Neutron Calibration with Machine Learning

DUNE Collaboration Meeting: CERN (05/21/2023)

Presented by *Nicholas Carrara* on behalf of the **PNS group** at **UC Davis** and **South Dakota School of Mines**:

Michael Mulhearn, Emilija Pantic, Robert Svoboda, Jingbo Wang

Yashwanth Bezawada, Junying Huang, Walker Johnson

Many thanks to Dominic Brailsford, Ken Herner, Tom Junk and Kyle Knoepfel!

Nuclear Science & Security Consortium

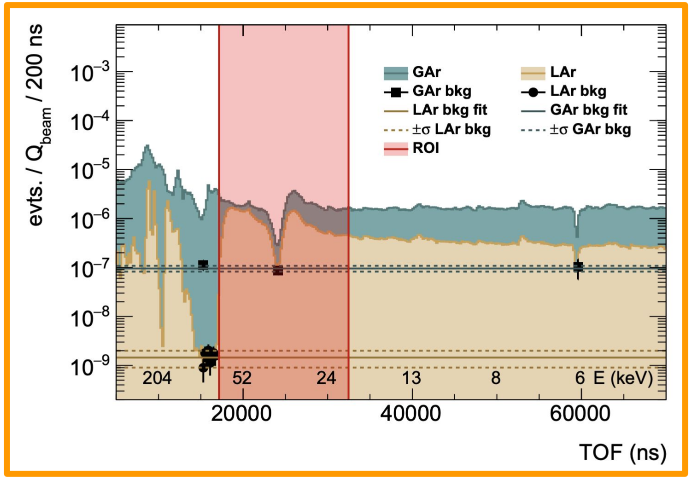
Neutron Calibration

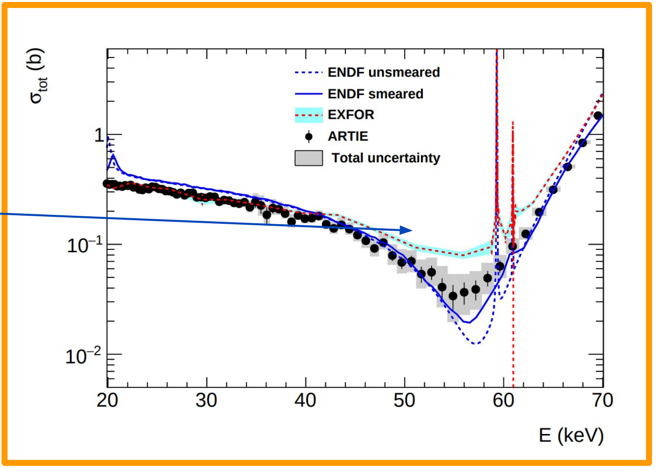
Benefits of low-energy neutrons for calibration:

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

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

- ***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 ***higher precision follow-up*** planned for this year.





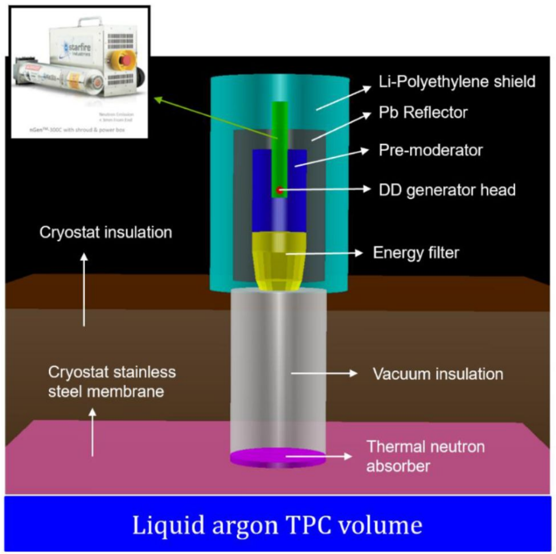
1 *Measurement of the total neutron cross section on argon in the 20 to 70 keV energy range,* The ARTIE Collaboration, In review at PRL, 2023, (https://arxiv.org/abs/2212.05448).

