Detecting Neutrons in LArTPCs with Machine Learning

DUNE Collaboration Meeting (01/25/2023)

Presented by *Nicholas Carrara* on behalf of the ***BLIP ML Group*** at **UC Davis[1], LANL[2], IIT[3], CIEMAT[4], SDSM[5], FNAL[6], NIKHEF[7], FSU[8], UNIBE[9],** :

Nicholas Carrara[1], David Rivera[2], Laura Perez-Molina[4],

Marjolein van Nuland[7], Georgette Kufatty[8], Jan Kunzmann[9], Livio Calivers[9],

Michael Mulhearn[1], Emilija Pantic[1], Robert Svoboda[1], Jingbo Wang[5]

Yashwanth Bezawada[1], Junying Huang[1], Will Forman[3], Walker Johnson[5], Ajib Paudel[6],

Why are neutrons important?

To turn neutrino physics into a precision science, we need to understand complex neutrino-nucleus interactions (specifically on Ar):

- Neutrons carry away a ***large fraction***[1] of energy.
- Neutron final states are ***model dependent***.
- Neutrons are ***difficult to detect*** in LAr.

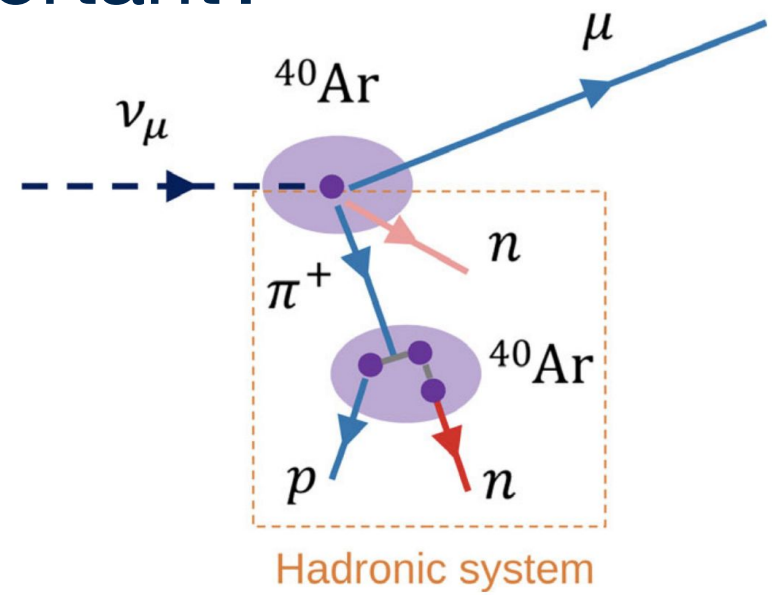


Figure from A. Friedland and W. Li (2019).

Neutrons are also important for low-energy physics in LAr:

- e.g., modeling **supernova** and **solar (**and now **galactic**[2]**)** neutrino physics.

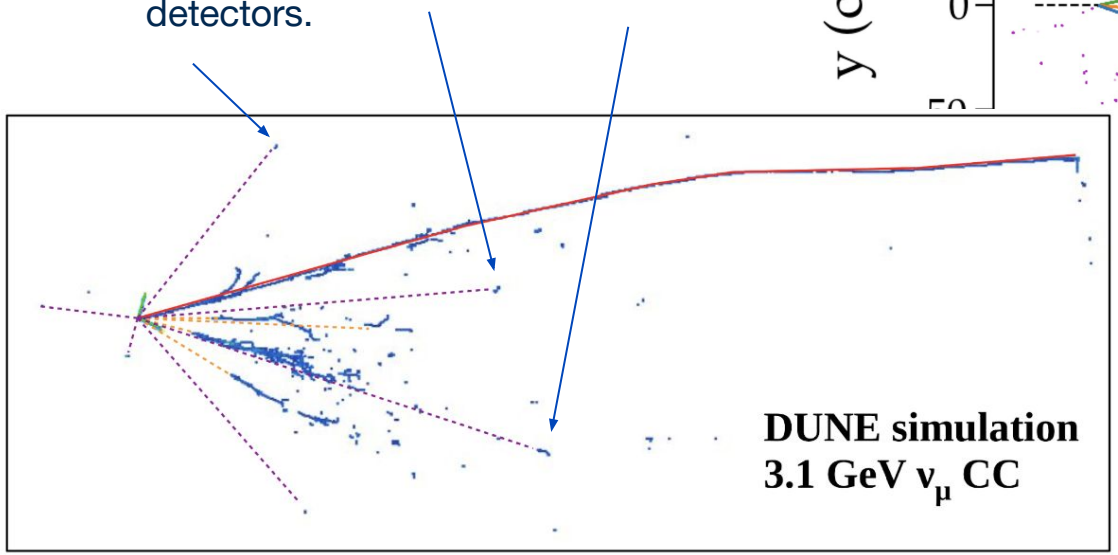
1 A. Friedland and W. Li, *Understanding the energy resolution of liquid argon neutrino detectors*, Physical Review D, **99**, 2019 (https://arxiv.org/abs/1811.06159)
2 ICECUBE Collaboration, *Observation of high-energy neutrinos from the Galactic plane*, Science, **380**, 2023 (https://www.science.org/doi/10.1126/science.adc9818)

UCDAVIS

Difficult to detect?

It is necessary to be able to account for the neutron “missing energy” in order to make precision oscillation measurements (production and transport).

- Neutrons show up as small ***blips*** in LAr detectors.



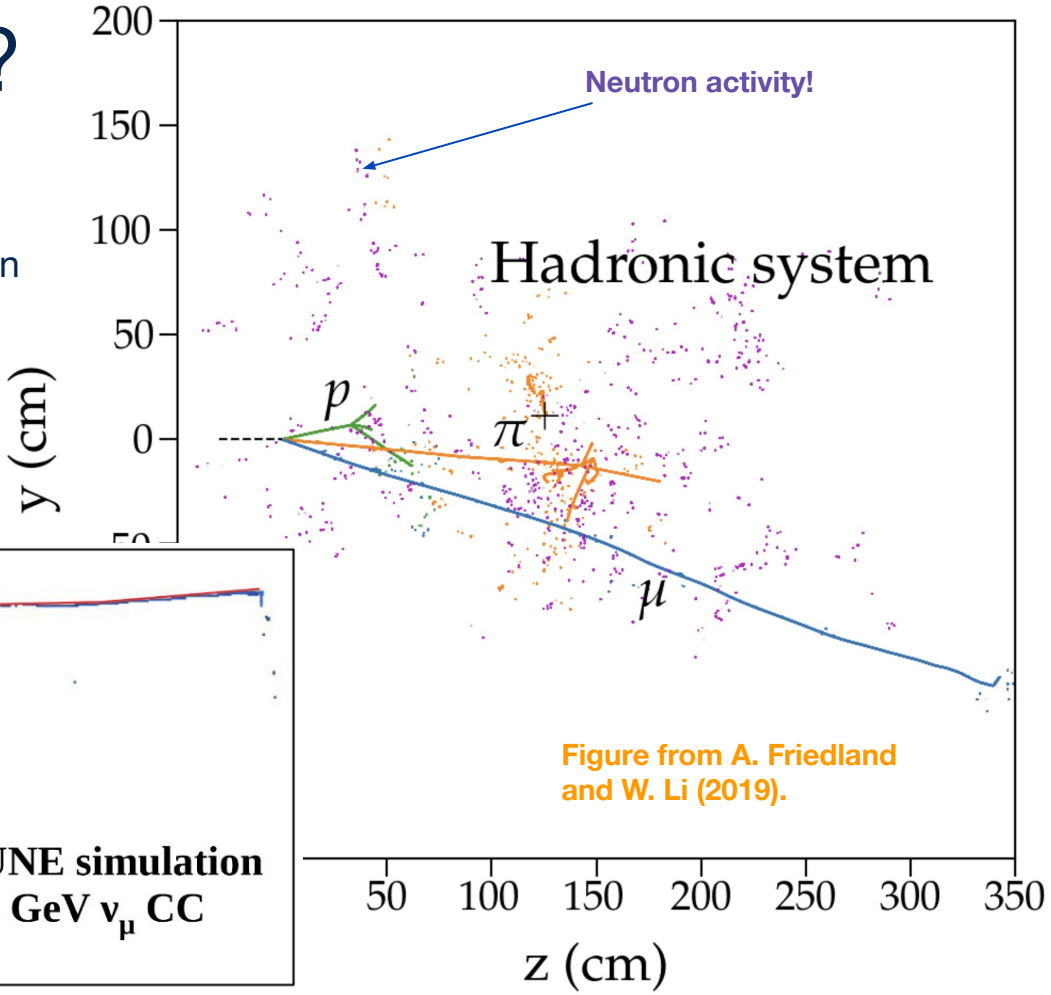


Figure from A. Friedland and W. Li (2019).

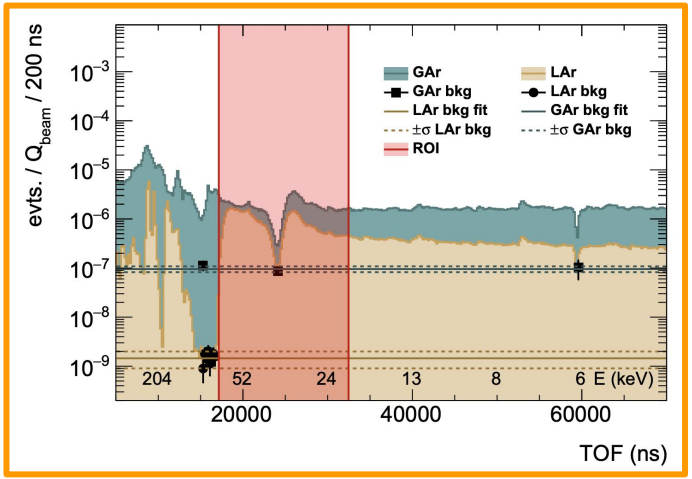
Neutron Calibration (PNS)

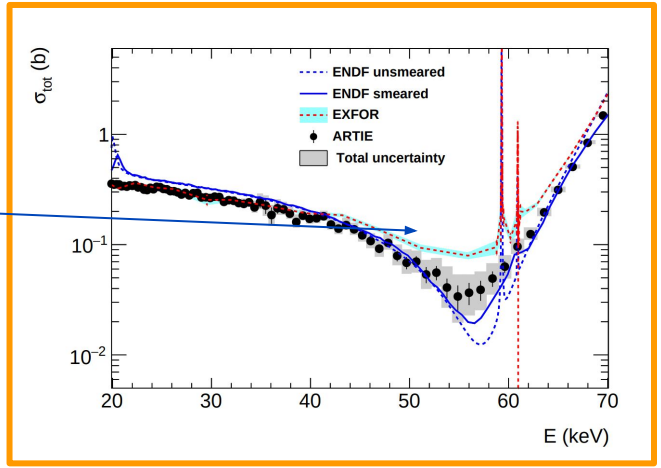
Benefits of low-energy neutrons for calibration:

- ***Standard Candle*** - Neutron captures on Ar-40 emit a 6.1 MeV gamma cascade.

$$\mathrm{n} + {}^{40}\mathrm{Ar} = {}^{41}\mathrm{Ar} + \{\gamma_j\}$$
$$\sum_j E(\gamma_j) \approx 6.1\mathrm{MeV}$$

- ***Scattering Length*** - Some percentage of neutrons above 57 keV will fall into the resonance well.
  - Average *fractional energy loss* is ~4.8%.
  - The *effective scattering length* is ~30 m.
  - The resonance well has been measured by the ARTIE[1] experiment at LANL, with a ***follow-up*** planned for this year (**ARTIE-II**).





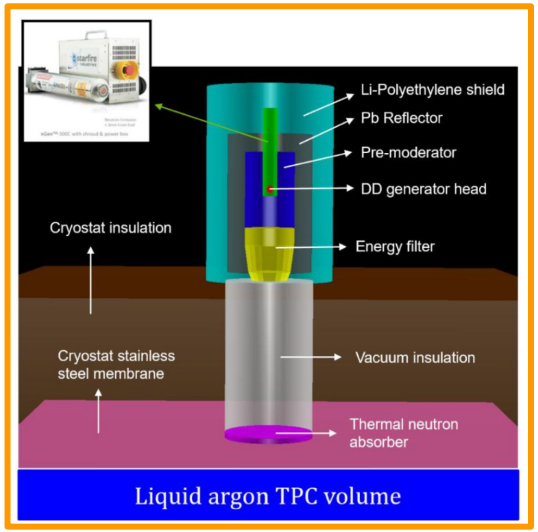
3 *Measurement of the total neutron cross section on argon in the 20 to 70 keV energy range,* The ARTIE Collaboration, In review at PRL, 2023, (https://arxiv.org/abs/2212.05448).

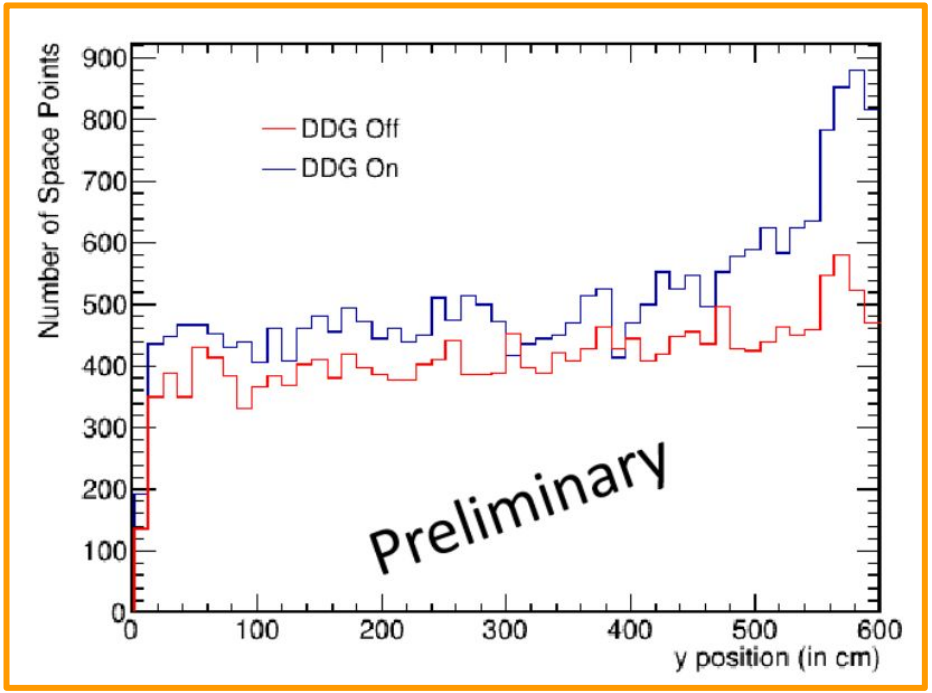
UCDAVIS

How Can We Isolate Captures?

A **pulsed neutron source**, such as a deuterium-deuterium generator (DDG), can create a *mono-energetic* spray of low-energy neutrons.

- A DDG was used in ProtoDUNE-I, from which the neutrons could be seen in the detector reconstruction.
- So far, we have not had the ability to isolate ***individual neutron captures***.





Work done by Y. Bezawada and J. Huang

UCDAVIS

Blip

Two neutron analyses

We introduce ***Blip***[2], a collection of ML algorithms for classifying (e.g. low energy) interactions in LArTPCs.

