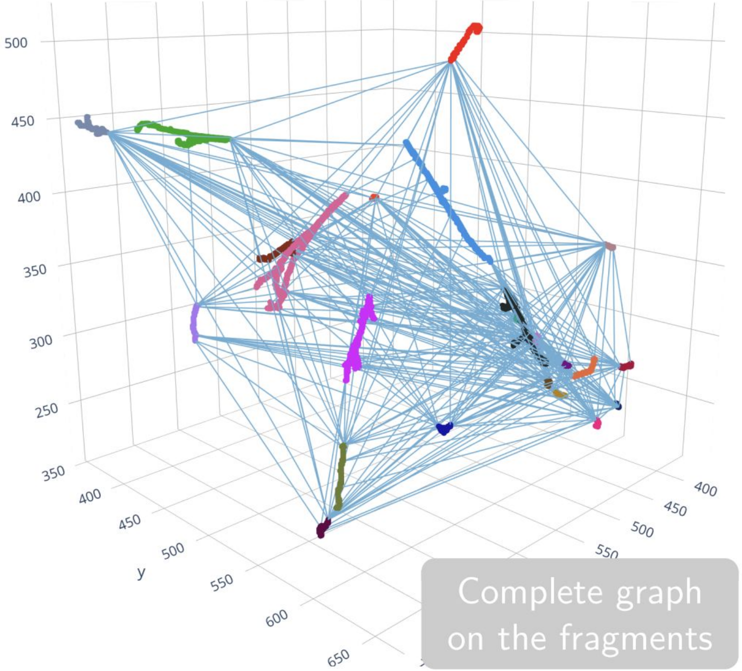
- Particle fragments

Node features:

- Particle centroid
- Covariance matrix, PCA
- End points (PPN), directions, dQ/dx

Input graph:

- All edges within some natural limit



Aggregation

Node encoding

Input:

- Particle fragments

Node features:

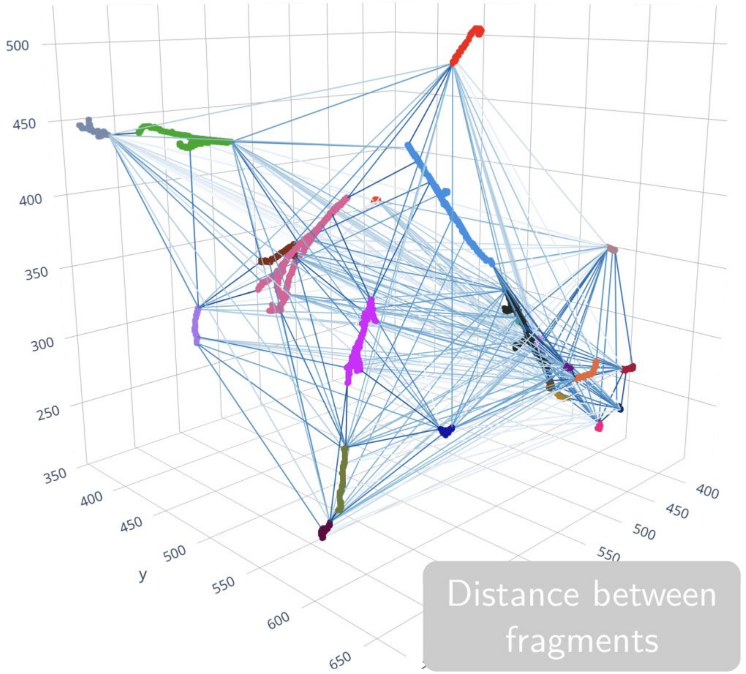
- Particle centroid
- Covariance matrix, PCA
- End points (PPN), directions, dQ/dx

Input graph:

- All edges within some natural limit

Edge features:

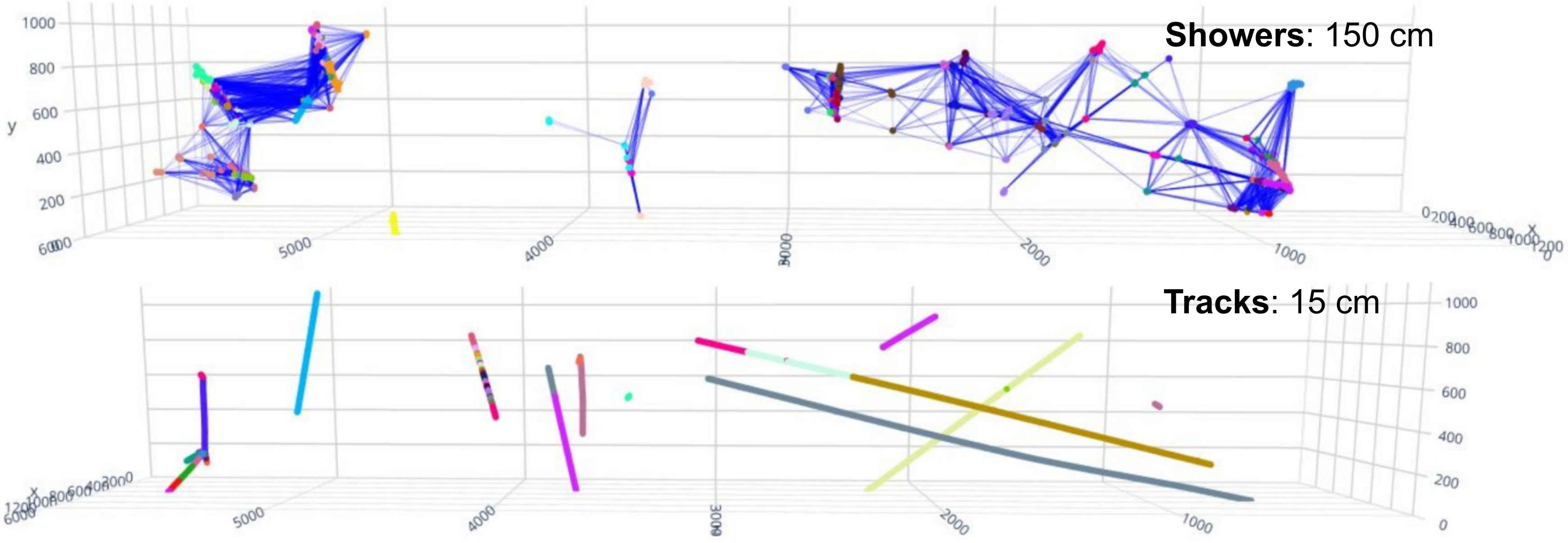
- Closest points of approach
- Displacement vector



Aggregation

Graph construction

Valid input edges careful **selected** depending on **particle type** (segmentation)



Aggregation

Message passing, loss

Lots of edges to sort through. Must propagate information through **message passing** (MetaLayer arXiv:1806.01261):

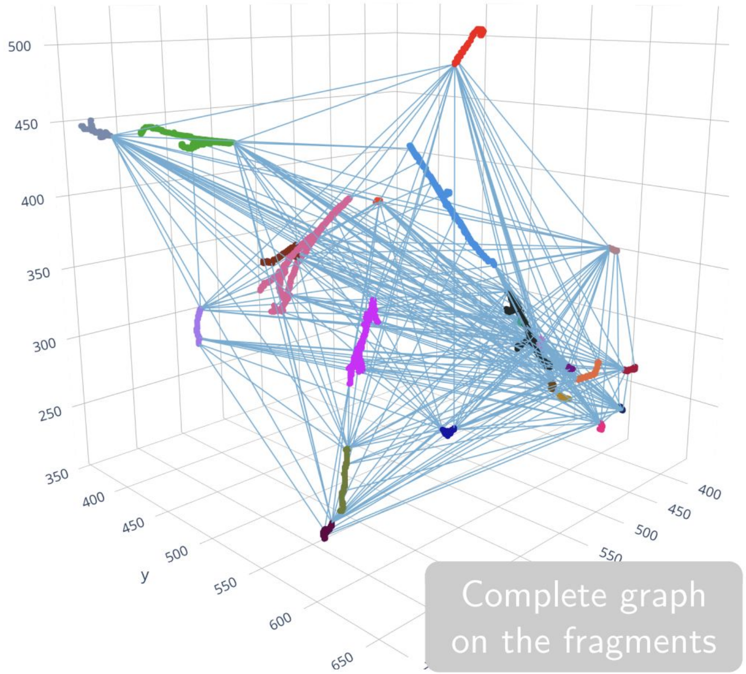
- Edge update:

$$\mathbf{e}'_{ij} = \phi_\Theta(\mathbf{x}_i, \mathbf{x}_j, \mathbf{e}_{ij})$$

- Node update:

$$\mathbf{m}_{ji} = \chi_\Theta(\mathbf{x}_j, \mathbf{e}_{ji})$$

$$\mathbf{x}'_i = \psi_\Theta(\mathbf{x}_i, \square_{j \in \mathcal{N}(i)} \mathbf{m}_{ji})$$