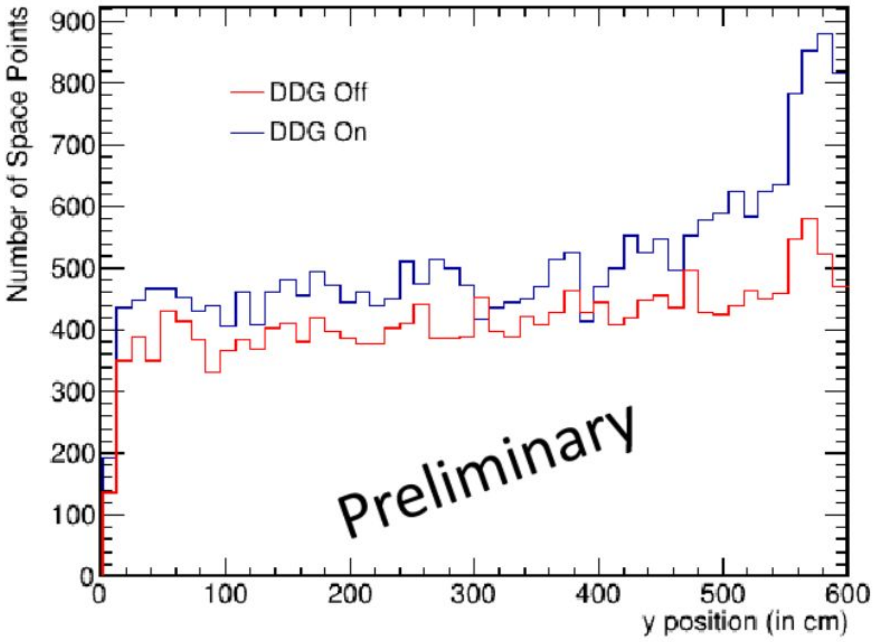
UCDAVIS

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***.





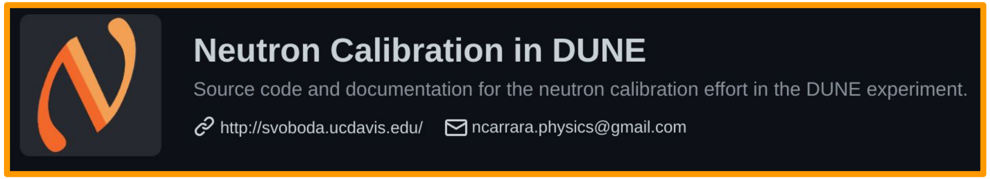
Work done by Y. Bezawada and J. Huang

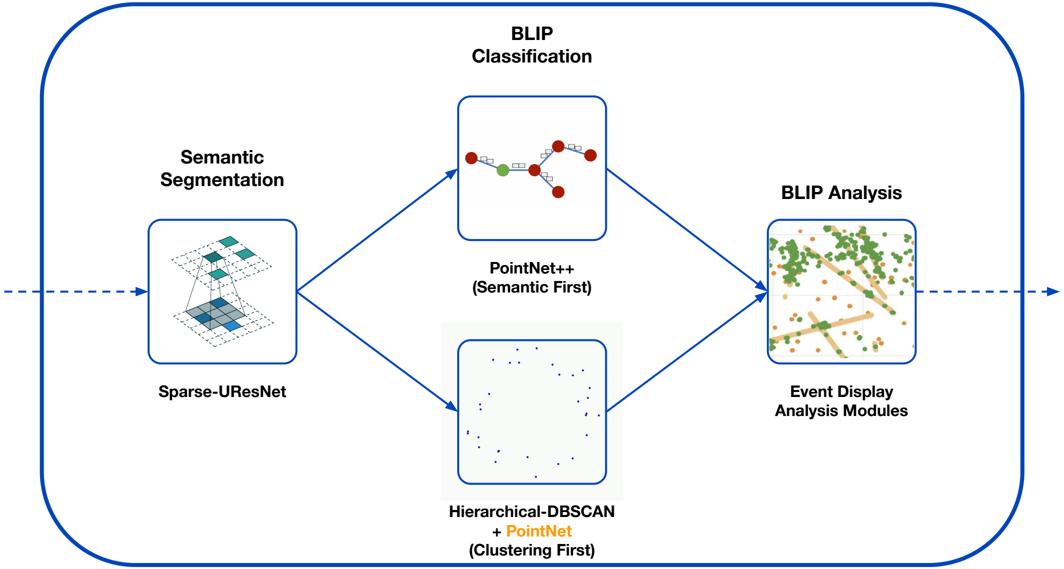
UCDAVIS

BLIP: (Blips and Low-energy Interaction PointNet)

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**:
  - PointNet++/PointNet 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).



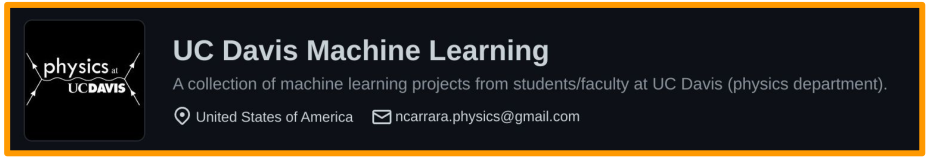


BLIP

UCDAVIS

2 *BLIP: Blips and Low-energy Interaction PointNet,* DUNE PNS Group (UC Davis), 2023, (https://github.com/Neutron-Calibration-in-DUNE/Blip).