- Nicholas Carrara (UC Davis)
- David Rivera (LANL) [**BlipNet**]
- Laura Perez-Molina (CIEMAT) [**BlipNet**]
- Marjolein van Nuland (NIKHEF)[**BlipSegmentation**, Arrakis]
- Georgette Kufatty (FSU) [**BlipSegmentation**]
- Jan Kunzmann (UNIBE) [**BlipLight**]
- Livio Calivers (UNIBE) [**BlipLight**]
- Ajib Paudel (FNAL) [**BlipLight**]
- Yash Bezawada (UC Davis) [**BlipNet**/PNS]
- Junying Huang (UC Davis) [**BlipNet**/PNS]
- Will Foreman (IIT)
- Walker Johnson (SDSMT)

pion-neutron final states/blip classification

n-Ar inelastic cross-section measurement at 2x2 (B. Russell)

neutron capture reconstruction for calibration

wire detector reconstruction

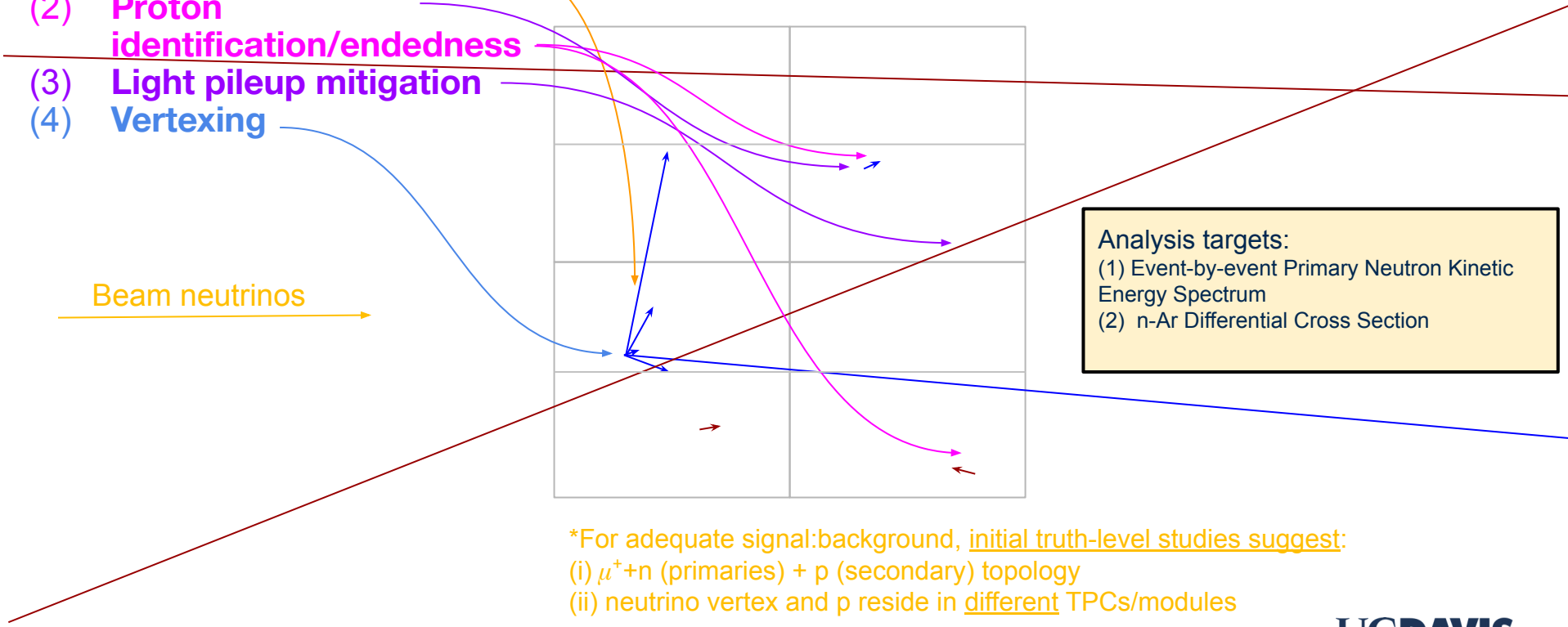
4 *BLIP, BLIP ML* Group (UC Davis), 2023, (https://github.com/Neutron-Calibration-in-DUNE/Blip).

UCDAVIS

n-Ar inelastic cross-section measurement at 2x2

Key analysis elements:

(1) **Neutrino identification**
(2) **Proton identification/endedness**
(3) **Light pileup mitigation**
(4) **Vertexing**



Analysis targets:
(1) Event-by-event Primary Neutron Kinetic Energy Spectrum
(2) n-Ar Differential Cross Section

*For adequate signal:background, initial truth-level studies suggest:
(i) μ^++n (primaries) + p (secondary) topology
(ii) neutrino vertex and p reside in different TPCs/modules

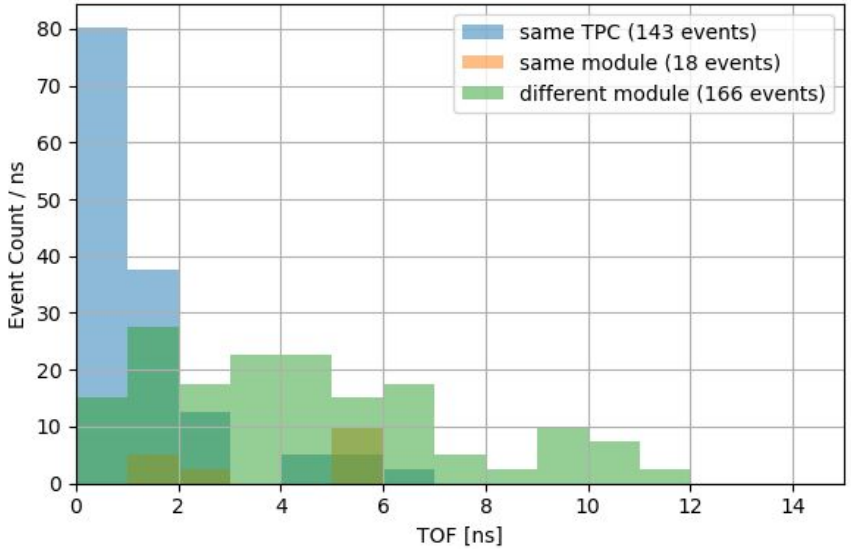
From 2x2 workshop (A. White)

Importance of Light Readout System:

1. **Light Pileup Mitigation**: need **time of flight** of neutron from proton track
   a. Need to have signal shape well-characterized
2. **T0** of interactions (neutrino & proton)
   a. Tof ~ **5 ns**
   b. Timing resolution ~ **2ns**
3. L (lever arm)
   a. Vertex resolution: to be investigated

$$T_n = (\gamma - 1) m_n$$

$$\text{where } \gamma = \frac{1}{\sqrt{1 - \left(\frac{l_{lever}}{c t_{tof}}\right)^2}}$$



From 2x2 workshop (A. White)

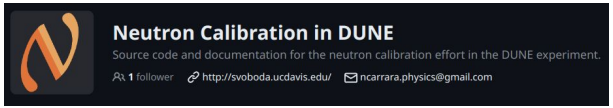
UCDAVIS

Reconstruction

Goal: Identify ***distended proton tracks*** originating from a ***neutrino interaction***.

The reconstruction consists of three stages,

1. Infer ***high-level variables*** from low-level detector reconstruction.
2. Select ***neutrino events*** with ***distended proton tracks***.
3. Infer ***inelastic n-Ar interactions*** from selection.



Key analysis elements:

(1) Neutrino identification
(2) Proton identification/endedness
(3) Light pileup mitigation
(4) Vertexing

From 2x2 workshop (A. White)

UCDAVIS

ArrakisND [N. Carrara, M. van Nuland]

ArrakisND works with h5flow to produce ***high-level variables*** for each reconstructed charge/light hit.

Each category represents a different ***scale*** and is constructed to be ***exhaustive*** and ***mutually exclusive***; (i.e. each hit MUST be described by ONE and ONLY ONE label in each category).

The categories are chosen to capture ***ALL*** relevant physical information.

From 2x2 workshop (A. White)

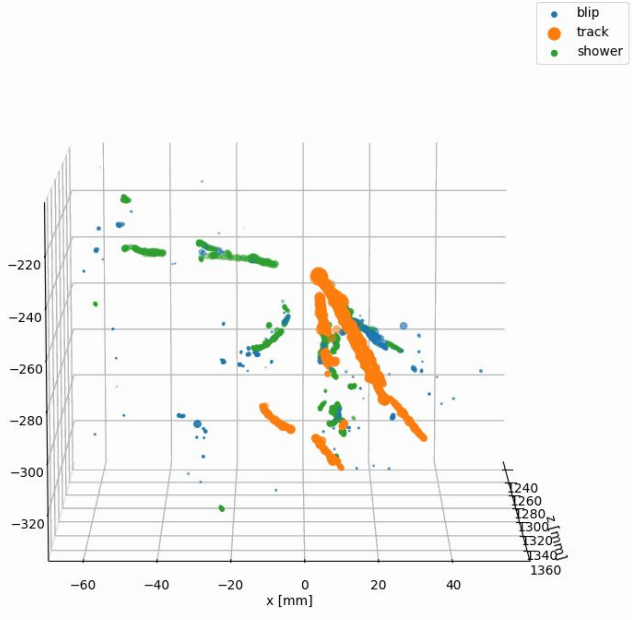
High-level features	Description
topology	topological descriptor of physics (e.g. blip, track, shower)
particle	the pdg code of the particle which caused the energy deposition
physics_micro	low-level descriptor of physics processes (e.g. mip_ionization, gamma_conversion, etc.)
physics_meso	mid-level descriptor of physics processes (e.g. hip, delta_electron, capture_gamma, etc.)
physics_macro	high-level descriptor of physics processes (e.g. cc_qe, neutron_capture, radiological, etc.)
unique_topology	unique identifier of individual topology instances
unique_particle	unique identifier of individual particle instances (i.e. track id)
unique_physics_*	unique identifier of individual physics instances

https://github.com/Neutron-Calibration-in-DUNE/ArrakisND/blob/main/arrakis_nd/dataset/common.py

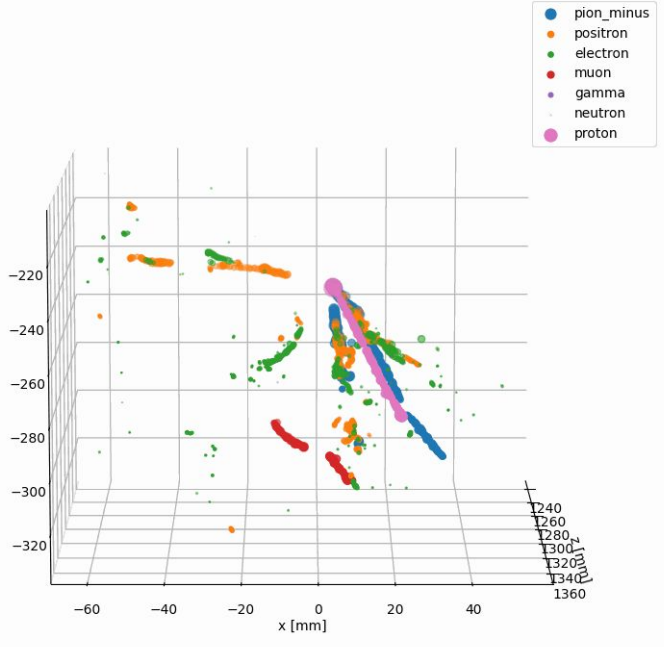
UCDAVIS

ArrakisND [N. Carrara, M. van Nuland]

Topology



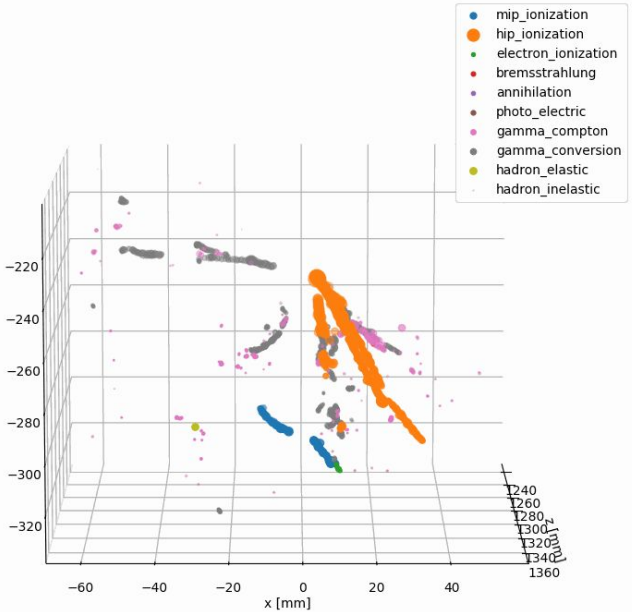
Particle



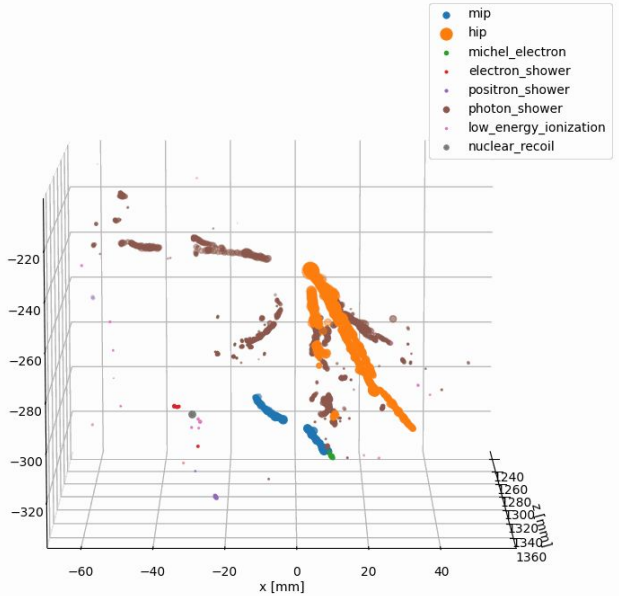
From 2x2 workshop (A. White)

ArrakisND [N. Carrara, M. van Nuland]

Physics Micro



Physics Meso



From 2x2 workshop (A. White)

ArrakisND [N. Carrara, M. van Nuland]

Work in progress:

- ❏ Add in ***track_topology*** variables (vertices, track begin/end, etc.).
- ❏ Add **CAF** variables for each event (tracks, showers, etc.)
- ❏ Add in ***light reco truth*** and ***low level variables*** for reconstruction.
- ❏ Finish ***physics_macro*** logic.

Arrakis is being optimized to run with the flow.

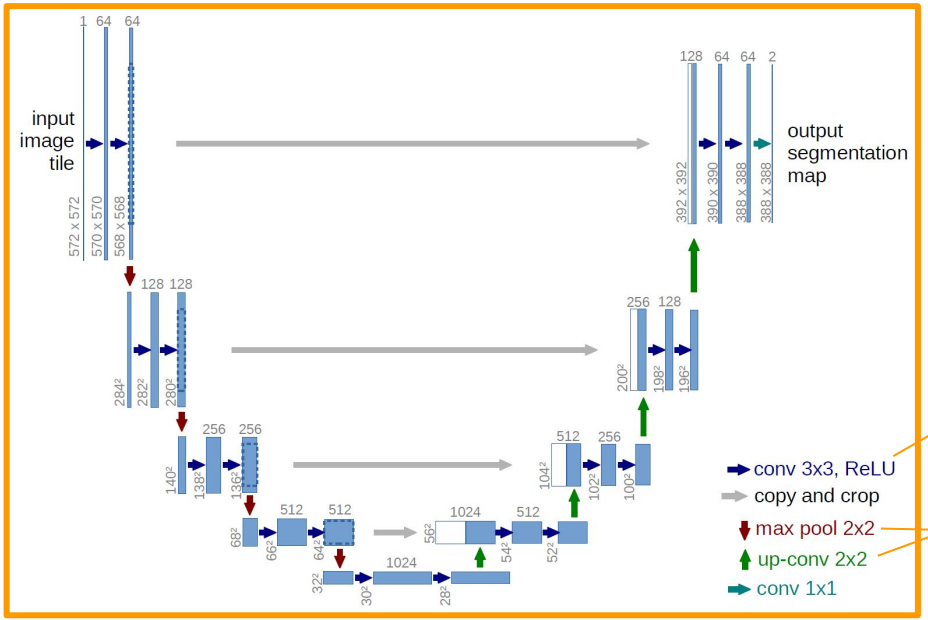
```
module:
  module_name:  'prep_data'
  module_type:  ['dataset']
  module_mode:  ['dataset_prep']
  gpu:          True
  gpu_device:   0
  verbose:      True

dataset:
  skip_processing:  false
  transform:        null
  pre_transform:    null
  pre_filter:       null
  dataset_type:   "tpc"
  dataset_folder: "/local_data/"
  dataset_files:  [
    "MiniRun4_1E19_RHC.flow.00000.FLOW.arrakis_nd.npz",
    "MiniRun4_1E19_RHC.flow.00001.FLOW.arrakis_nd.npz",
    "MiniRun4_1E19_RHC.flow.00002.FLOW.arrakis_nd.npz",
    "MiniRun4_1E19_RHC.flow.00003.FLOW.arrakis_nd.npz",
    "MiniRun4_1E19_RHC.flow.00004.FLOW.arrakis_nd.npz",
    "MiniRun4_1E19_RHC.flow.00005.FLOW.arrakis_nd.npz",
    "MiniRun4_1E19_RHC.flow.00006.FLOW.arrakis_nd.npz",
    "MiniRun4_1E19_RHC.flow.00007.FLOW.arrakis_nd.npz",
    "MiniRun4_1E19_RHC.flow.00008.FLOW.arrakis_nd.npz",
    "MiniRun4_1E19_RHC.flow.00009.FLOW.arrakis_nd.npz",
    "MiniRun4_1E19_RHC.flow.00010.FLOW.arrakis_nd.npz",
    "MiniRun4_1E19_RHC.flow.00011.FLOW.arrakis_nd.npz"
    "MiniRun4_1E19_RHC.flow.00012.FLOW.arrakis_nd.npz",
    "MiniRun4_1E19_RHC.flow.00013.FLOW.arrakis_nd.npz",
    "MiniRun4_1E19_RHC.flow.00014.FLOW.arrakis_nd.npz"
  ]
  variables:
    positions:      ['x', 'y', 'z']
    features:       []
    classes:        ['particle', 'topology', 'physics_micro', 'physics_meso']
    labels:         []
    clusters:       []
    hits:           []
    position_normalization:   []
    features_normalization:   []
    classes_mask: []
    labels_mask:  [[]]
    voxelization: [1, 1, 1]

  weights:
    class_weights:  []
    sample_weights: False      You, 3 weeks ago • some changes
```

From 2x2 workshop (A. White)

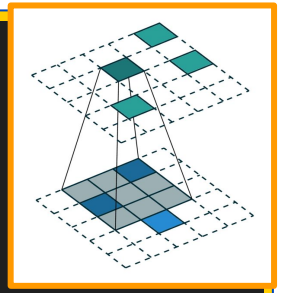
BlipSegmentation [G. Kuffaty]

Blip is run using a config file structure by specifying ***modules*** (such as the “ml” module).