Aggregation

Message passing, loss

Lots of edges to sort through. Must propagate information through **message passing** (MetaLayer arXiv:1806.01261):

- Edge update:

$$\mathbf{e}'_{ij} = \phi_\Theta(\mathbf{x}_i,\ \mathbf{x}_j,\ \mathbf{e}_{ij})$$

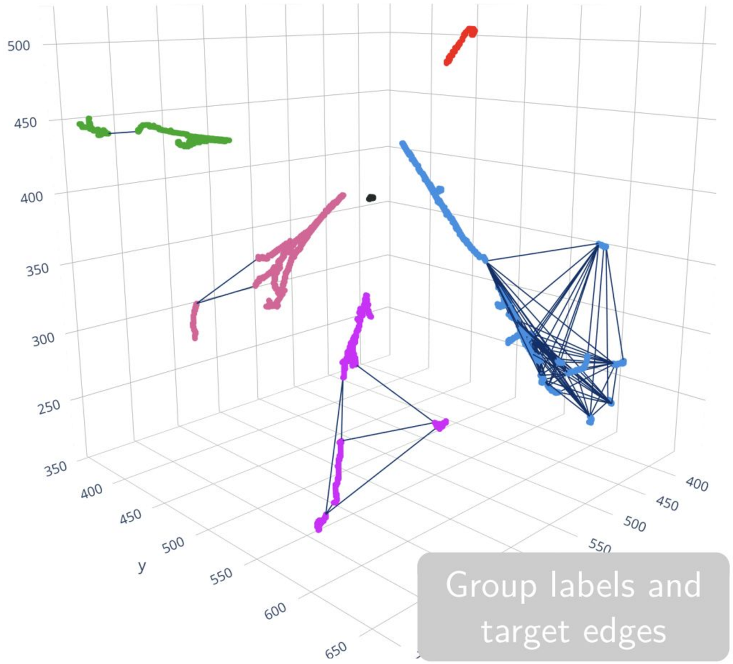
- Node update:

$$\mathbf{m}_{ji} = \chi_\Theta(\mathbf{x}_j,\ \mathbf{e}_{ji})$$

$$\mathbf{x}'_i = \psi_\Theta(\mathbf{x}_i,\ \square_{j\in\mathcal{N}(i)}\mathbf{m}_{ji})$$

After 3 iterations:

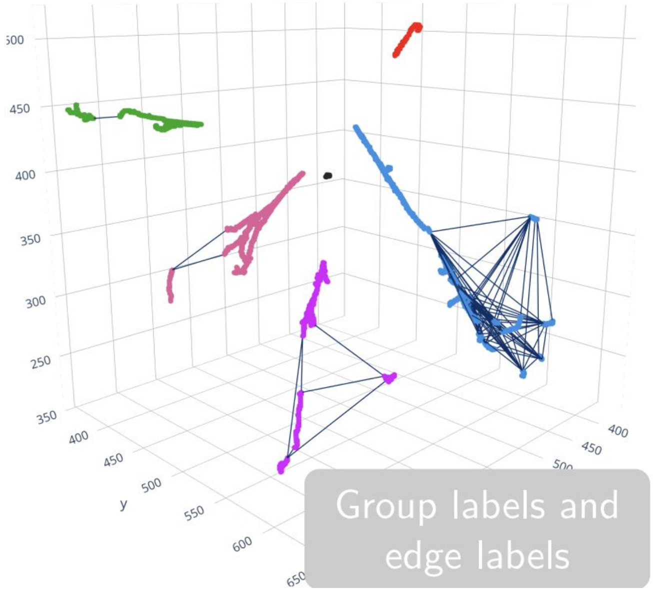
- Edge binary classification
- **Target**: 1 if same particle, 0 otherwise

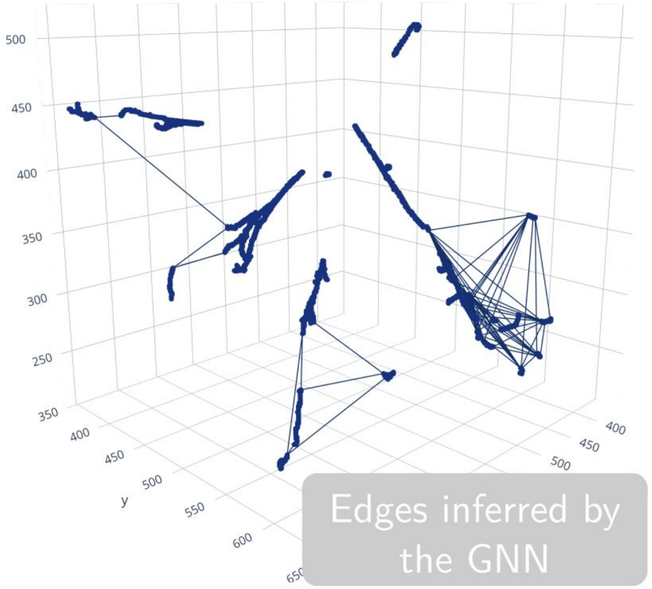


Aggregation

Edge selection

Find **connected components** and it's a done deal? **Not quite**…



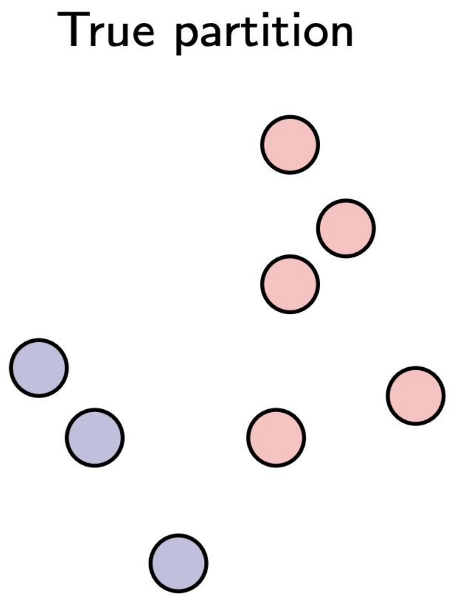


Aggregation

Edge selection

The network predicts a score matrix, **S**, which is proxy for the adjacency matrix, **A**

- What is the best partition, **g***?

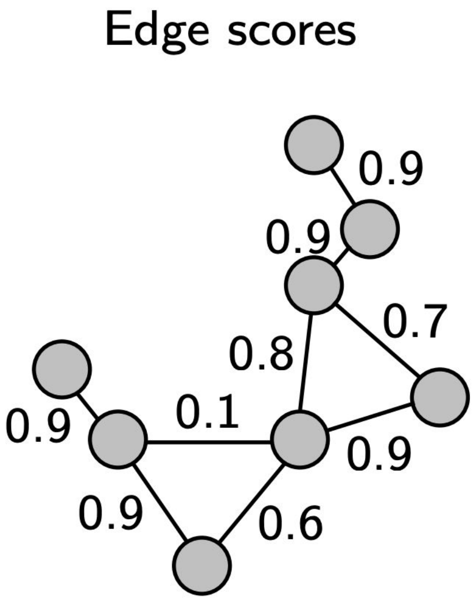


Aggregation

Edge selection

The network predicts a score matrix, **S**, which is proxy for the adjacency matrix, **A**

- What is the best partition, **g***?



Aggregation

Edge selection

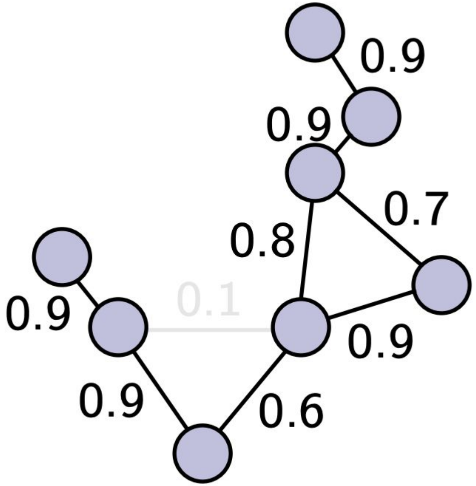
The network predicts a score matrix, **S**, which is proxy for the adjacency matrix, **A**

- What is the best partition, **g***?

We want to minimize the CE loss:

$$L(S|g) = -\frac{1}{N_e}\sum_{(i,j)\in E} \delta_{g_i,g_j}\ln(s_{ij}) + (1-\delta_{g_i,g_j})\ln(1-s_{ij})$$

Thresholded graph



$L \simeq 3.92$

Aggregation

Edge selection

The network predicts a score matrix, **S**, which is proxy for the adjacency matrix, **A**

- What is the best partition, **g***?