The simplest model is a UNet structure which can be parameterized at run time.




```
module:
  module_name:  'optimize_blip_segmentation'
  module_type:  ['ml']
  module_mode:  ['training']
  gpu:          true
  gpu_device:   0
  verbose:      true
  debug:        true

dataset:
  dataset_params:   ""
  dataset_type:     "tpc"
  skip_processing:  true

loader:
  loader_type:          "minkowski"
  quantization_mode:    "random_subsample"
  minkowski_algorithm:  "speed_optimized"
  batch_size:       16
  test_split:       0.1
  test_seed:        42
  validation_split: 0.3
  validation_seed:  42
  num_workers:      4

training:
  iterations:   1       You, 3 weeks ago • some changes
  epochs:       100
  checkpoint:   25
  progress_bar: "all"     # train, validation, test, all
  rewrite_bar:      false # wether to leave bars after each epoch
  save_predictions: false # wether to save network outputs in original file
  no_timing:    true      # wether to keep timing/memory info in callback
  skip_metrics: true      # wether to skip metrics except for testing sets
  seed:         0

model:
  model_type:       "single"
  BlipSegmentation:
    in_channels:      1
    out_channels:     [5]
    classifications:  ['particle', 'topology', 'physics_micro', 'physics_meso']
    filtrations:      [64, 128, 256, 512]
    residual:         true
    sparse_conv_params:
      kernel_size:     3
      stride:     1
      dilation:   1
      activation: "prelu"
      dimension:  3
      batch_norm: True
      num_of_convs: 2
      dropout:    0.1
    conv_transpose_params:
      kernel_size: 2
      stride:     2
      dilation:   1
      dimension:  3
    max_pooling_params:
      kernel_size: 2
      stride:     2
      dilation:   1
      dimension:  3
```



Hyper-parameter Tuning [N. Carrara, G. Kuffaty]

Blip is configured through a container on NERSC.

A Blip program can generate a collection of config files with varying parameters for hyperparameter tuning.

Several models can be deployed for training simultaneously on Perlmutter. The results are then analyzed after training.

infophysics/blip ☆

By infophysics • Updated 2 days ago

Docker image for the blip package.

Image

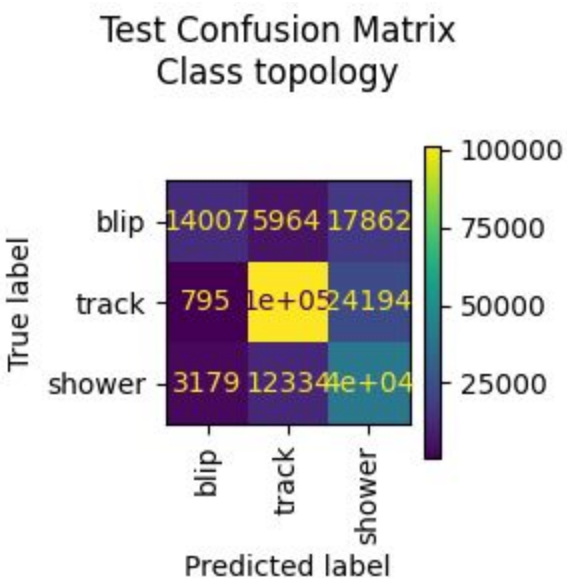
```
setfacl -m u:nobody:x /global/cfs/cdirs/dune/users/${USER}
shifter --image=docker:infophysics/blip:latest \
        --volume="${LOCAL_SCRATCH}:/local_scratch;${LOCAL_BLIP}:/local_blip;${LOCAL_DATA}:/local_data" \
        ./blip_segmentation_optimize.sh $hyper_parameter_config      You, 3 weeks ago • some changes
```

```
#!/bin/bash
cd /local_scratch

prep_data_config="/local_blip/BlipModels/blip_segmentation/2x2/configs/prep_data.yaml"
optimization_config="/local_blip/BlipModels/blip_segmentation/2x2/configs/optimize_blip_segmentation.yaml"

# download data to /local_data/ and process it
blip "$prep_data_config"

# generate hyper-parameter configs
create_hyper_parameter_configs "$optimization_config" -hyper_parameter_location=${LOCAL_BLIP}      You, 3 w
```

From 2x2 workshop (A. White)

BlipSegmentation [N. Carrara, G. Kuffaty]

Very early!



Losses and metrics can also be configured at run time.

```
criterion:
  CrossEntropyLoss:
    alpha:    1.0
    target_type:  "classes"
    targets:  ['particle', 'topology', 'physics_micro', 'physics_meso']
    outputs:  ['particle', 'topology', 'physics_micro', 'physics_meso']
    augmentations:  0

metrics:
  ConfusionMatrixMetric:
    target_type:  "classes"
    when_to_compute:  "test"
    targets:  ['particle', 'topology', 'physics_micro', 'physics_meso']
    outputs:  ['particle', 'topology', 'physics_micro', 'physics_meso']
    augmentations:  0
  AdjustedRandIndexMetric:
    target_type:  "classes"
    when_to_compute:  "test"
    targets:  ['particle', 'topology', 'physics_micro', 'physics_meso']
    outputs:  ['particle', 'topology', 'physics_micro', 'physics_meso']
    augmentations:  0
  AUROCMetric:
    target_type:  "classes"
    when_to_compute:  "test"
    targets:  ['particle', 'topology', 'physics_micro', 'physics_meso']
    outputs:  ['particle', 'topology', 'physics_micro', 'physics_meso']
    augmentations:  0
  DiceScoreMetric:
    target_type:  "classes"
    when_to_compute:  "test"
    targets:  ['particle', 'topology', 'physics_micro', 'physics_meso']
    outputs:  ['particle', 'topology', 'physics_micro', 'physics_meso']
    augmentations:  0
  JaccardIndexMetric:
    target_type:  "classes"
    when_to_compute:  "test"
    targets:  ['particle', 'topology', 'physics_micro', 'physics_meso']
    outputs:  ['particle', 'topology', 'physics_micro', 'physics_meso']
    augmentations:  0
  AveragePrecisionMetric:
    target_type:  "classes"
    when_to_compute:  "test"
    targets:  ['particle', 'topology', 'physics_micro', 'physics_meso']
    outputs:  ['particle', 'topology', 'physics_micro', 'physics_meso']
    augmentations:  0
  PrecisionMetric:
    target_type:  "classes"
    when_to_compute:  "test"
    targets:  ['particle', 'topology', 'physics_micro', 'physics_meso']
    outputs:  ['particle', 'topology', 'physics_micro', 'physics_meso']
    augmentations:  0
  RecallMetric:
    target_type:  "classes"
    when_to_compute:  "test"
    targets:  ['particle', 'topology', 'physics_micro', 'physics_meso']
    outputs:  ['particle', 'topology', 'physics_micro', 'physics_meso']
    augmentations:  0
```

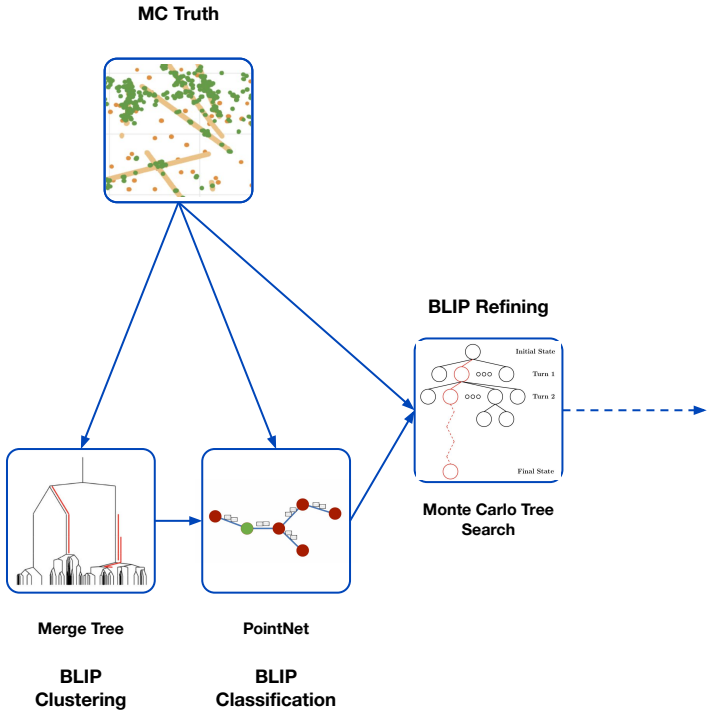
From 2x2 workshop (A. White)

BlipNet

N. Carrara, D. Rivera, L. Perez-Molina

Let's compare our results with MC Truth at each stage of the reconstruction. The full chain begins with,

1. **Merge Tree** - First we construct a topological description of the event called a *decorated merge tree*.
2. **BLIP Classification** - BlipGraph is used to classify blips as different types (Ar39, Kr85, Rn222, Capture Gammas, etc.)
3. **BLIP Refining** - A Monte Carlo Tree Search algorithm is used to refine blips and construct higher-level structures.



UCDAVIS

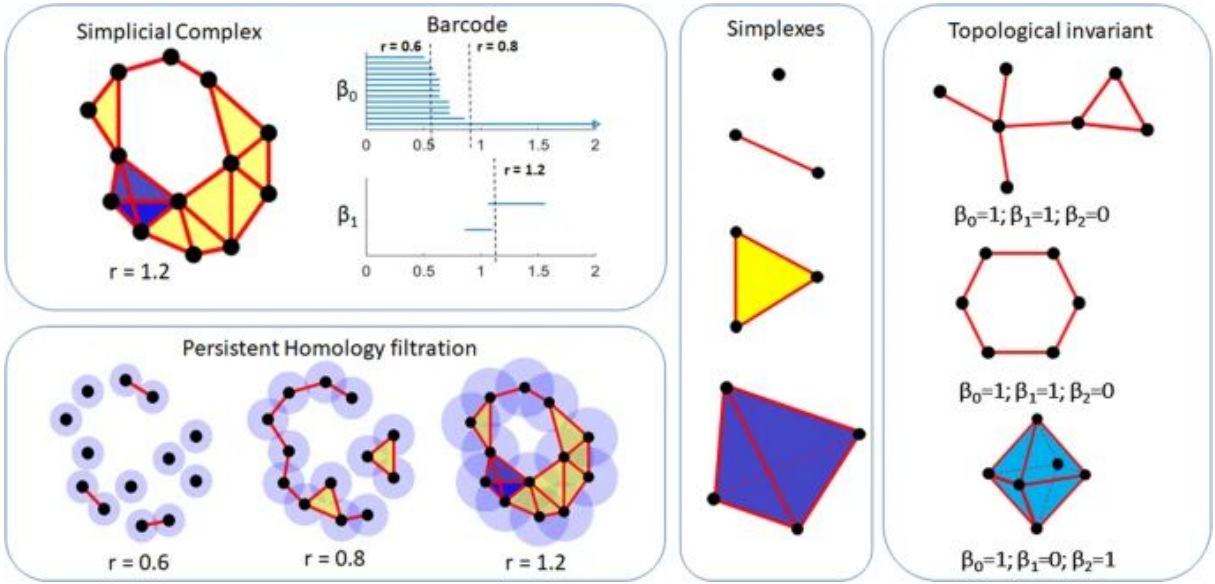
BlipNet

N. Carrara, D. Rivera, L. Perez-Molina

Let's compare our results with MC Truth at each stage of the reconstruction. The full chain begins with,

1. **Merge Tree** - First we construct a topological description of the event called a *decorated merge tree*.

Persistent Homology - A set of theorems and techniques in ***topological data analysis*** for studying topological features of point sets.



5 G. Carlsson, Topology and Data, (https://www.ams.org/journals/bull/2009-46-02/S0273-0979-09-01249-X/S0273-0979-09-01249-X.pdf).

6 H. Edelsbrunner and J. Harer, Persistent Homology - A Survey, (https://www.maths.ed.ac.uk/~v1ranick/papers/edelhare.pdf).

7 H. Edelsbrunner et al., Topological Data Analysis in Information Space, (https://arxiv.org/abs/1903.08510).

BlipNet

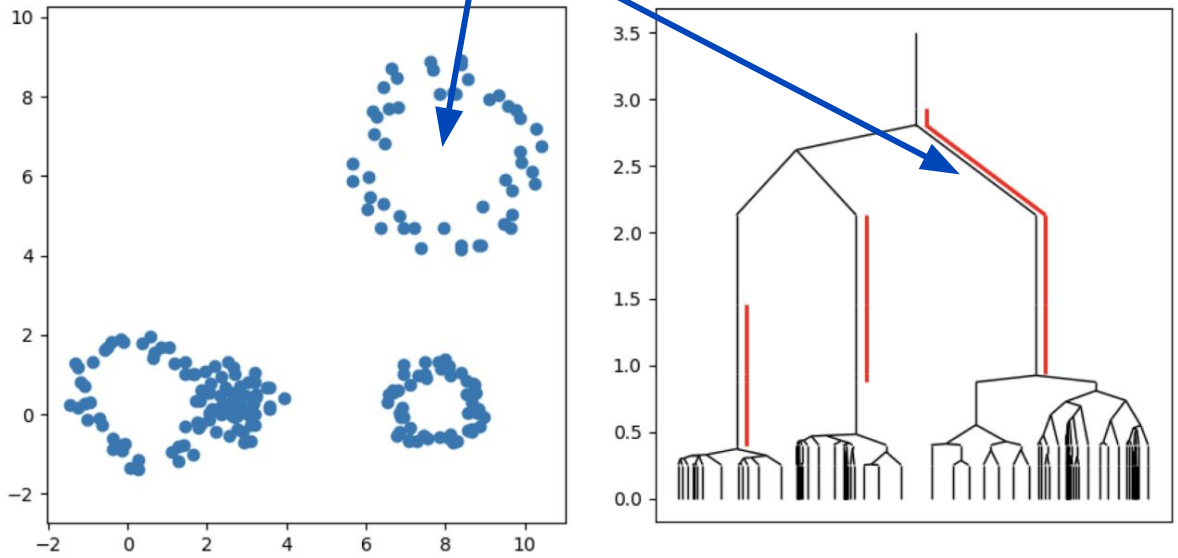
N. Carrara, D. Rivera, L. Perez-Molina

Let's compare our results with MC Truth at each stage of the reconstruction. The full chain begins with,

1. **Merge Tree** - First we construct a topological description of the event called a *decorated merge tree*.

Decorated Merge Tree - Topological description of a point set where a *merge tree* is decorated with degree (n > 0) features.

Degree 1 features



8 J. Curry et al., Decorated Merge Trees for Persistent Homology, (https://arxiv.org/abs/2103.15804).

BlipNet

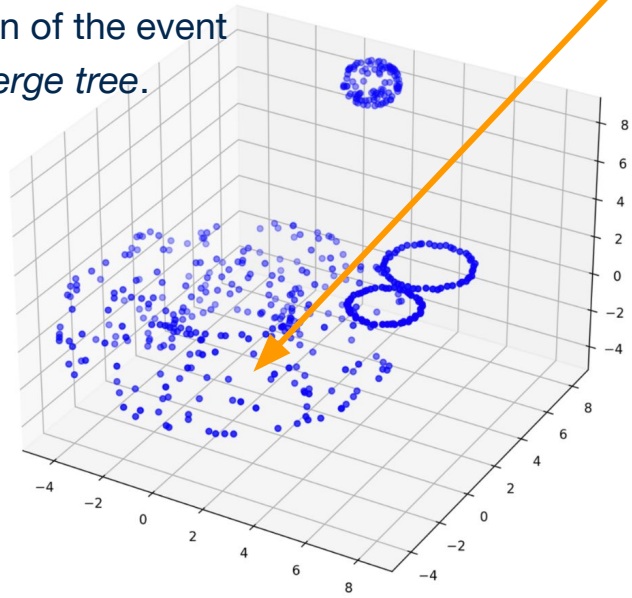
N. Carrara, D. Rivera, L. Perez-Molina

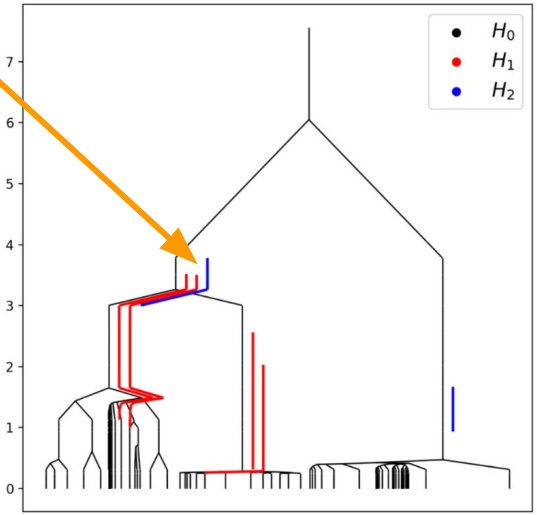
Let's compare our results with MC Truth at each stage of the reconstruction. The full chain begins with,

1. **Merge Tree** - First we construct a topological description of the event called a *decorated merge tree*.

Decorated Merge Tree - Topological description of a point set where a *merge tree* is decorated with degree (n > 0) features.

Degree 2 features





BlipNet

N. Carrara, D. Rivera, L. Perez-Molina

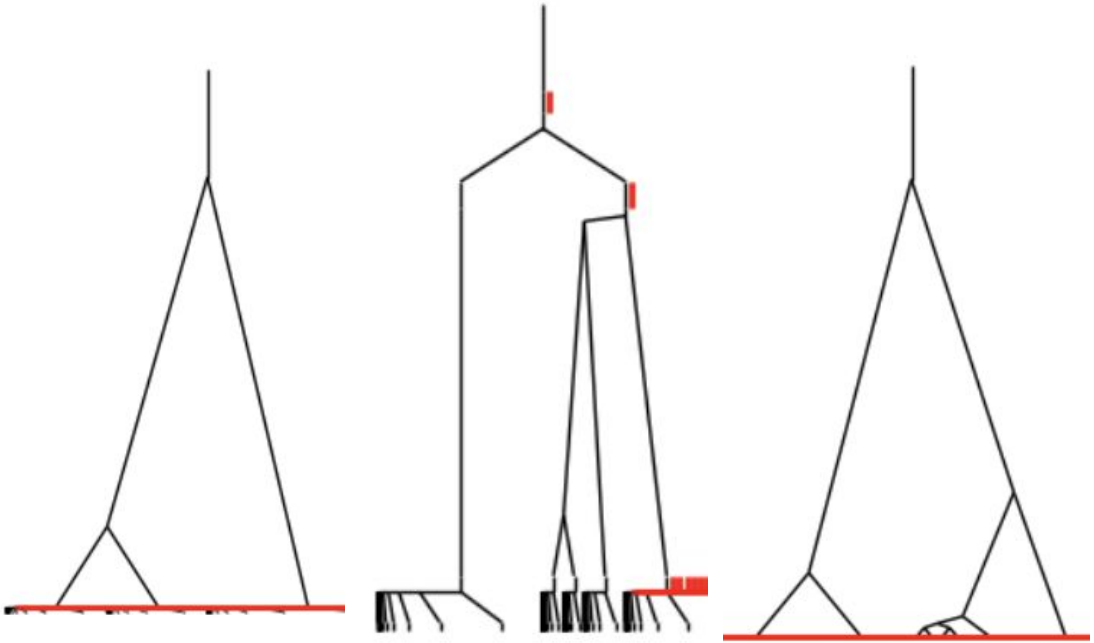
Let's compare our results with MC Truth at each stage of the reconstruction. The full chain begins with,

1. **Merge Tree** - First we construct a topological description of the event called a *decorated merge tree*.

Merge Tree code developed by Laura Perez-Molina.

Decorated Merge Tree - Topological description of a point set where a *merge tree* is decorated with degree ($n > 0$) features.

Merge Trees from capture gammas

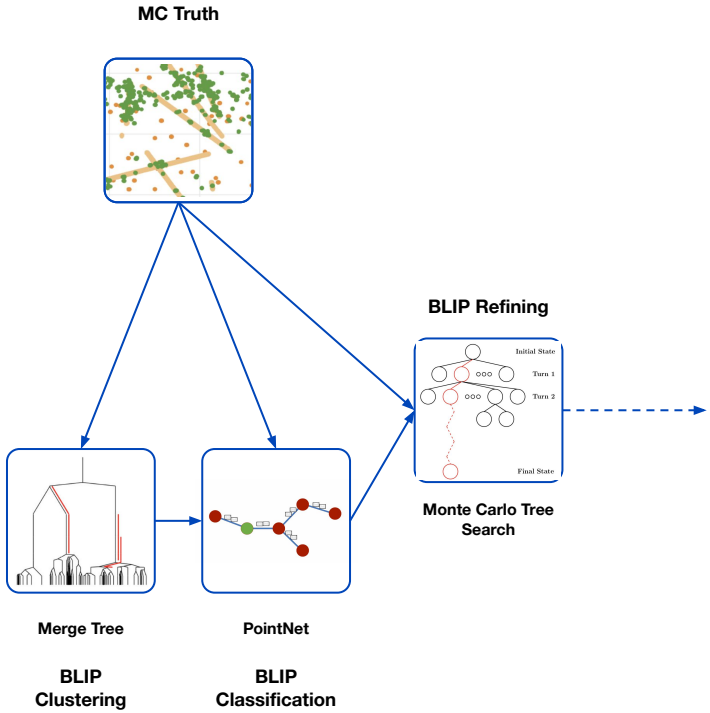


BlipNet

N. Carrara, D. Rivera, L. Perez-Molina

Let's compare our results with MC Truth at each stage of the reconstruction. The full chain begins with,

1. **Merge Tree** - First we construct a topological description of the event called a *decorated merge tree*.
2. **BLIP Classification** - PointNet is used to classify blips as different types (Ar39, Kr85, Rn222, Capture Gammas, etc.)
3. **BLIP Refining** - A Monte Carlo Tree Search algorithm is used to refine blips and construct higher-level structures.

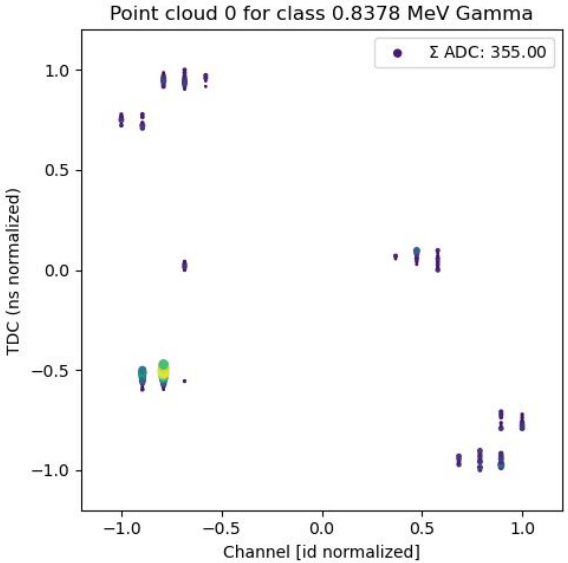


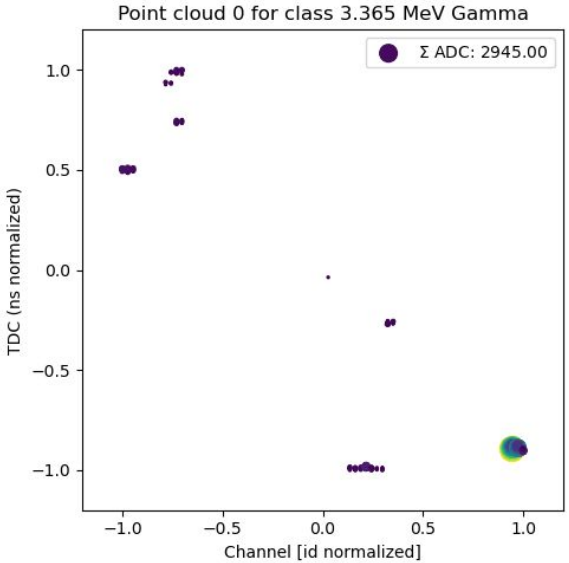
UCDAVIS

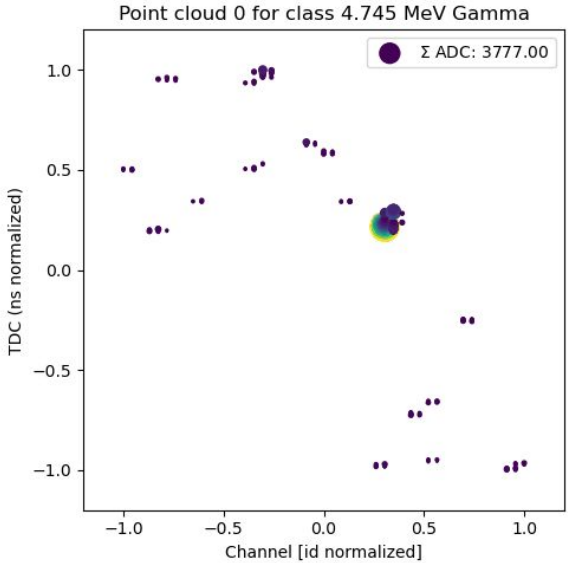
Blip Examples

Below are examples of different neutron capture gammas simulated in LArSoft.

- Each cluster is normalized so that it is centered at zero and extends from [-1,1] in each variable.
- ADC is normalized over all events.



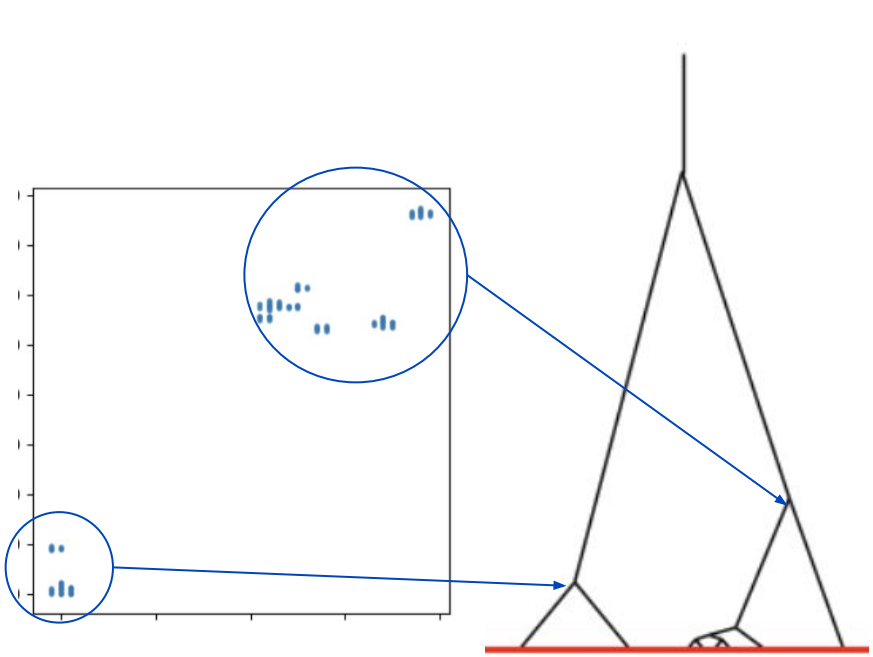


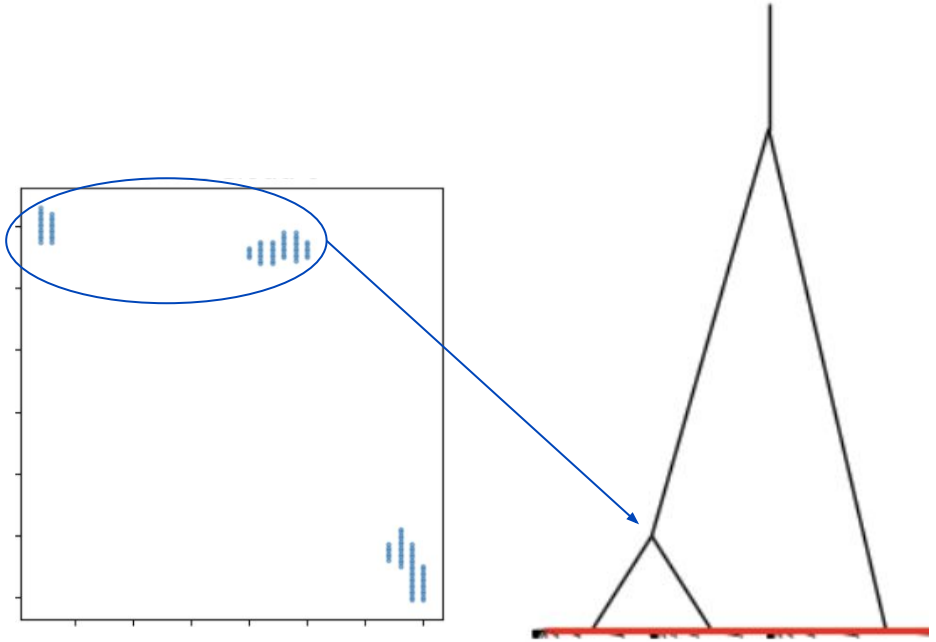


Blip Examples

Below are examples of different neutron capture gammas simulated in LArSoft.

- Merge Trees for 2.5661 MeV Gammas.



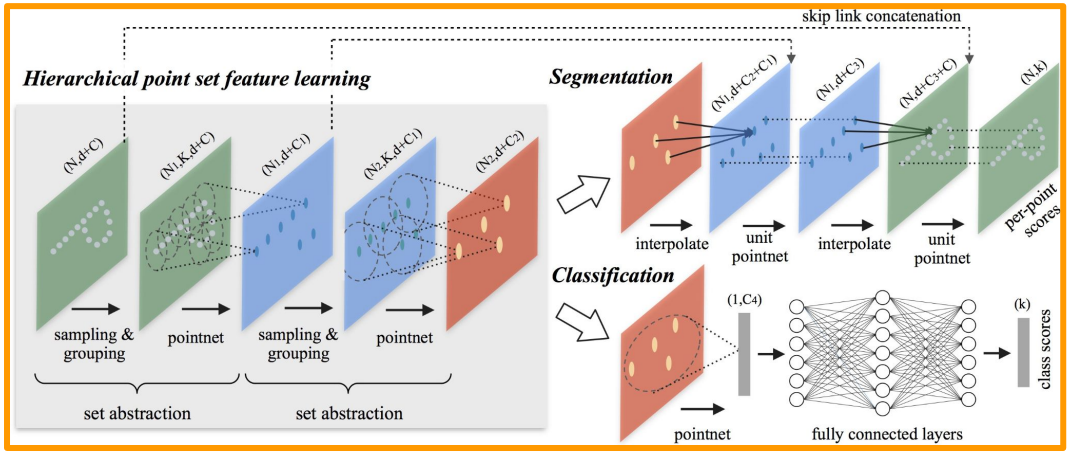


BLIP Classification

BLIP Classification will consist of at least the following two models:

- **PointNet++** - The first network will take in an entire detector readout for an event and learn to semantically label blips (e.g. PointNet++[5]).

- **BlipGraph** - The second network will learn to classify clusters at different scales (e.g. PointNet[4], DynamicEdgeConv[6]).



Models are built using the ***PyTorch Geometric*** API (https://pytorch-geometric.readthedocs.io/en/latest/)

4 *PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation,* C. Qi et. al., CVPR 2017, (https://arxiv.org/abs/1612.00593).
5 *PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space*, C. Qi et. al., (https://arxiv.org/abs/1706.02413)
6 *Dynamic Graph CNN for Learning on Point Clouds,* Y. Wang et. al., 2018, (https://arxiv.org/abs/1801.07829)

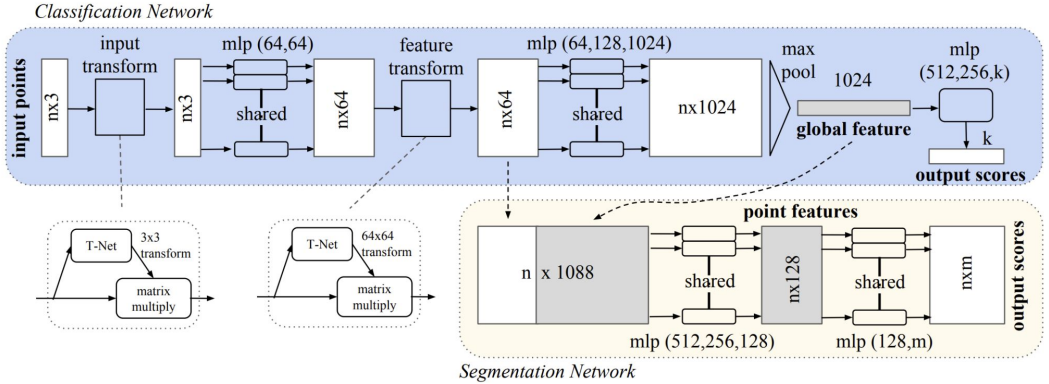
UCDAVIS

PointNet/DynamicEdgeConv

Figure from C. Qi et al. (2017).

Point cloud neural networks take advantage of three main properties of the datasets:

1. **Unordered** - Point clouds, unlike pixel arrays, are unordered and can have a variable number of points!
2. **Distance Metric** - Our point clouds have a geometry and local interaction information.
3. **Symmetries** - Due to physics/geometry, the point clouds are invariant under certain transformations.



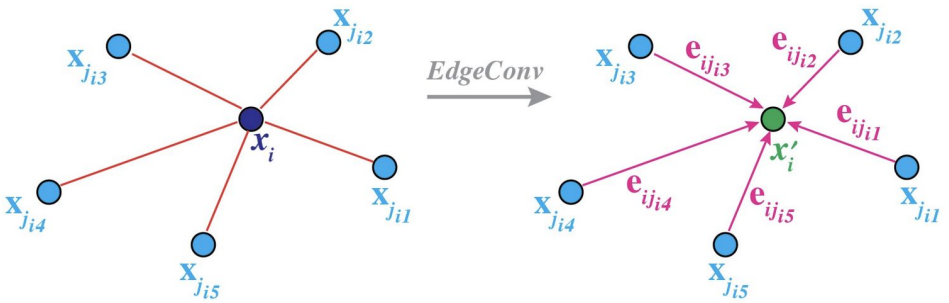


Figure from Y. Wang et al. (2019).

UCDAVIS

Contrastive Learning

We can utilize symmetries in our dataset to conduct a semi-supervised learning in which point clouds are grouped together by symmetries in their representation.