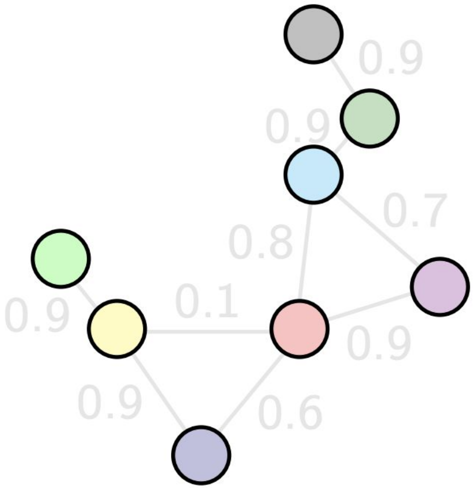
We want to minimize the CE loss:

$$L(S|g) = -\frac{1}{N_e} \sum_{(i,j)\in E} \delta_{g_i,g_j} \ln(s_{ij}) + (1-\delta_{g_i,g_j}) \ln(1-s_{ij})$$

G is the set of all possible partitions:

- Cannot bruteforce (B_{20} = 5x10^{13})
- Start with an empty graph

Empty graph



$$L \simeq 15.35$$

Aggregation

Edge selection

The network predicts a score matrix, **S**, which is proxy for the adjacency matrix, **A**

- What is the best partition, **g***?

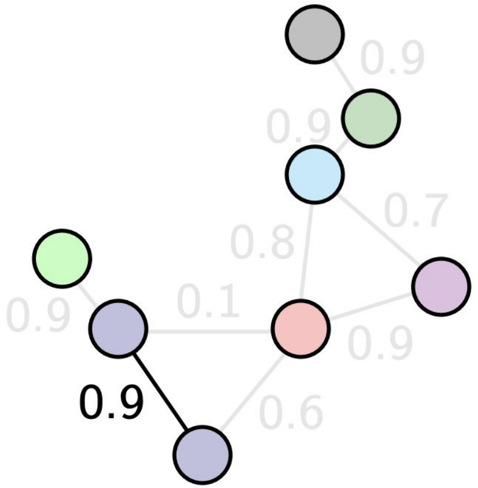
We want to minimize the CE loss:

$$L(S|g) = -\frac{1}{N_e} \sum_{(i,j)\in E} \delta_{g_i,g_j} \ln(s_{ij}) + (1-\delta_{g_i,g_j}) \ln(1-s_{ij})$$

G is the set of all possible partitions:

- Cannot bruteforce (B_{20} = $5\text{x}10^{13}$)
- Start with an empty graph
- Iteratively add most likely edges until the next best edge is < 0.5

First edge



$L \simeq 13.15$

Aggregation

Edge selection

The network predicts a score matrix, **S**, which is proxy for the adjacency matrix, **A**

- What is the best partition, **g***?

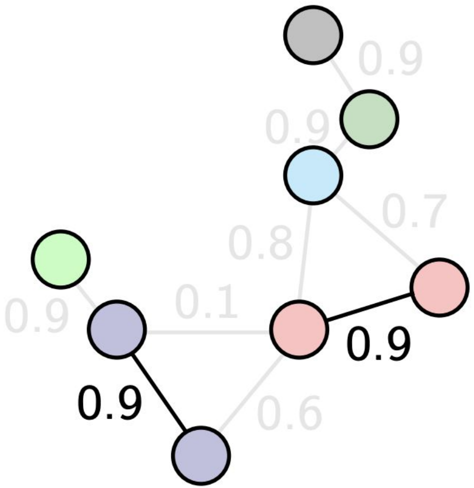
We want to minimize the CE loss:

$$L(S|g) = -\frac{1}{N_e} \sum_{(i,j)\in E} \delta_{g_i,g_j} \ln(s_{ij}) + (1-\delta_{g_i,g_j}) \ln(1-s_{ij})$$

G is the set of all possible partitions:

- Cannot bruteforce ($B_{20} = 5\text{x}10^{13}$)
- Start with an empty graph
- Iteratively add most likely edges until the next best edge is < 0.5

Second edge



$L \simeq 10.95$

Aggregation

Edge selection

The network predicts a score matrix, **S**, which is proxy for the adjacency matrix, **A**

- What is the best partition, **g***?

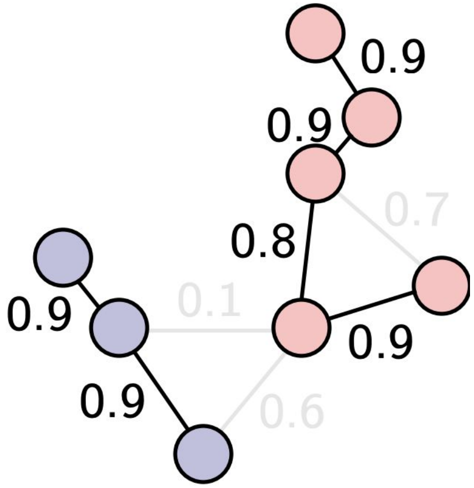
We want to minimize the CE loss:

$$L(S|g) = -\frac{1}{N_e} \sum_{(i,j)\in E} \delta_{g_i,g_j} \ln(s_{ij}) + (1-\delta_{g_i,g_j}) \ln(1-s_{ij})$$

G is the set of all possible partitions:

- Cannot bruteforce (B_{20} = $5\text{x}10^{13}$)
- Start with an empty graph
- Iteratively add most likely edges until the next best edge is < 0.5

Optimized partition

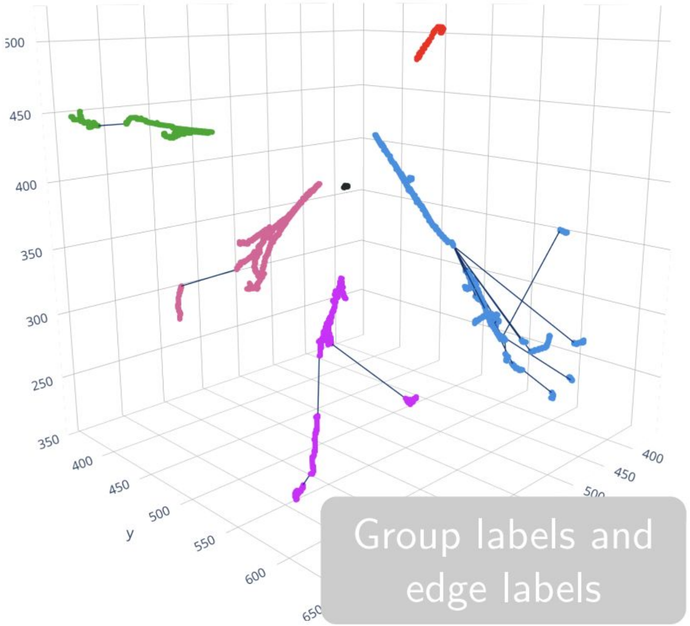


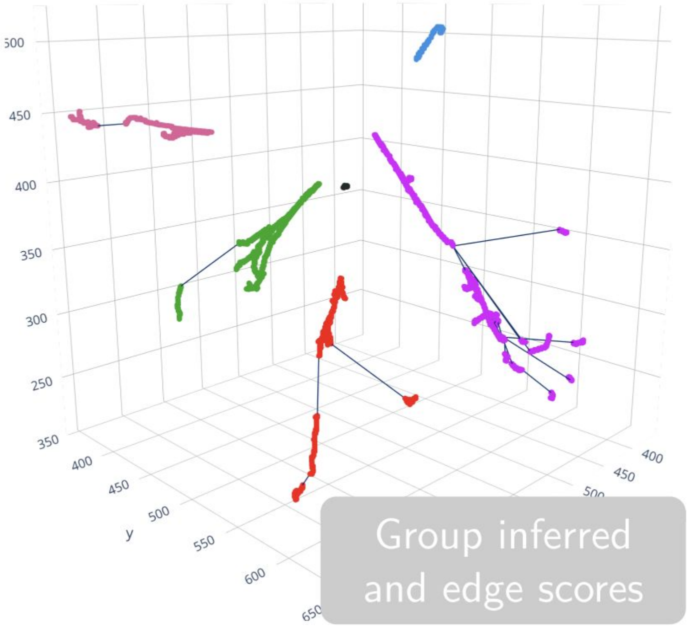
$L \simeq 2.13$

Aggregation

Edge selection

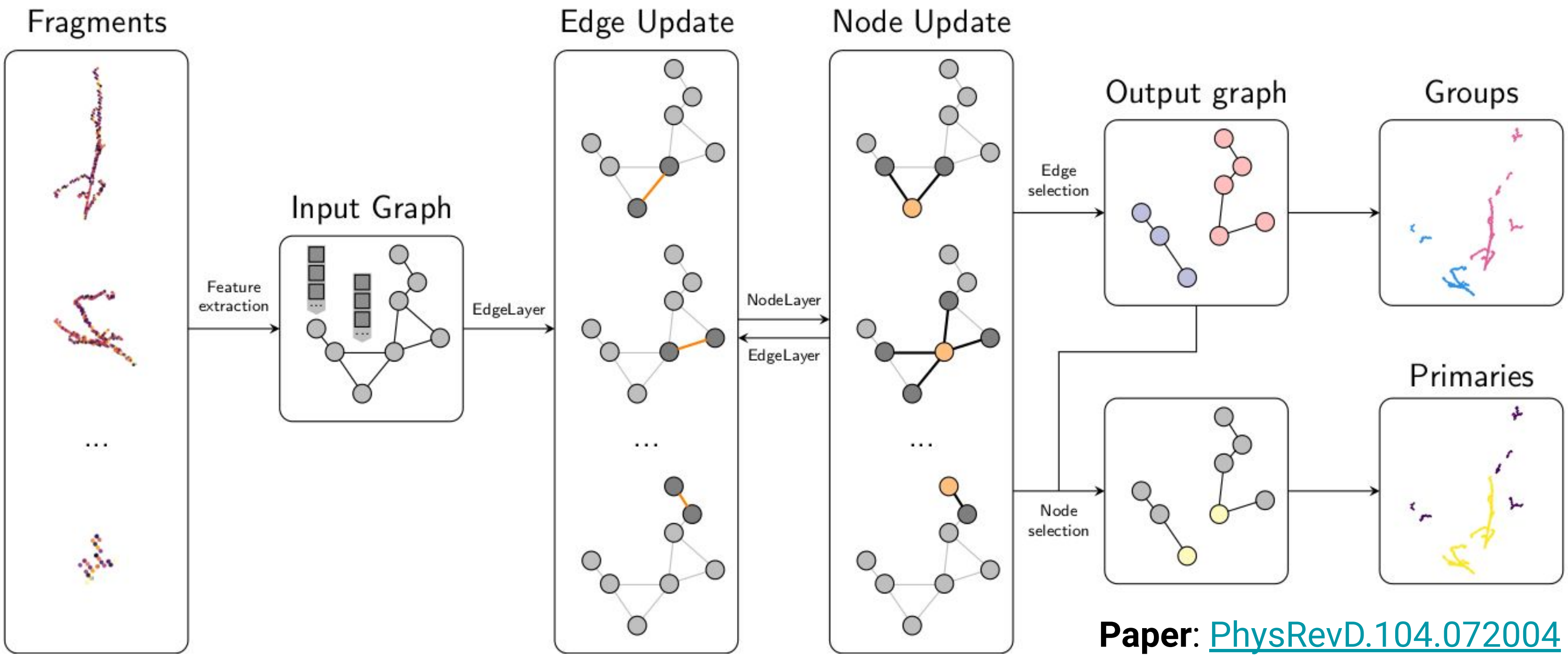
This automatically gets rid of spurious positive edges!





Graph Particle Aggregator (GrapPA)

Architecture



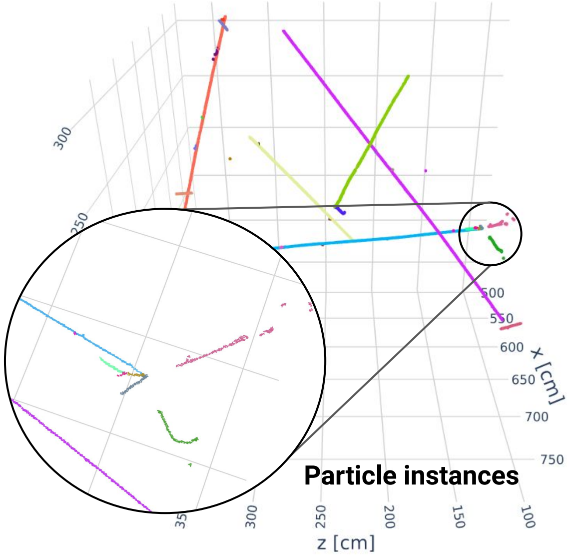
Paper: [PhysRevD.104.072004]

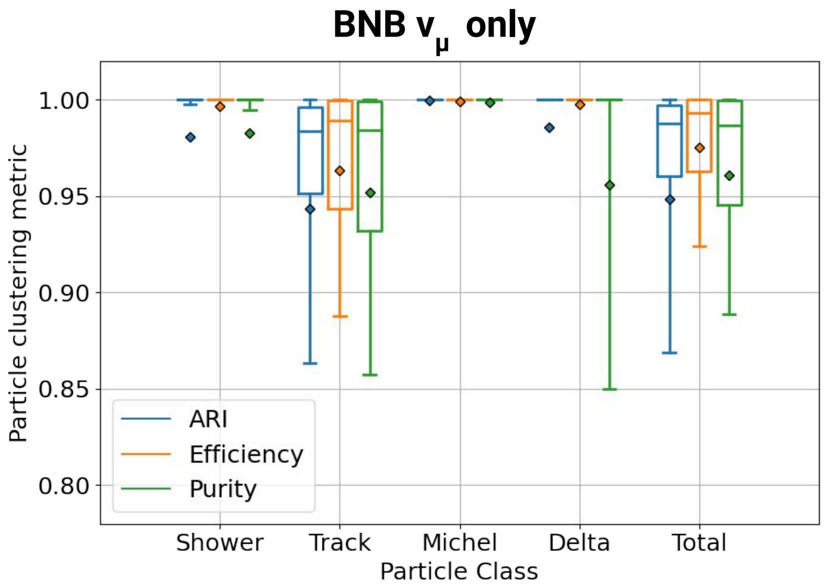
Graph Particle Aggregator (GrapPA)

Fragment aggregation performance

Build particles from individual particle fragments

- **Excellent** clustering performance. Tracks hardest, as expected





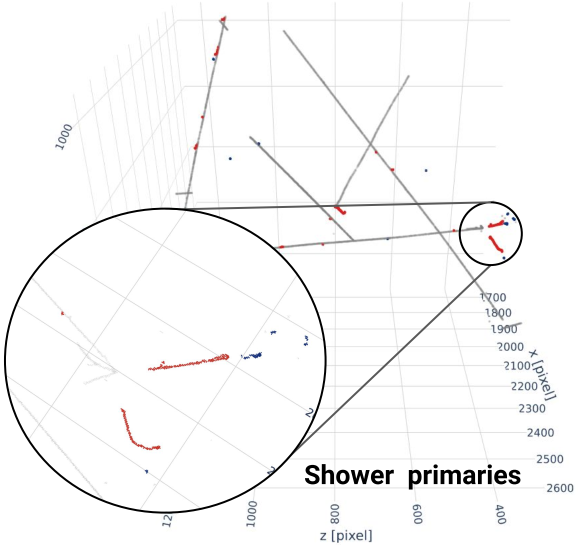
Paper: PhysRevD.104.072004

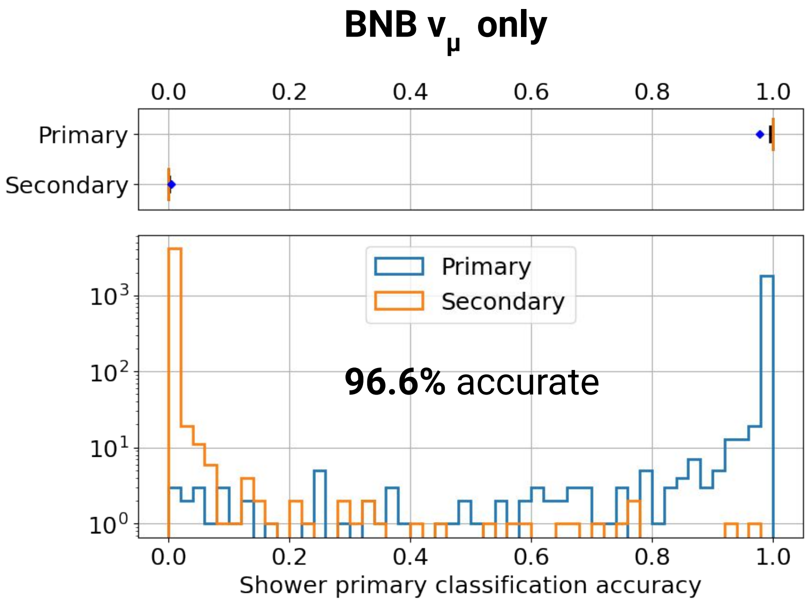
Graph Particle Aggregator (GrapPA)

Shower primary identification

Classify each shower fragment as either **primary** or **secondary**

- **Very reliably** finds the start of a shower





Paper: PhysRevD.104.072004

Aggregation

Interaction Aggregation

Now we wanna go further and:

- Cluster particles into **interactions**
- Classify particles: **species** and **primary**

Very similar problem to fragment clustering

- **Input**: ~~fragment~~ -> particle
- **Edge input**: ~~fragment expected gaps~~ -> particle expected gaps
- **Edge target**: ~~particle~~ -> interaction
- **Node target**: ~~shower primary~~ -> (PID, primary)

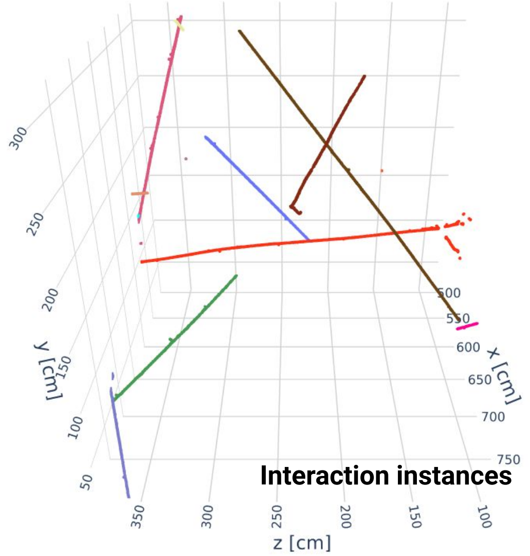
Reuse GrapPA and train on this task instead!