Using the *Normalized Temperature-scaled Cross Entropy Loss* (***NTXent***):

$$\ell_{i,j} = -\log \frac{\exp(\text{sim}(\boldsymbol{z}_i, \boldsymbol{z}_j)/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp(\text{sim}(\boldsymbol{z}_i, \boldsymbol{z}_k)/\tau)}, \qquad (1)$$

we push “like” clusters closer together in the embedding space and push all others away.

? K. Sohn, *Improved Deep Metric Learning With Multi-class N-pair Loss Objective*, NIPS (2016), (https://proceedings.neurips.cc/paper_files/paper/2016/file/6b180037abbebea991d8b1232f8a8ca9-Paper.pdf)

Contrastive learning scheme (https://ai.googleblog.com/2020/04/advancing-self-supervised-and-semi.html)

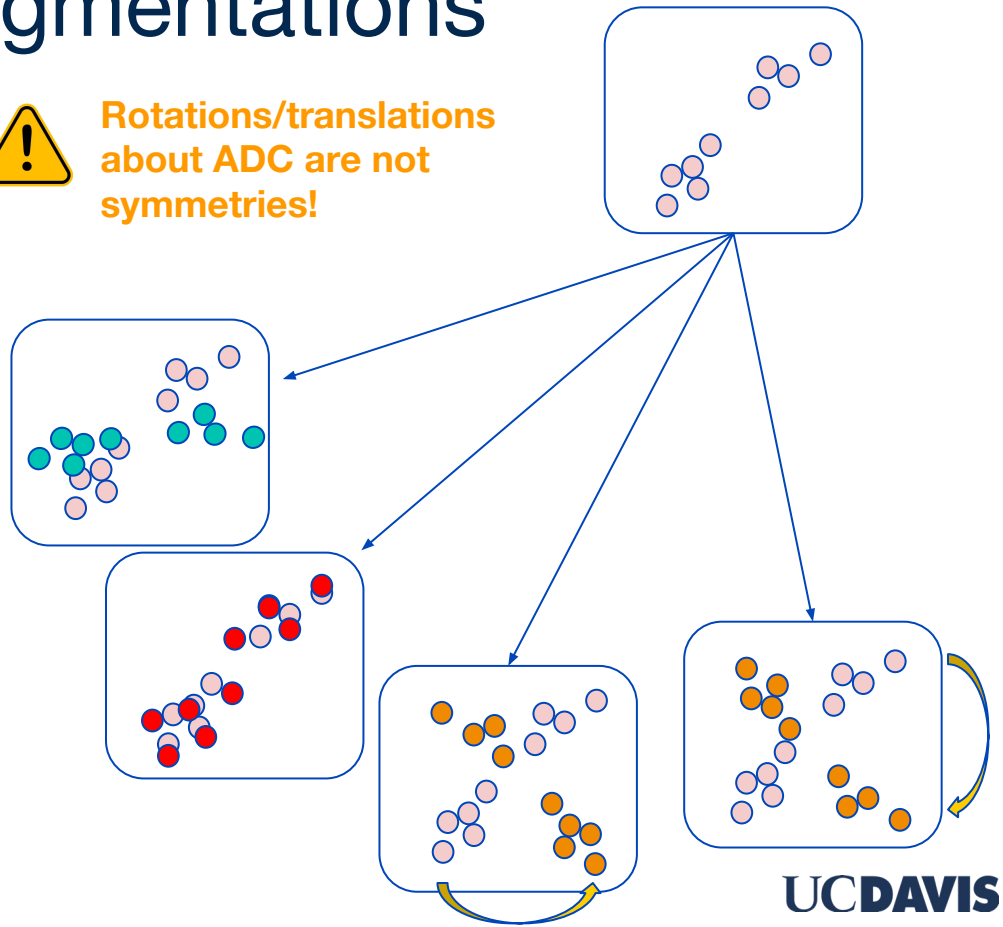
UCDAVIS

Symmetries -> Augmentations

There are several symmetries in our dataset due to physics:

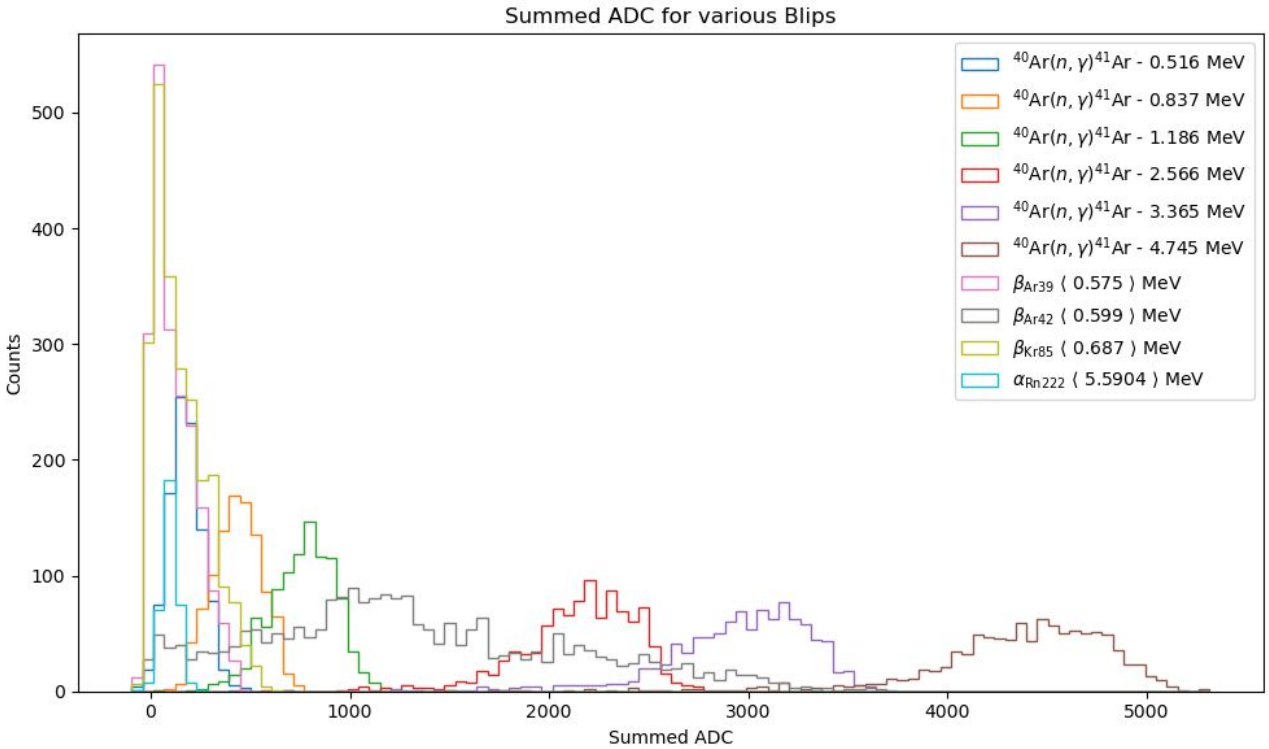
1. **Translation** - Translations in (channel,tdc) are handled by normalizing each point cloud to the origin.
2. **Rotation** - Rotations about (channel,tdc) are implemented as part of the augmentations.
3. **Fluctuations** - Fluctuations can occur from many sources including the electron drift.
4. **Parity** - Flipping of the momentum of the particle along the (channel, tdc) directions.

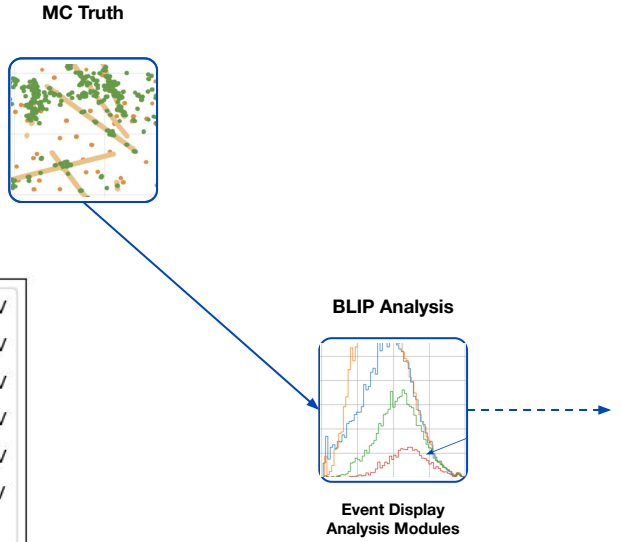
Rotations/translations about ADC are not symmetries!



UCDAVIS

Far Detector (channel, tdc, adc)

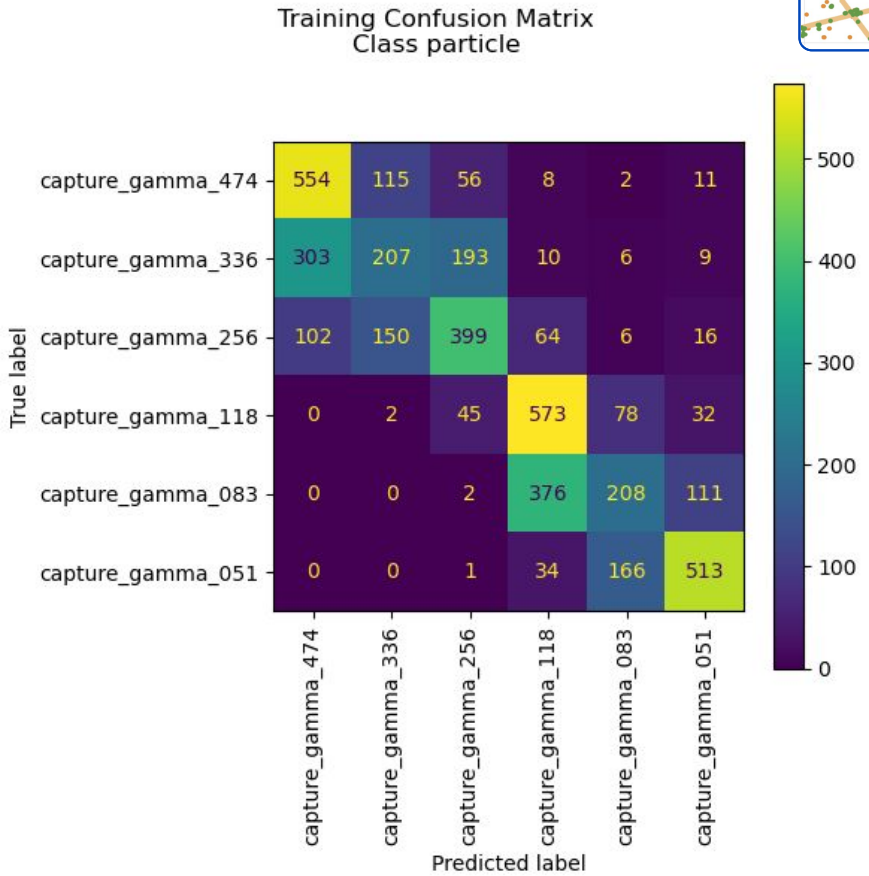


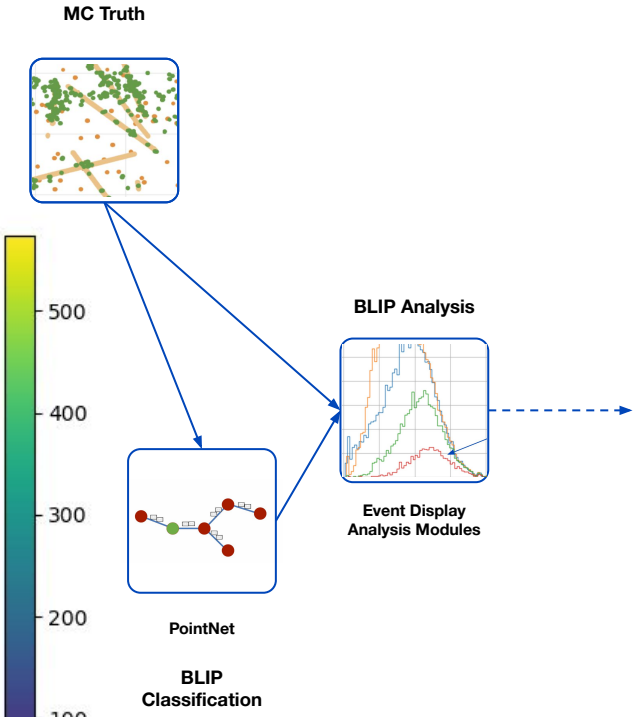


UCDAVIS

MCTruth + BlipGraph

The model needs some tuning, but the current version seems to be learning something about the shapes of the point clouds.

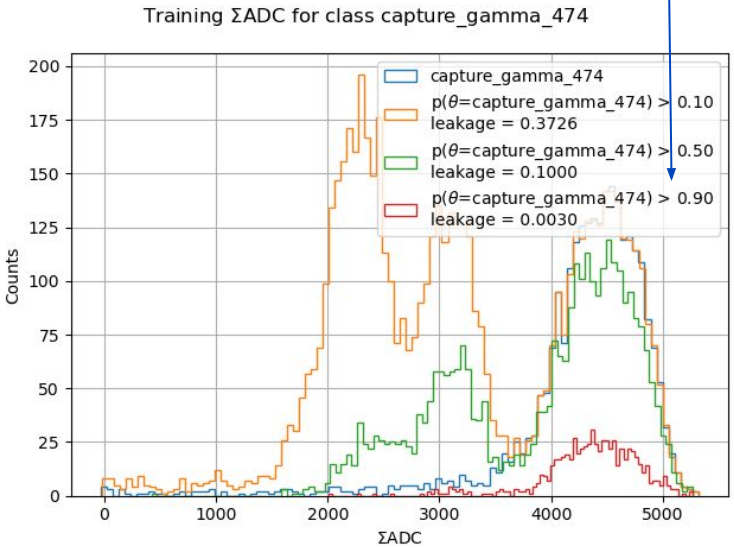


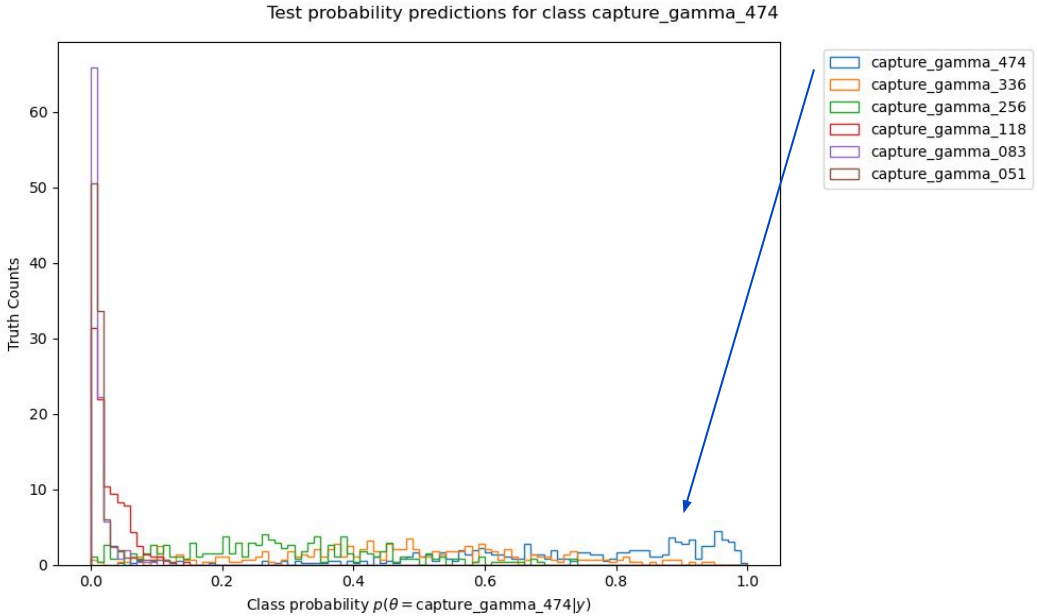


UCDAVIS

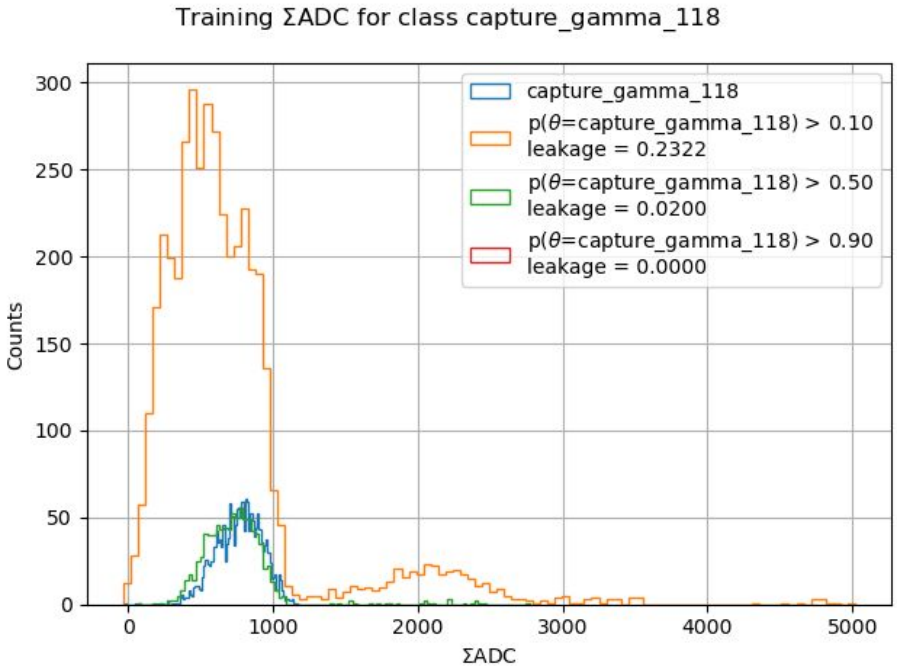
MCTruth + BlipGraph

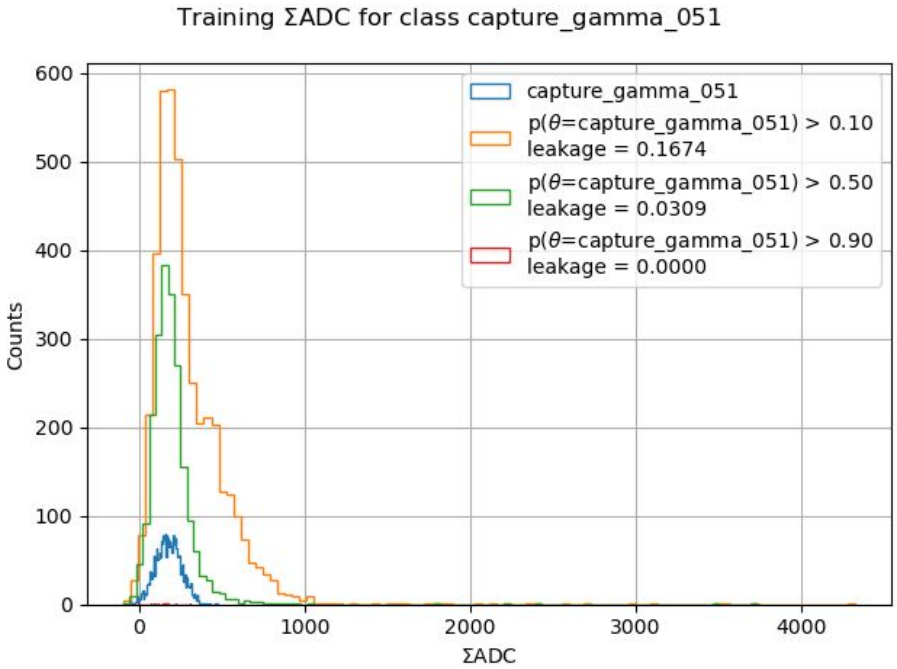
This early model rejects 99.7 % of all other blip types when imposing a 90% cut on the probability to being a 4.745 MeV gamma.





MCTruth + BlipGraph



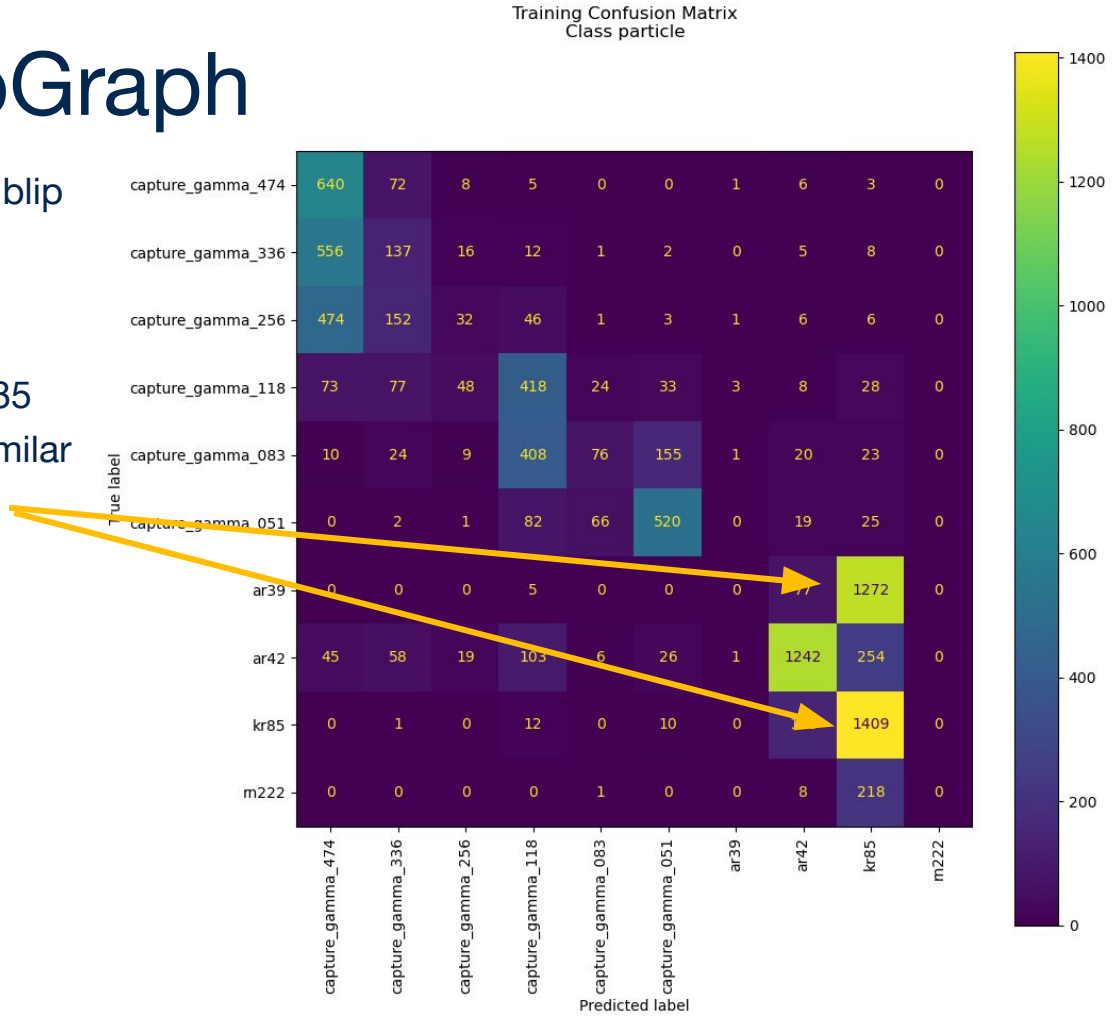


UCDAVIS

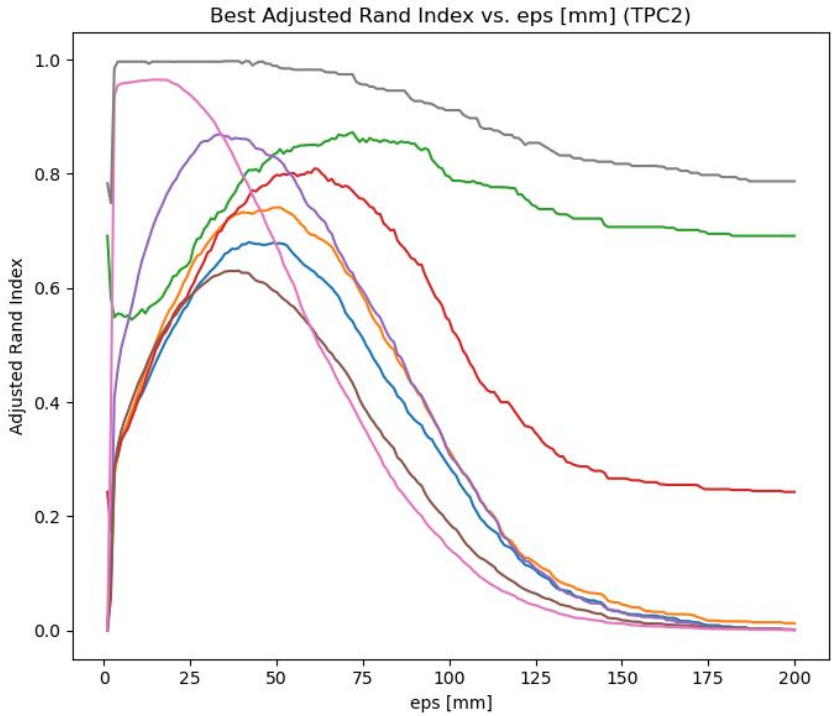
MCTruth + BlipGraph

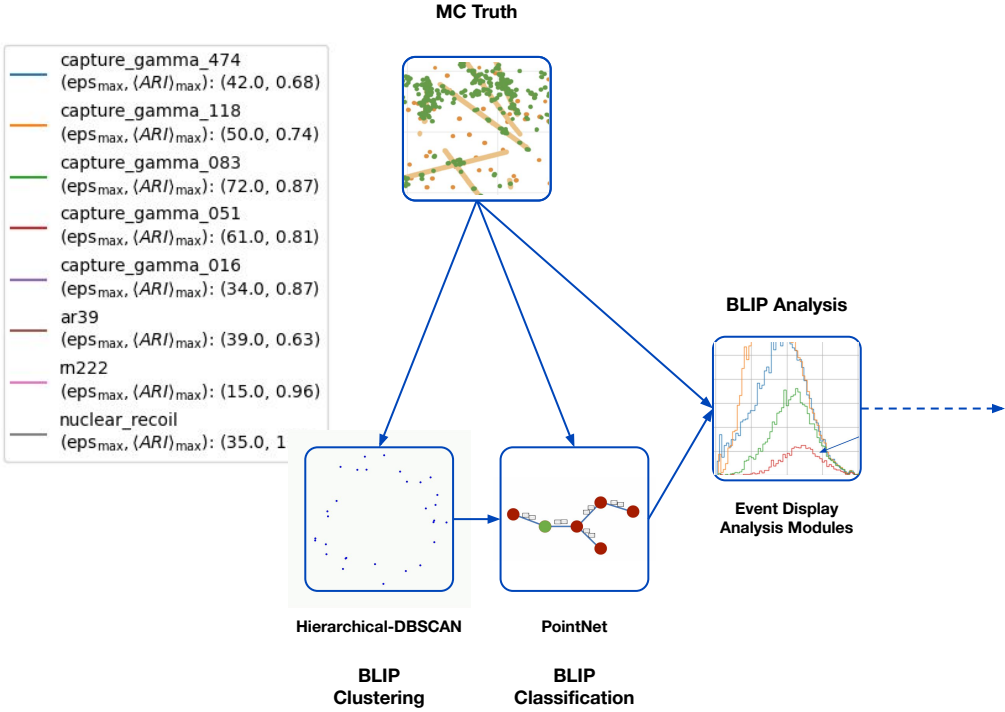
Some preliminary results on all current blip types.

Perhaps unsurprising that Ar39 and Kr85 are difficult to distinguish given their similar physics.



MCTruth + Clustering + BlipGraph





UCDAVIS

BlipNet

Let's compare our results with MC Truth at each stage of the reconstruction. The full chain begins with,

1. **Merge Tree** - First we construct a topological description of the event called a *decorated merge tree*.
2. **BLIP Classification** - PointNet is used to classify blips as different types (Ar39, Kr85, Rn222, Capture Gammas, etc.)
3. **BLIP Refining** - A Monte Carlo Tree Search algorithm is used to refine blips and construct higher-level structures.

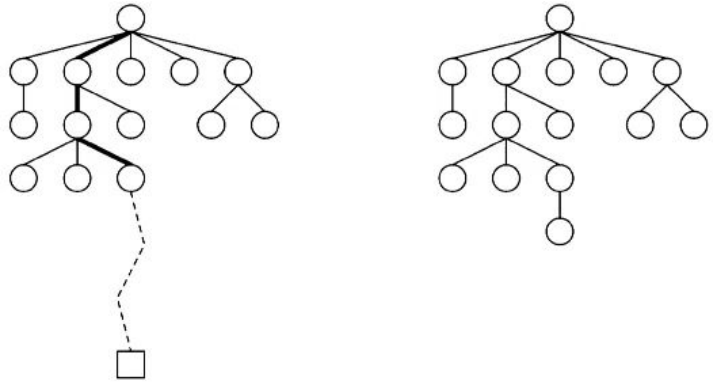
N. Carrara, D. Rivera, L. Perez-Molina

Stay tuned for progress on BlipNet! See this recent publication (https://arxiv.org/abs/2309.08516) for some details on Monte Carlo Tree Search:

USING MONTE CARLO TREE SEARCH TO CALCULATE MUTUAL INFORMATION IN HIGH DIMENSIONS

Nick Carrara
Physics Department
University of California, Davis
Davis, CA 95616
nmcarrara@ucdavis.edu

Jesse Ernst
Physics Department
University at Albany, SUNY
Albany, NY 12222
jae@albany.edu



Work in Progress

Arrakis:

- ✓ module for gathering simulation/data products.
  - ✓ Wire plane/Raw digit data.
  - ❑ Reconstructed 3D space point data.
- ✓ module for labeling logic.
- ❑ integrate into ‘lar’ repositories.
- ❑ methods for ‘registering’ custom labeling logic.

Wilson Cluster/NERSC:

- ❑ Implement BLIP as a module that users can easily access without having to install (e.g. ‘module load blip’).
- ❑ Jupyter-lab/event-display over ssh integration.

BLIP:

- ✓ Construct infrastructure for generating training data/implementing models and training.
- ✓ Construct Semantic Segmentation model.
  - ❑ Optimize semantic segmentation.
- ✓ Construct Simple PointNet model.
  - ❑ Optimize BLIP Classification models (PointNet++ and HDBSCAN-PointNet).
- ❑ Continue adding new models/modules to extend the scope of the neural network applications.
- ✓ Implement a system to allow users to integrate their own model/analysis code.

UCDAVIS

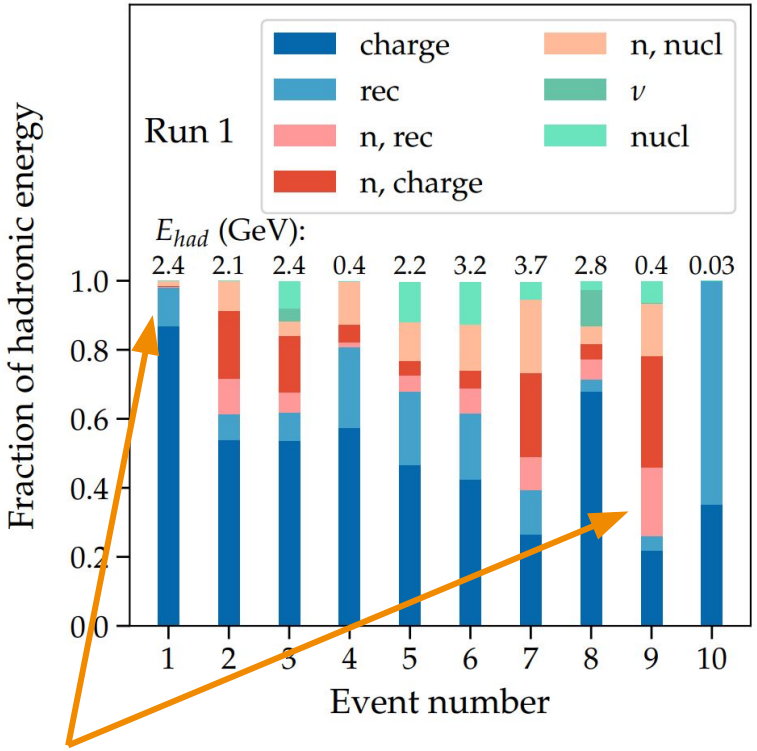
Backup

Energy resolution?

It is necessary to be able to account for the neutron "***missing energy***" in order to make precision oscillation measurements (production and transport).

A. Friedland and W. Li (2019) characterized the sources of missing energy in LArTPCs as,

1. Electronic shower sprays (**charge**)
2. Recombination effects (**rec**)
3. Nuclear breakup (**nucl**)
4. Outgoing neutrinos (***ν***)
5. Subthreshold particles
6. **Primary and secondary neutrons**



Contributions from neutrons vary widely from event to event!

Figure from A. Friedland and W. Li (2019).

UCDAVIS

Energy resolution?

This pie chart from the same paper (2019) shows the average amount of missing energy over many events (~10K).

- Neutrons contribute nearly **~30%** of the missing energy on average!

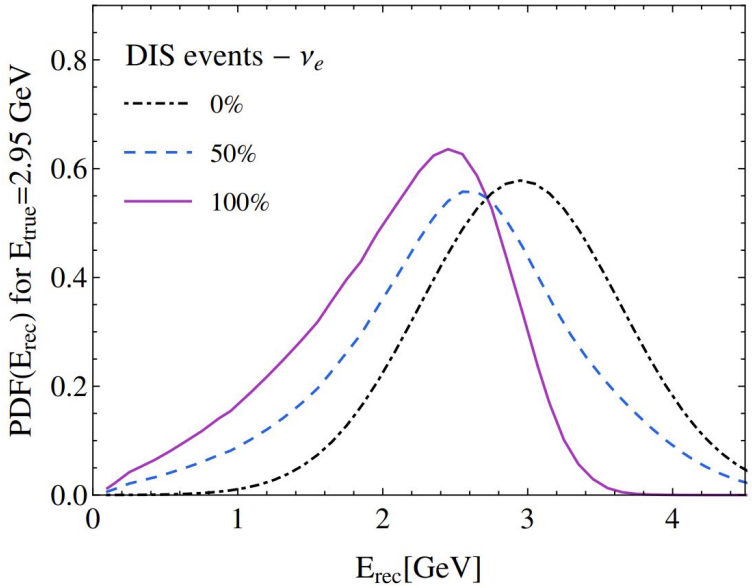


Figure from A. Ankowski et al. (2015).