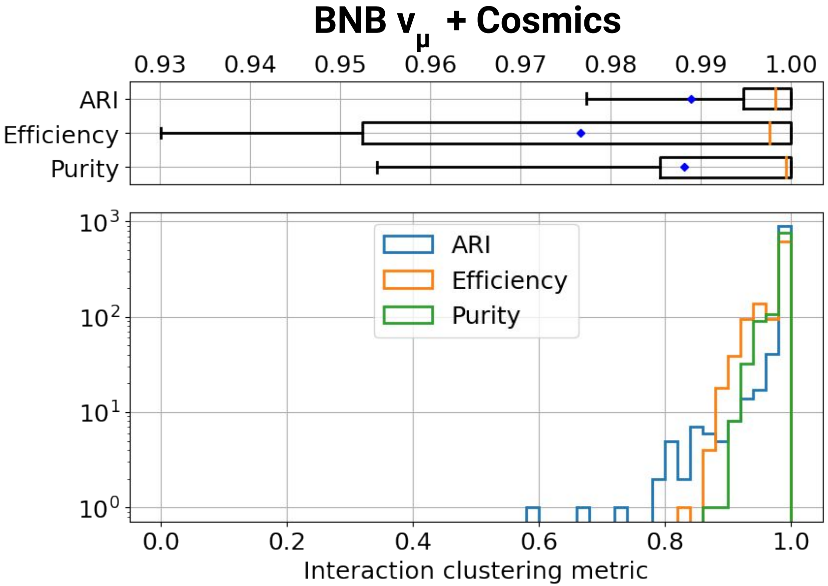
Aggregation

Graph edge classification

Build interactions from individual particles

- **Easily** cluster disjoint particles, most inefficiencies come from n activity





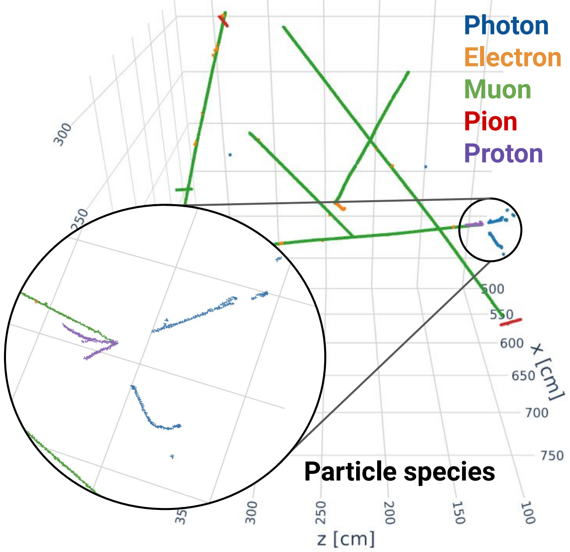
Paper: PhysRevD.104.072004

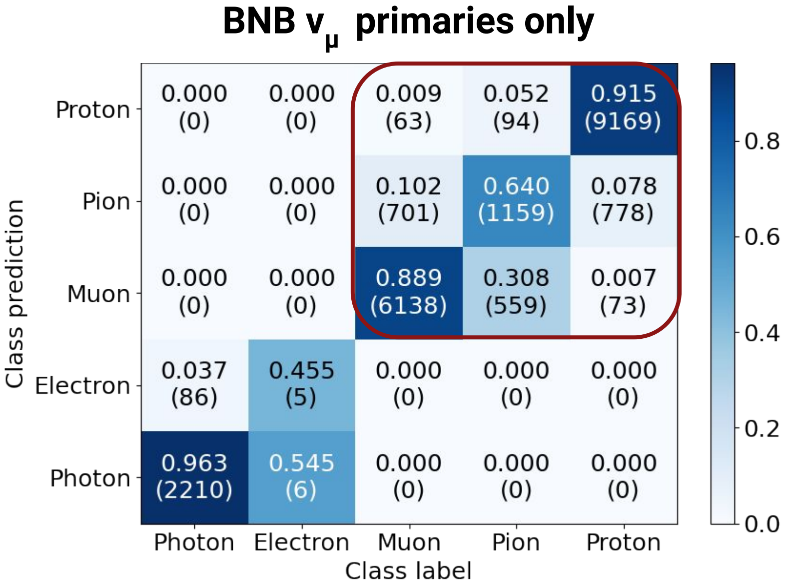
Particle Identification

Graph node classification

Particle species much easier to **infer in context**

- Michel decays, secondary hadrons, shower conversion gaps, etc.



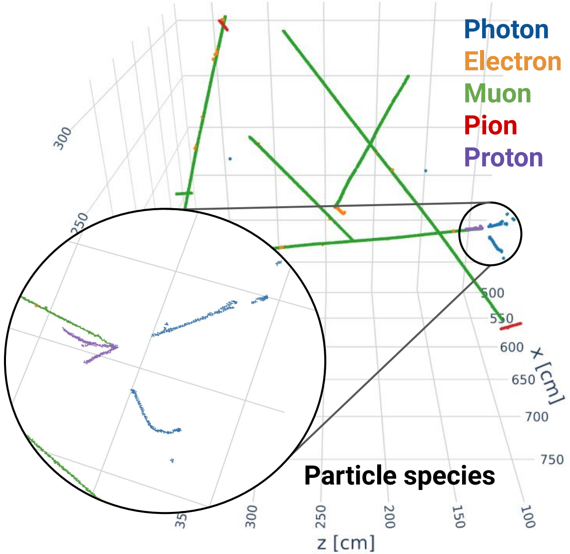


Particle Identification

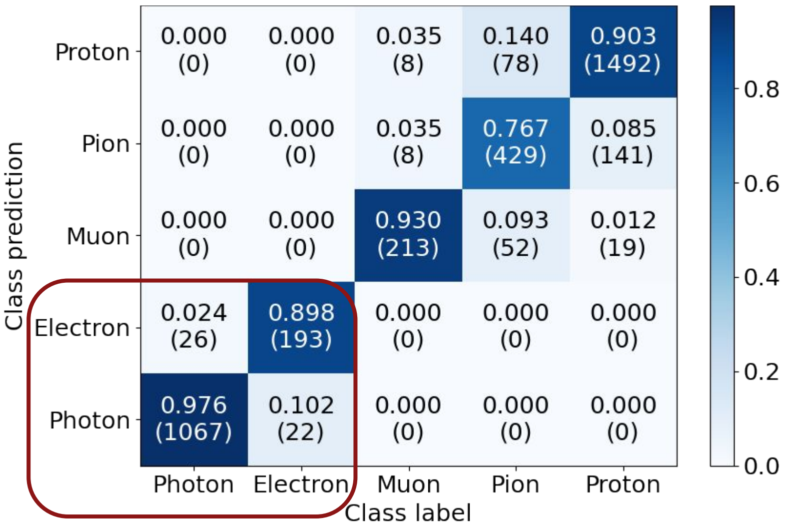
Graph node classification

Particle species much easier to **infer in context**

- Michel decays, secondary hadrons, shower conversion gaps, etc.