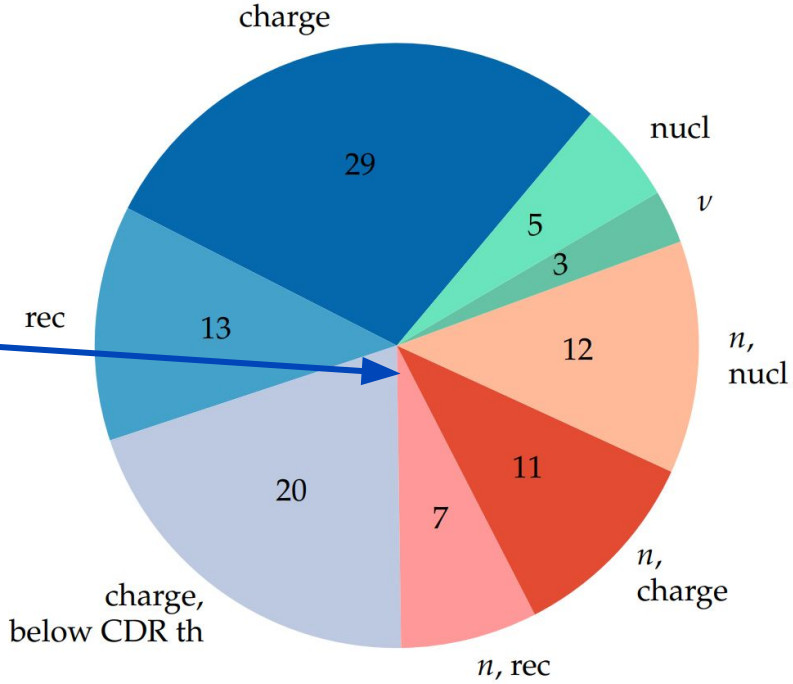
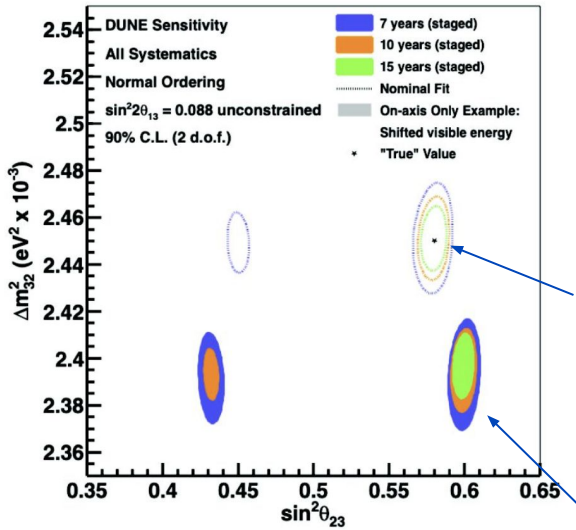


Figure from A. Friedland and W. Li (2019).

UCDAVIS

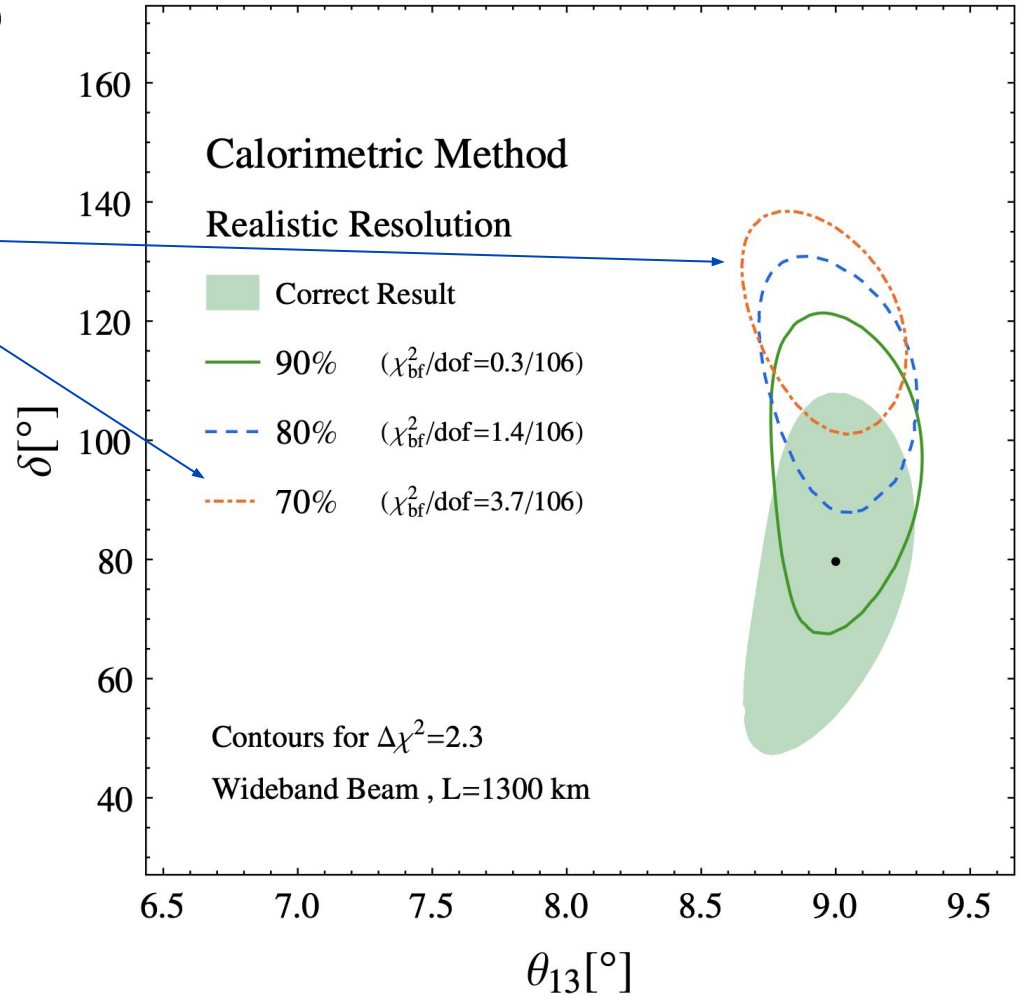
Energy resolution?

- Measurements of the CP phase[3] are shifted by a large amount if neutrons are ignored!



fit with 20% neutron energy missing

actual

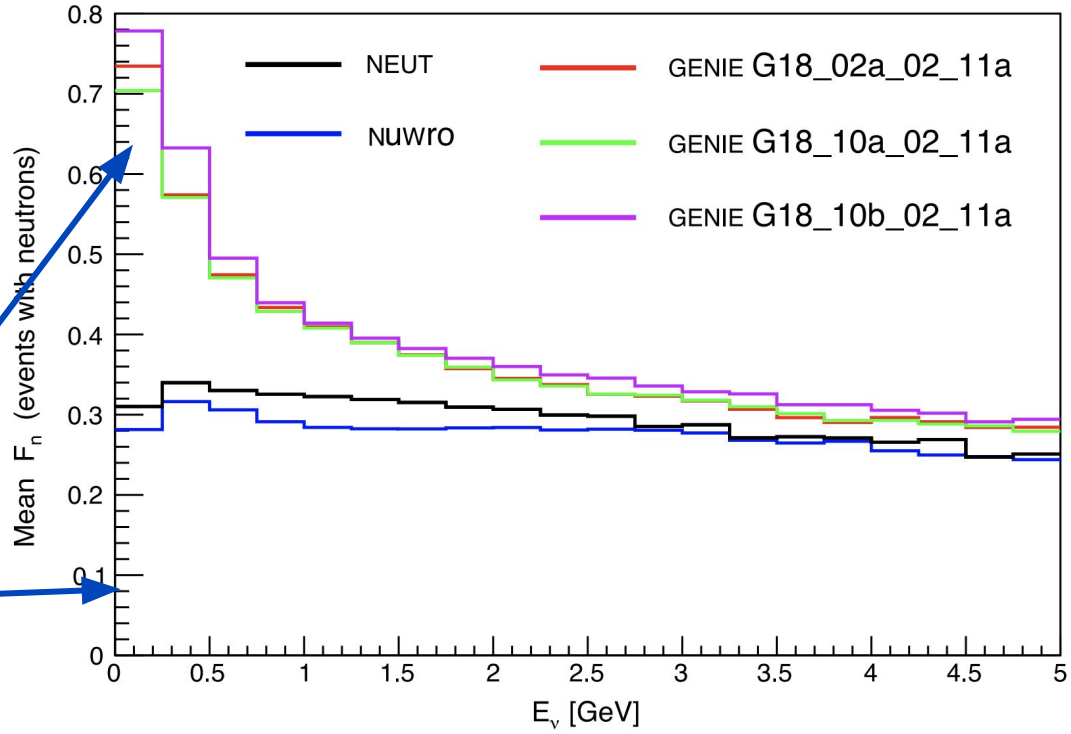


3 A. M. Ankowski et al., *Missing energy and the measurement of the cp-violating phase in neutrino oscillations*, Phys. Rev. D, **92** 2015 (https://arxiv.org/abs/1507.08561)

Model dependent final-states?

M. Buizza Avanzini et al. examined the differences in neutron final-states from the generators **NEUT**, **NuWro**, and **GENIE**.

- The fraction of total energy given to hadrons (when at least one hadron is a neutron) is shown as a function of incident neutrino energy in a CC-interaction.
- GENIE varies in the lower energy regime by a factor of 2!



4 M. Buizza Avanzini, *Comparisons and challenges of modern neutrino-scattering experiments*, Phys. Rev D, **105** 2022 (https://journals.aps.org/prd/pdf/10.1103/PhysRevD.105.092004)

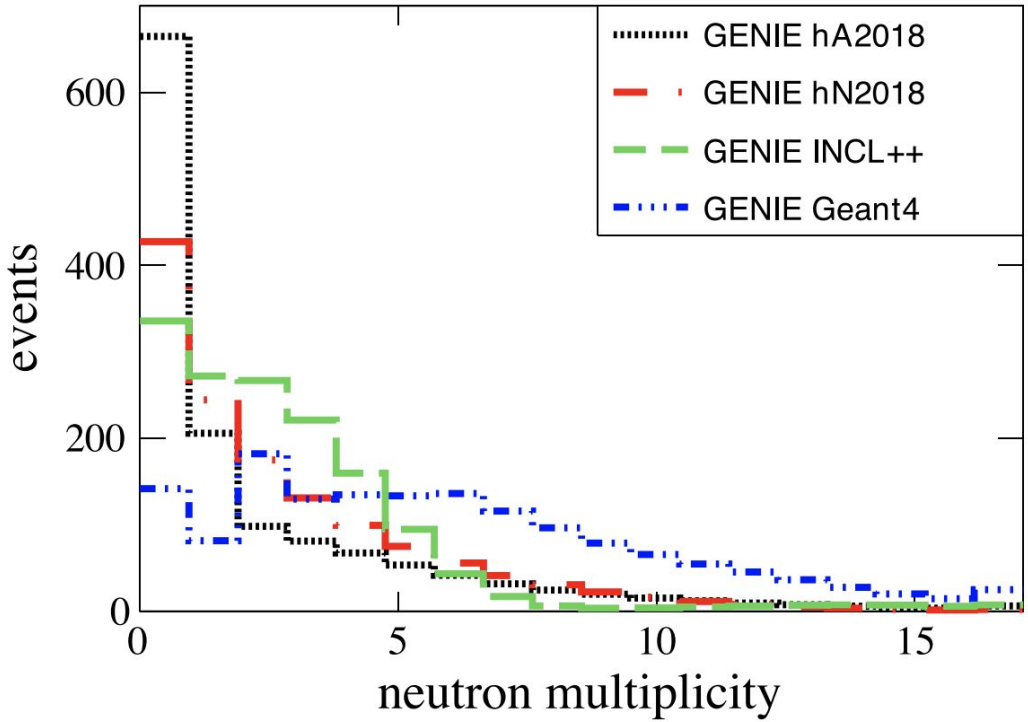
UCDAVIS

Model dependent final-states?

The various models within GENIE[5] also show a discrepancy in the predictions of neutron multiplicity.

- Neutron distributions from a simulated 2GeV muon neutrino interaction on Ar40.

Our group is working on a potential experiment to be deployed in ANNIE to measure neutron multiplicity on Ar!



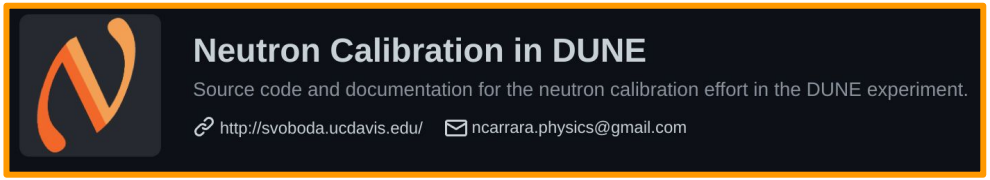
5 The GENIE Collaboration, *Recent highlights from GENIE v3*, Eur. Phys. J. Spec. Top. (2021) (https://link.springer.com/article/10.1140/epjs/s11734-021-00295-7)

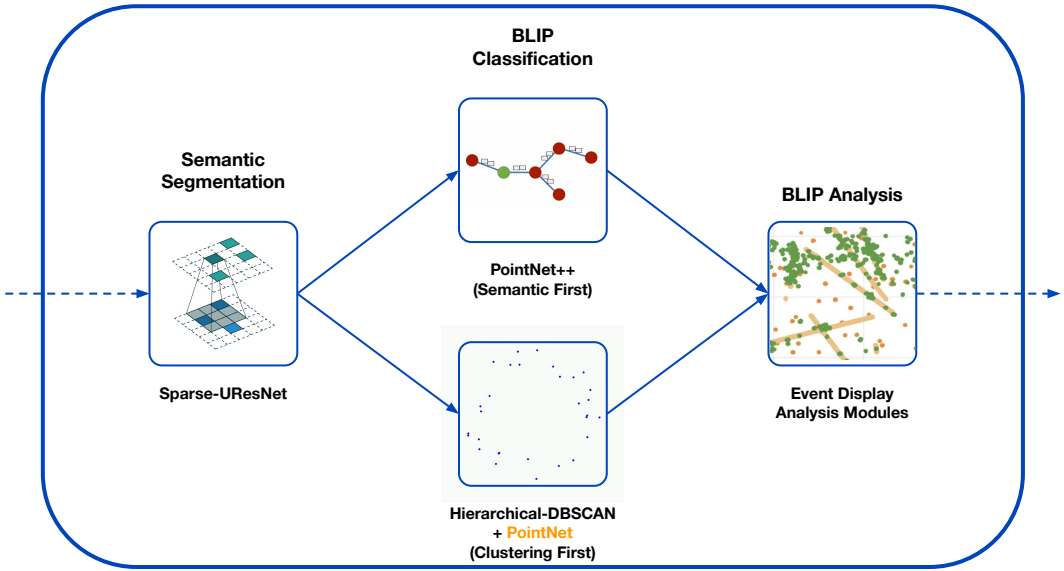
UCDAVIS

BLIP

We introduce ***BLIP***[2], a collection of ML algorithms for classifying low energy interactions in LArTPCs.

- **Semantic Segmentation**:
  - Pixel/point-cloud level classification for detector readout and/or reconstructed space points.
  - Identification of tracks/showers in order to **isolate** Blips.
- **Heirarchical Clustering**:
  - DBSCAN and Heirarchical-DBSCAN for clustering BLIPs.
- **Point-cloud Classification**:
  - BlipNet models for classifying BLIP point clouds.
- **Topological Data Analysis (TDA)**:
  - State-of-the-art algorithms for characterizing the topology of point sets (e.g. persistent homology).



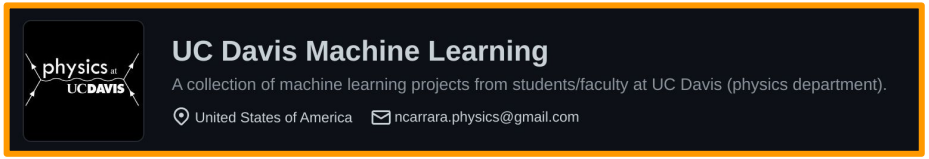


2 *BLIP, BLIP ML* Group (UC Davis), 2023, (https://github.com/Neutron-Calibration-in-DUNE/Blip).

BLIP

UCDAVIS

Arrakis LArSoft Module



The ***Arrakis***[3] LArSoft module is responsible for collecting MC truth/detector output information and generating point clouds for training.