Generic dataset (particle bombs)

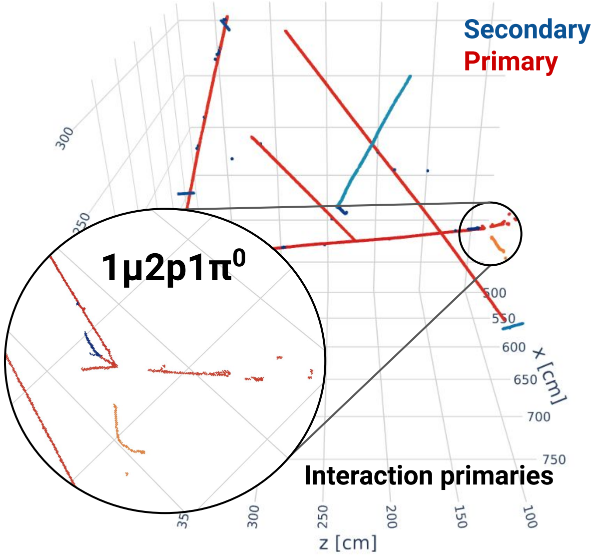


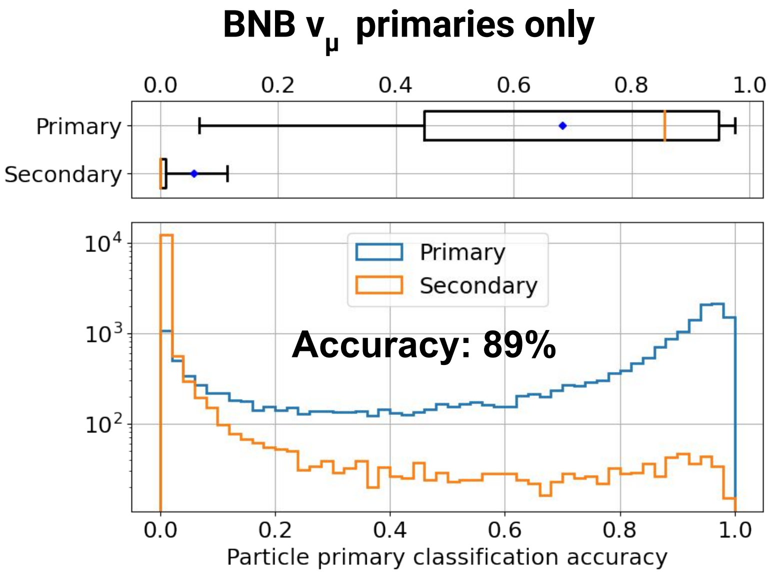
Primary Identification

Graph node classification

Important to know **which particle originate from the vertex**

- Central to any **exclusive analysis** (study specific channels)



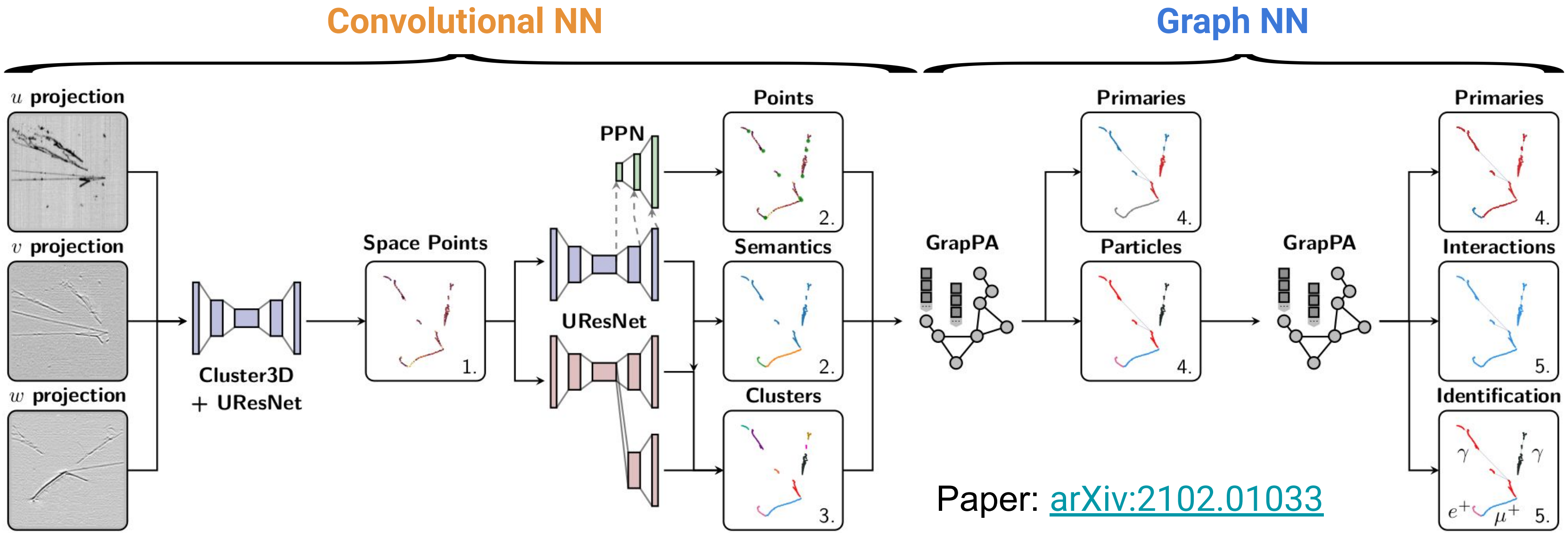


Reconstruction in LArTPCs

Full Reconstruction Chain Architecture

End-to-end ML-based reconstruction chain

- **UResNet** for pixel feature extraction, **GrapPA** for superstructure formation



Paper: [arXiv:2102.01033](https://arxiv.org/abs/2102.01033)

Reconstruction in LArTPCs

Scalability

A fair bit of work invested in speeding up the execution speed. On **ICARUS**:

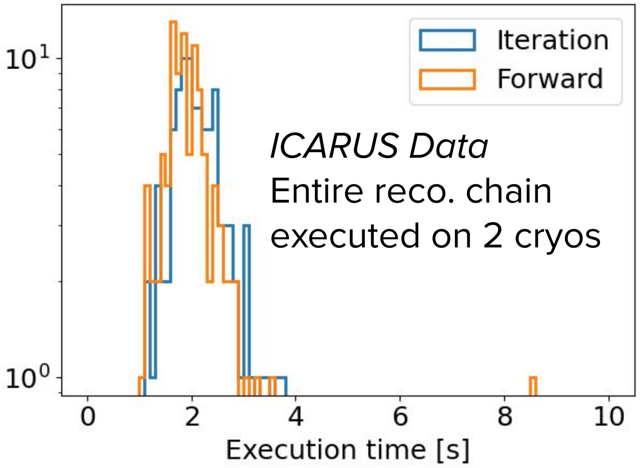
- ~2 M input space points/event
- O(1) M edges in the aggregator graphs
- **TPC reco: 2 s/event on an A100**

Very cheap to run on large datasets:

- **1 year of ICARUS beam-on data** can be reconstructed **in 1 day** with <200 A100s
- Perlmutter (NERSC): > 6000 A100s

Only **scales** with **space point count**

- Very cheap to run on DUNE-FD

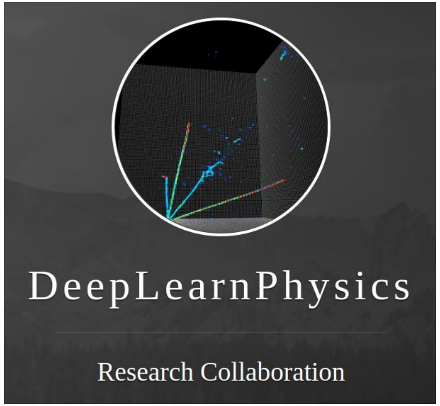


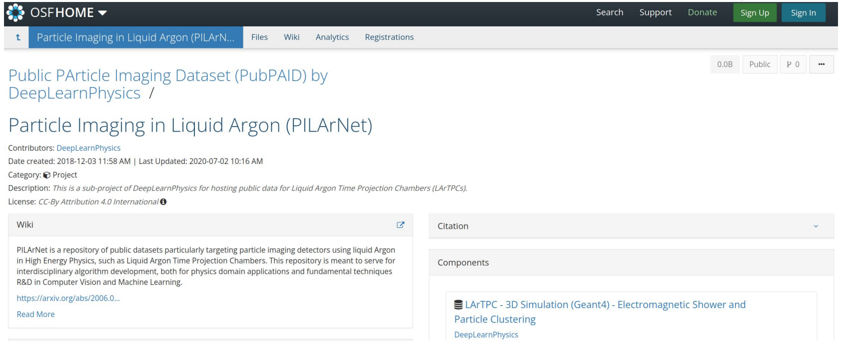
Reconstruction in LArTPCs

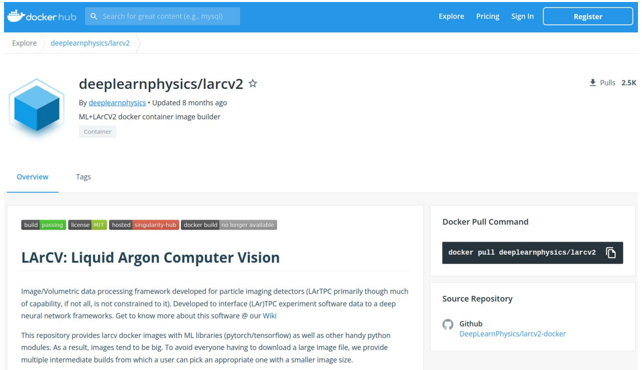
Open-source ecosystem

DeepLearnPhysics collaboration (ML techniques R&D)