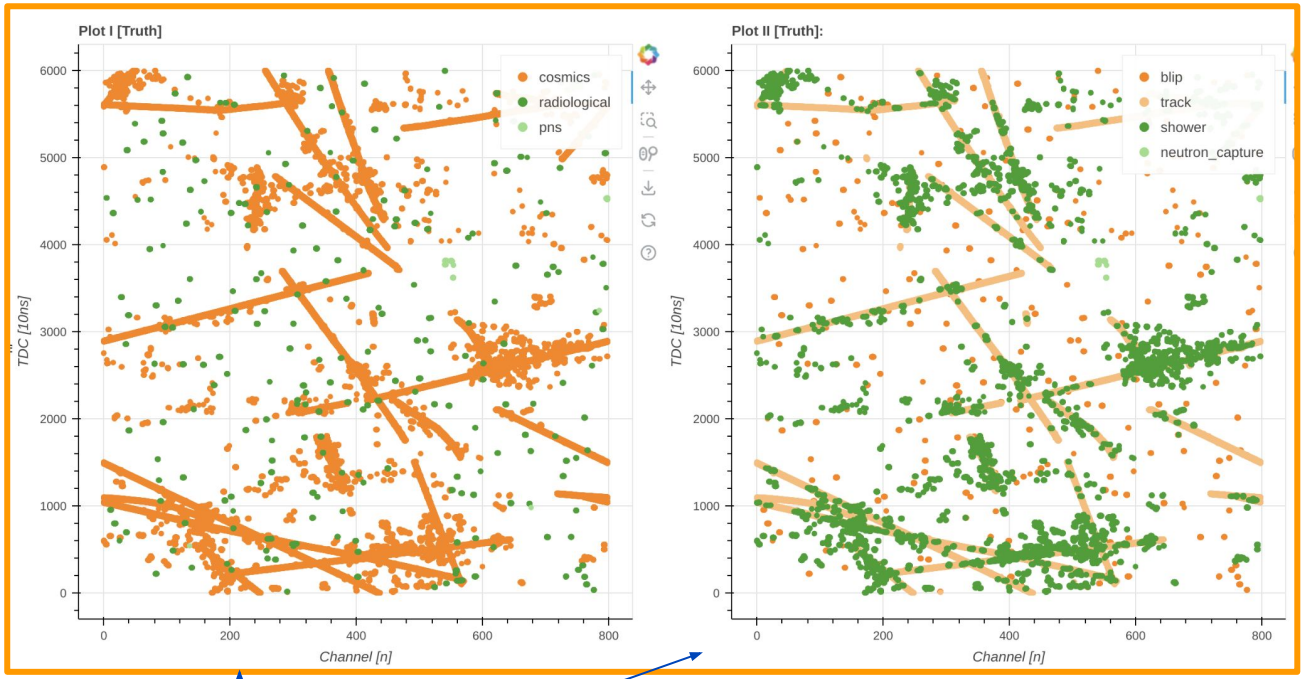
Current Labeling Schemes:

[Segmenation]

- **Source -** specifies the generator.
- **Shape -** generic topological descriptor (track, shower, blip, …)
- **Particle -** particle type responsible for the energy deposition (mu+-, neutron, proton, michel, delta, …)

[Clustering]

- **UniqueShape -** individual instance of a particular shape.
- **UniqueParticle -** individual instance of a particular particle.



‘source’ and ‘shape’ labels for a simulated ProtoDUNE event

UCDAVIS

3 *Arrakis (LArSoft module),* DUNE PNS Group (UC Davis), 2023, (https://github.com/UC-Davis-Machine-Learning/Arrakis).

Arrakis LArSoft Module

To use Arrakis,

[Current setup]

- Can use the LArSoftArrakis repository to set up a local LArSoft install on an FNAL server.
- Download Arrakis into larana and compile.
- Add “arrakis” as an analyzer module and specify parameters.

```
#include "Arrakis.fcl"

physics:
{
    analyzers: {ana: @local::Arrakis}
    analysis:  [ana]
    trigger_paths:  [simulate]
    end_paths:      [analysis]

}
```

[Future setup]

- Eventually, Arrakis will be integrated into LArSoft and will be available as an analyzer module.
- Allow the user to register custom *logic* for generating training datasets.

```
module_type:    "Arrakis"

# which products for the wrangler to handle
ProcessType:                "data"  # simulation, data
ProcessMCTruth:             true
ProcessMCParticles:         true
ProcessSimEnergyDeposits:   true
ProcessSimChannels:         true
ProcessRawDigits:           true

# module labels
LArGeantProducerLabel:      "largeant"
SimEnergyDepositProducerLabel:   "IonAndScint"
SimEnergyDepositInstanceLabel:   "priorSCE"
SimChannelProducerLabel:    "tpcrawdecoder"
SimChannelInstanceLabel:    "simpleSC"
RawDigitProducerLabel:      "tpcrawdecoder"
RawDigitInstanceLabel:      "daq"

GeneratorLabels:
{
    Ar39Label:          "Ar39"
    Ar42Label:          "Ar42"
    Kr85Label:          "Kr85"
    Rn222Label:         "Rn222"
    BeamLabel:          "Beam"
    CosmicsLabel:       "Cosmics"
    HEPevtLabel:        "HEPevt"
    PNSLabel:           "PNS"
}

# which products to save
SaveMeta:       true
SaveGeometry:   true
SaveSimulationWrangler:         false   # save SimulationWrangler maps
SaveWirePlanePointCloud:        true    # save wire plane point cloud data
SaveEnergyDepositPointCloud:    true    # save energy deposit point cloud data

# whether to collect detector simulation by edep or
# by particle track id.
FilterDetectorSimulation:   "TrackID"   # ["TrackID", "EdepID"]

# labeling scheme for neutron captures,
# simple labels all gammas as "other", while medium labels
# 4.75 and 1.81 MeV gammas separately from others, and full
# labels all energies as different types.
NeutronCaptureGammaDetail:  "Simple"    # ["Simple", "Medium", "Full"]

ADCThreshold:                   20      # ADC threshold for saving points
InducedChannelInfluence:        10      # number of wires which are influenced by signals
InducedTDCInfluence:            200     # number of tdc ticks which are influenced by signals
```

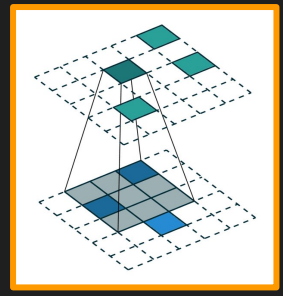
FHiCL parameters

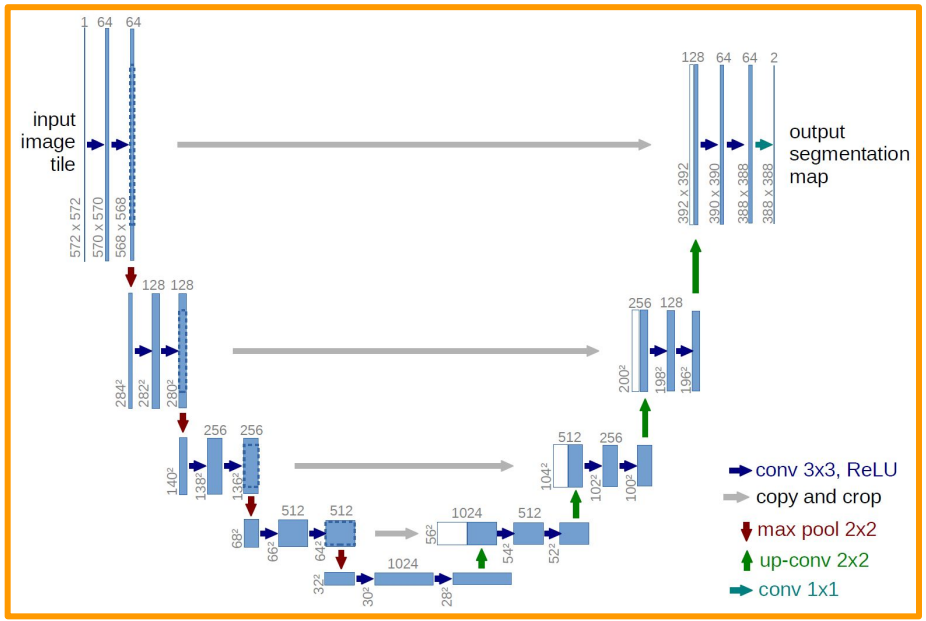
Config file training/evaluation

BLIP models can be constructed at run time with tunable parameters.

- The long term goal is to make a completely modular setup so that different models can be chained together.
- Model parameters/weights are saved in a config dictionary for reproducability.

```
model:
  # uncomment the line below and specify the model to load from a checkpoint.
  # load_model:   ".checkpoints/checkpoint_200.ckpt"

  # multiple options for model_type:
  # [ "single", "composite", ... ]
  model_type: "single"
  PointNet:
    input_dimension:      3
    classifications:      ["particle"]
    augmentations:
      jitter: 0.03
      flip:
        positions:     [0, 1]
        probabilities: [0.5, 0.5]
      shear:  0.2
      rotate:
        degrees:  [15]
        axis:     [2]
    number_of_augmentations:  5
    embedding_type:       "dynamic_edge_conv"
    number_of_embeddings: 4
    number_of_neighbors:  [5, 10, 15, 20]
    aggregation:          ["max", "add", "mean", "max"]
    embedding_mlp_layers: [
      [64, 128, 64],
      [64, 128, 64],
      [64, 128, 64],
      [64, 128, 64]
    ]
    embedding_activations: ['leru', 'leru', 'leru', 'leru']
    linear_output:        512
    mlp_output_layers:    [512, 256, 128, 64, 32]
    out_channels:         [10]
```



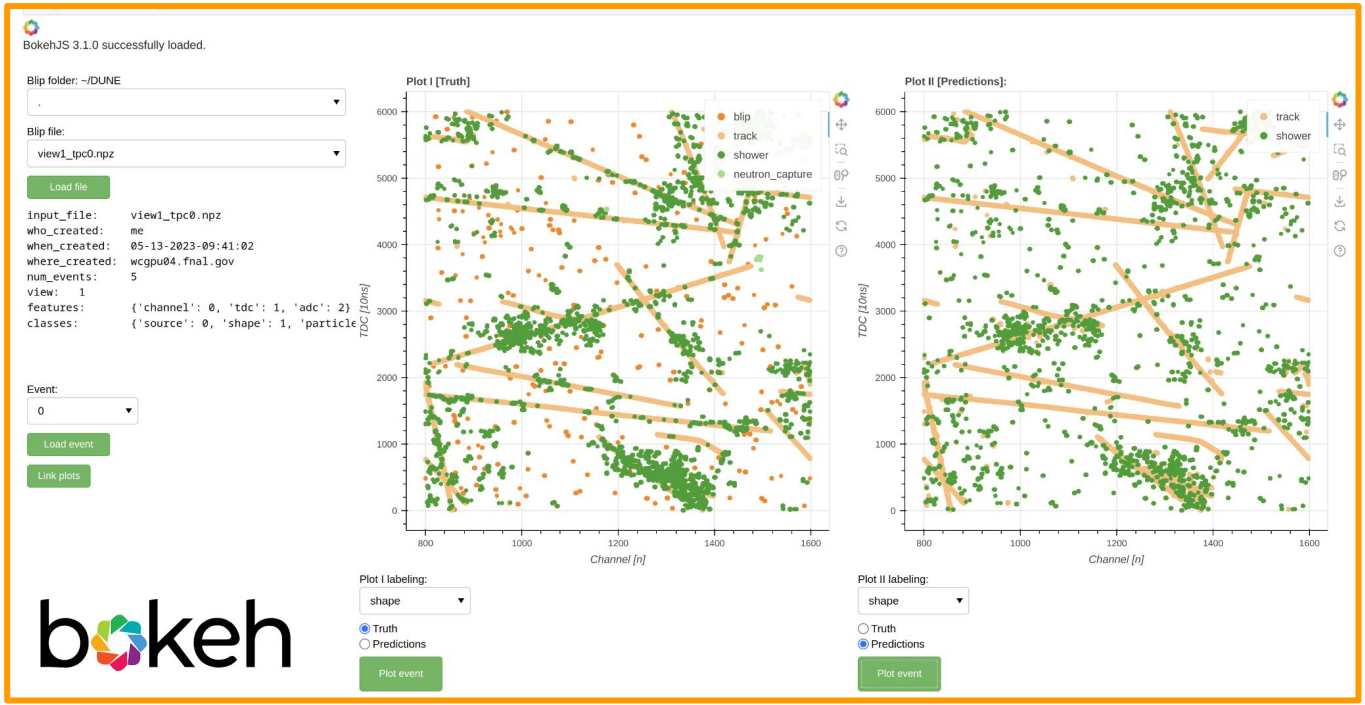


UCDAVIS

Event display

We are also working on an event display for BLIP which has the following features:

- Uses *bokeh* to create an interactive GUI.
- GUI can be run in a Jupyter notebook, or as a stand-alone html page.
- Obtain **point by point information** to quickly diagnose/access Arrakis and BLIP performance.
- Interface for analysis on network output.
- The BLIP-display could be made accessible through a **Wilson Cluster** jupyter interface.

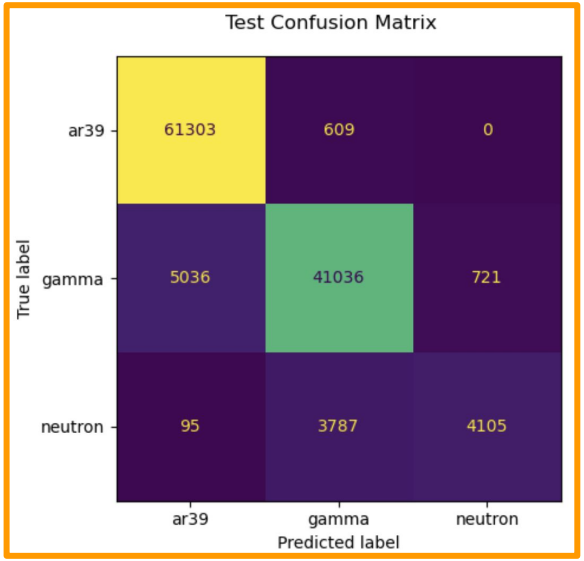


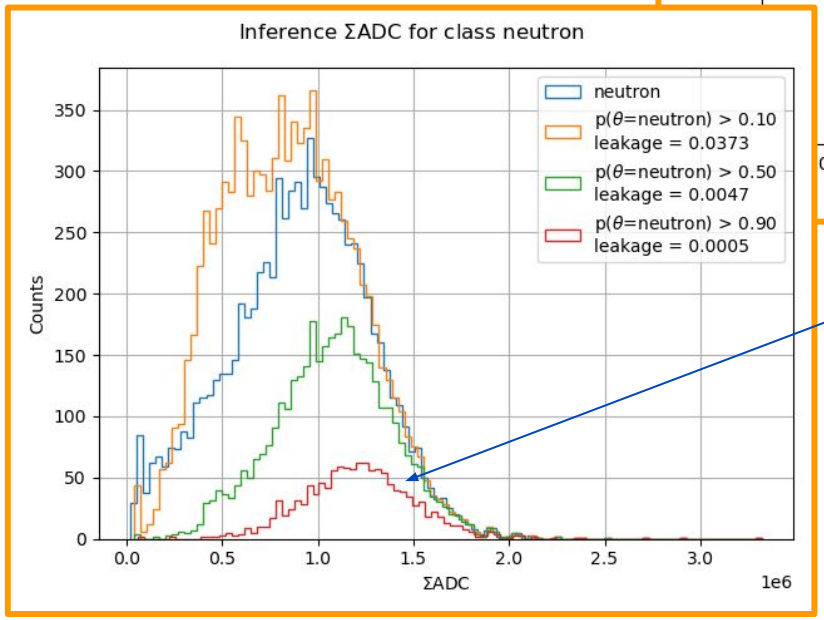
UCDAVIS

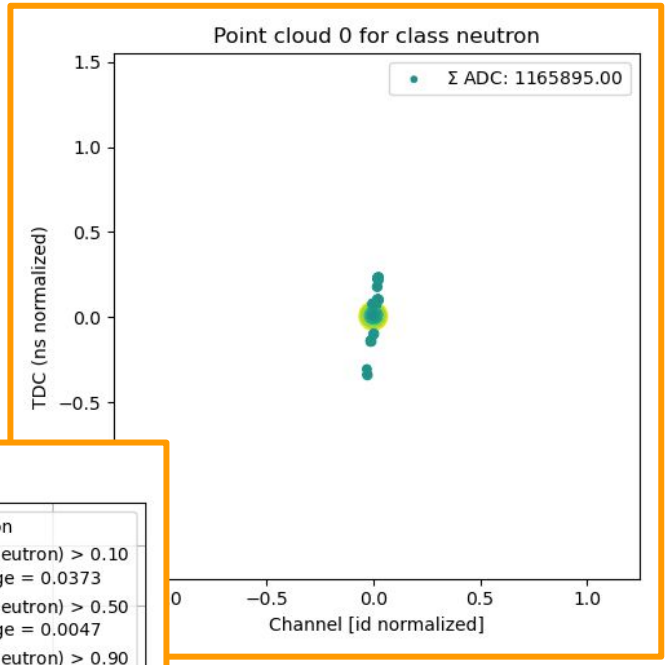
Preliminary PointNet Results

We trained a BLIP model on 116K point clouds with a 70/30 train/test split and the following classes:

- Argon-39: 61,912 events (~53%)
- Capture Gammas: 46,793 events (~40%)
- Neutron Captures: 7,987 events (~7%)







Can calibrate against this distribution!

UCDAVIS