- Public LAr simulation
  - Potential for open real data from prototypes
- Shared software dependencies with Docker/Singularity
- Open reconstruction software on GitHub
- Fully **reproducible results**
  - Readers have reproduced PhysRevD.102.012005







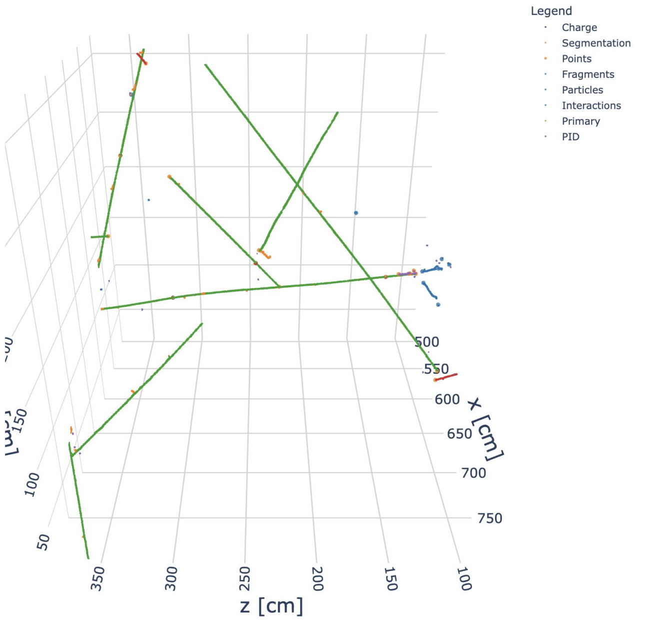
Conclusions

SLAC NATIONAL ACCELERATOR LABORATORY

Takeaways

End-to-end ML-based reconstruction chain mature and functional

- Used on **ICARUS** sim./data and **DUNE-ND** (high neutrino pileup) sim. **today**
- Check out this ICARUS **interactive reconstructed event** !



Exciting times:

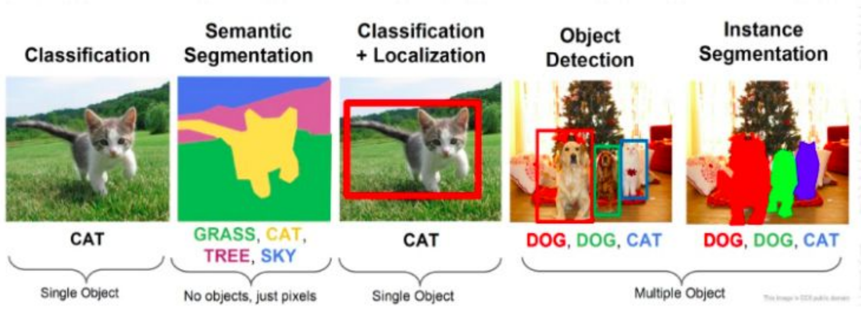
- Many great analysis underway
- A lot of data in the can, ready to go
- Let's bring it home!

Backup Slides

Reconstruction in LArTPCs

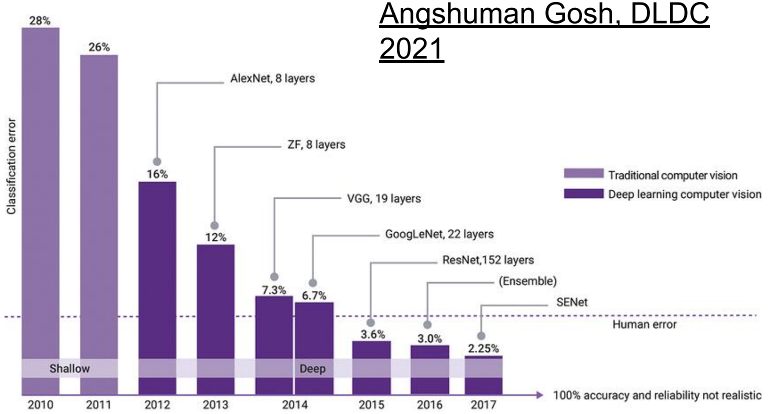
Machine Learning in Computer Vision (CV)

ML is the state-of-the-art in CV, i.e. extracting high level information from images
- ML revolutionized accuracy on image processing tasks
- Should **leverage those techniques in HEP**



Stanford, CS231

Angshuman Gosh, DLDC 2021



Reconstruction in LArTPCs

Hierarchical feature extraction

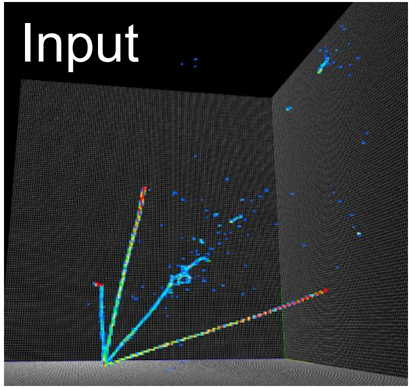
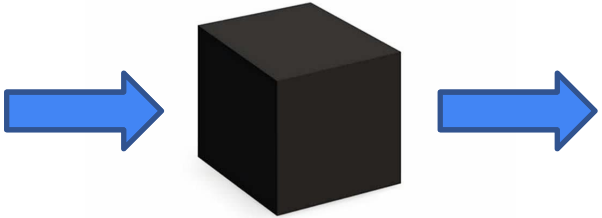
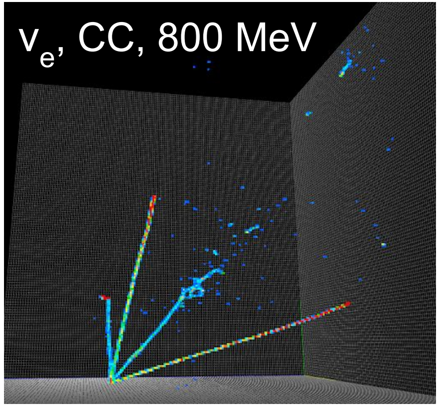


Image Classifier (CNN)

- What to do with > 1 interaction ?
- What if it fails ? Why ?
- What behavior if unknown interaction?



77



Reconstruction in LArTPCs

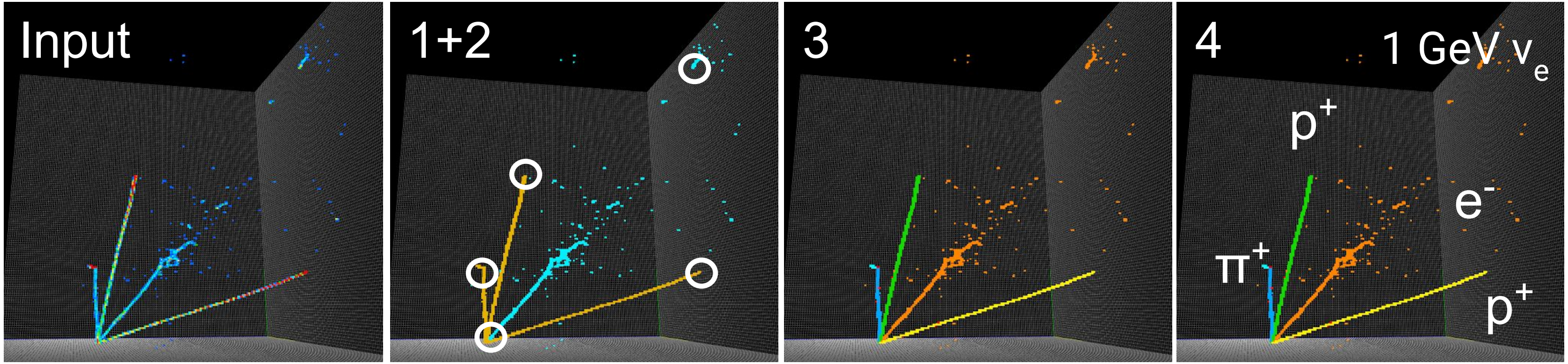
Hierarchical feature extraction

What is relevant to pattern recognition in a detailed interaction image?

1. Separate topologically distinguishable **types of activity**
2. Identify **important points** (vertex, start points, end points)
3. Cluster individual **particles** (tracks and full showers)
4. Cluster **interactions**, identify **particle properties** in context

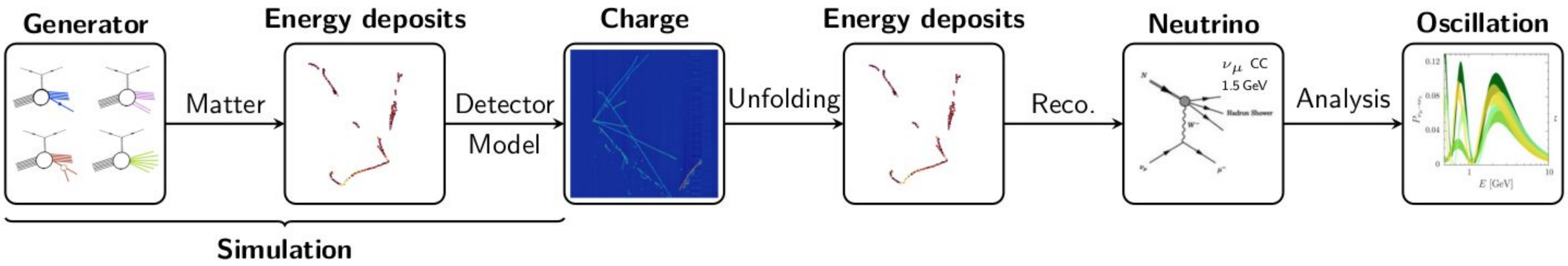
Pixel-level (1–2)

Cluster-level (3–4)



Non-Reconstruction ML Efforts in LArTPCs

Future prospects



So far, we have tackled the **reconstruction challenge**, what's next?

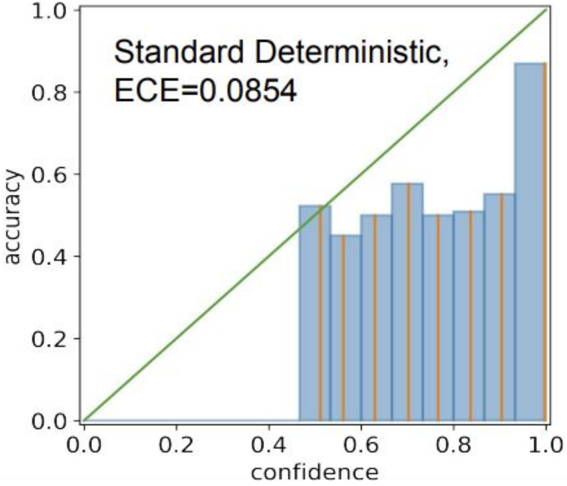
- Can we go beyond "most likely" prediction and **quantify an uncertainty** ?
- Can we **mitigate differences** between simulation and data ?
- Can we **optimize detector modeling from data** and remove the issue altogether ?
- Can we **unfold detector effects** directly ? Yes, **learn inverse function automatically!**
- Can we **learn physics** (generators) **from data** ? Yes and no

Uncertainty Quantification

Overview

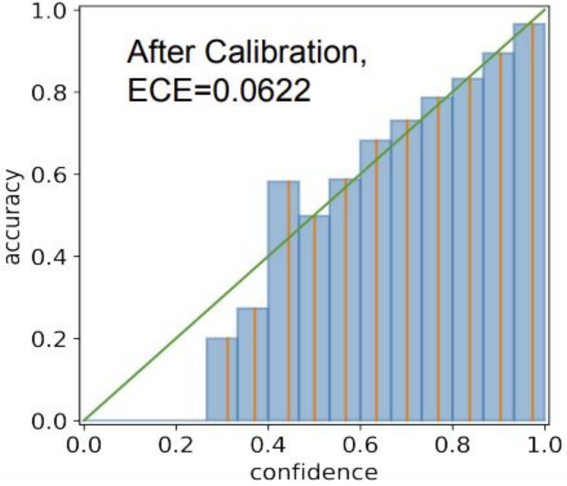
Goals of Uncertainty Quantification in Probabilistic Models:

- **Calibration**: Score p in [0,1] <=> probability p to be correct
- **Error detection**: Low confidence <=> large uncertainty



$$\hat{p}(x, T) = \max_i \frac{e^{l_i(x)/T}}{\sum_j e^{l_j(x)/T}}$$

Temperature Scaling



Uncertainty Quantification

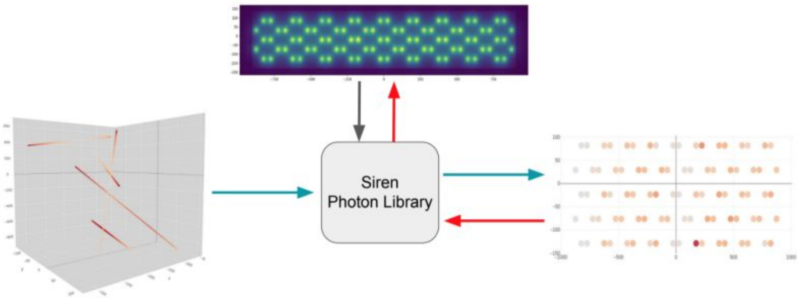
Photon visibility map

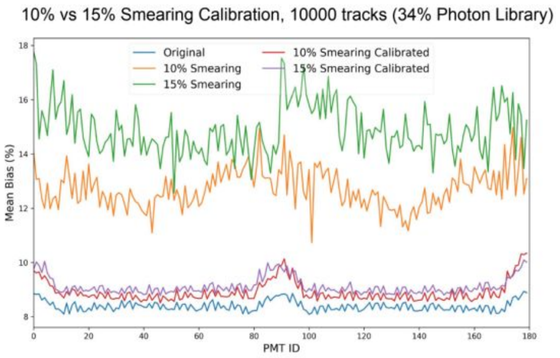
Can we make the **light simulation** differentiable ?

- Photon library maps x = (x, y, z) to visibility in each PMT (number of photons)
- Learn photon library using **scene representation** (SIREN): F (x, θ) differentiable

Calibration process: bias in library (offset in the actual visibility): θ ' = θ + δ

- Compare observed visibility to predicted visibility, use gradient descent to find θ ' !



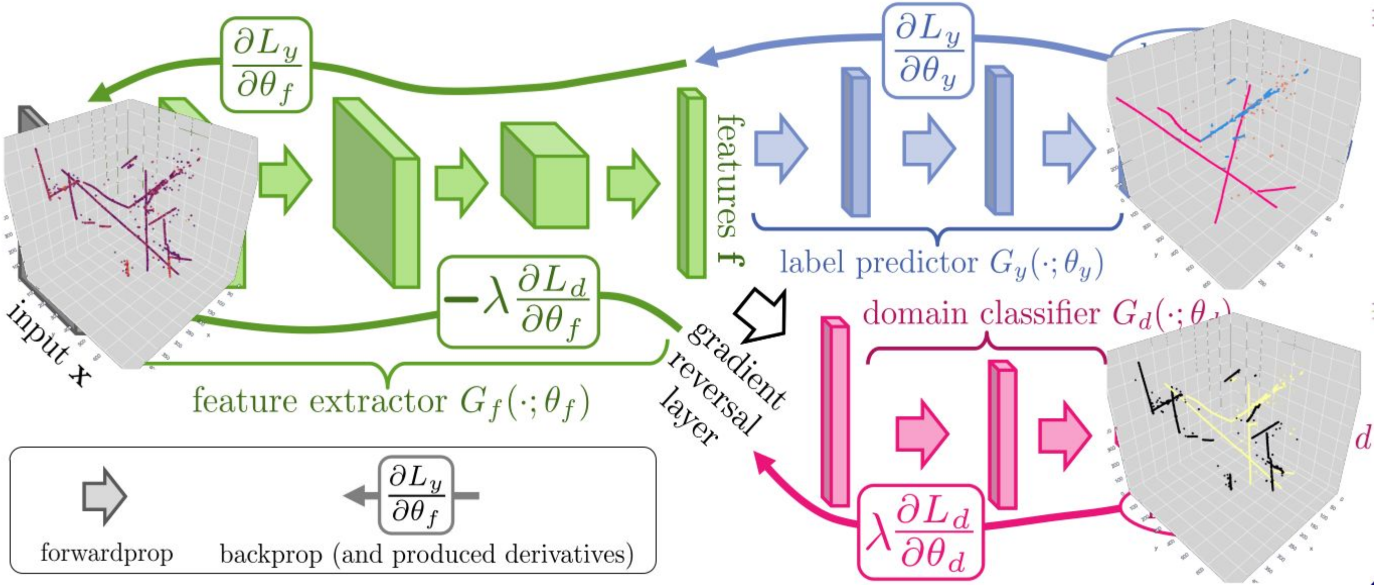


Domain Adversarial Training

Overview

Basics of **Domain Adversarial Networks**:

- Penalized for producing features that are different between



Domain Adversarial Training

Overview

Basics of **Domain Adversarial Networks**:

- Penalized for producing features that are different between sim. and data