Arrakis LArSoft Module



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

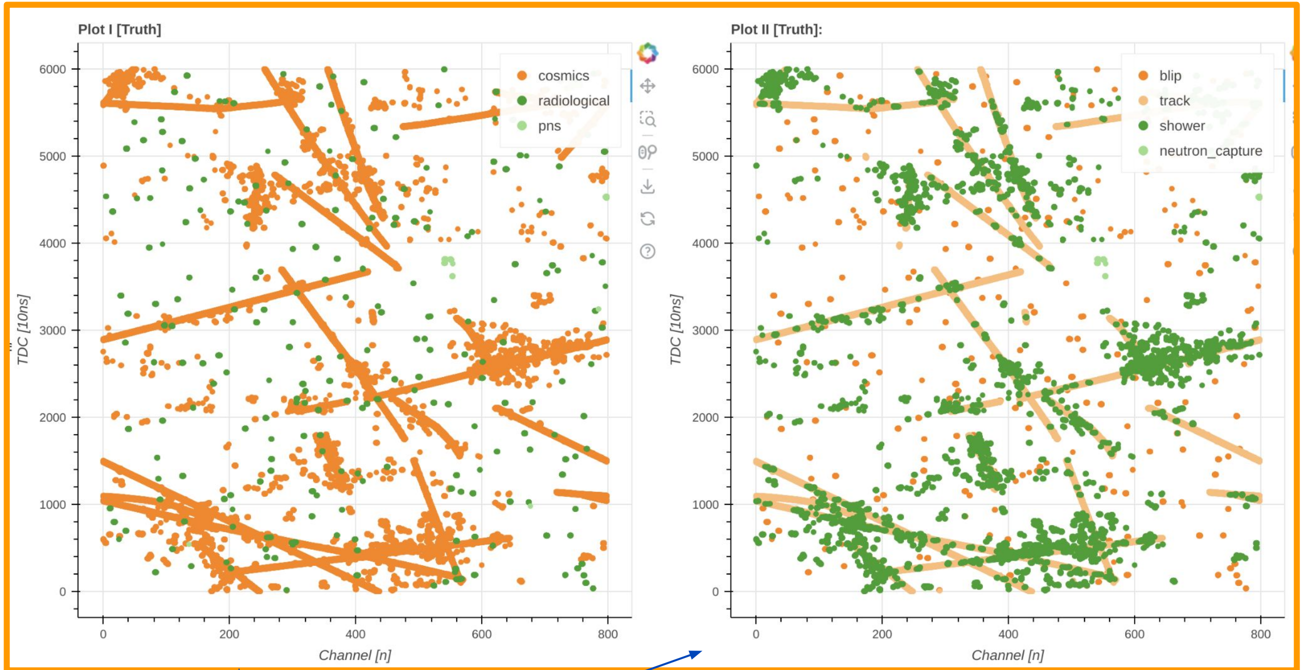
Current Labeling Schemes:

[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]

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



'source' and 'shape' labels for a simulated ProtoDUNE event

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

UCDAVIS

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.

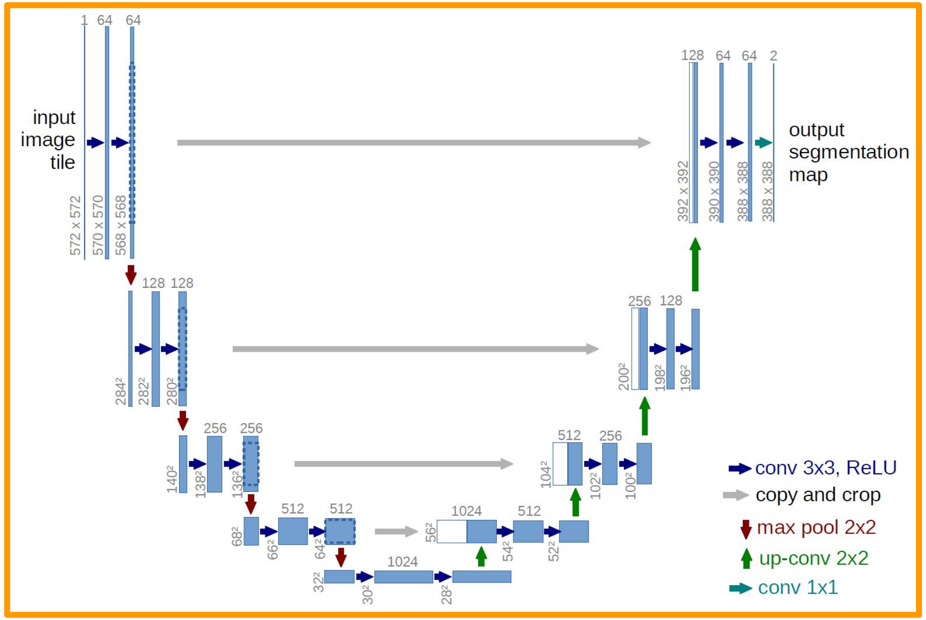
FHiCL parameters

```
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
```

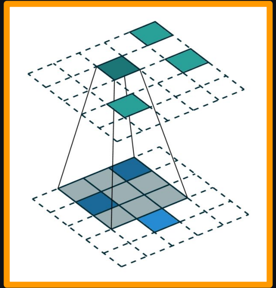
Config file training/evaluation

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.



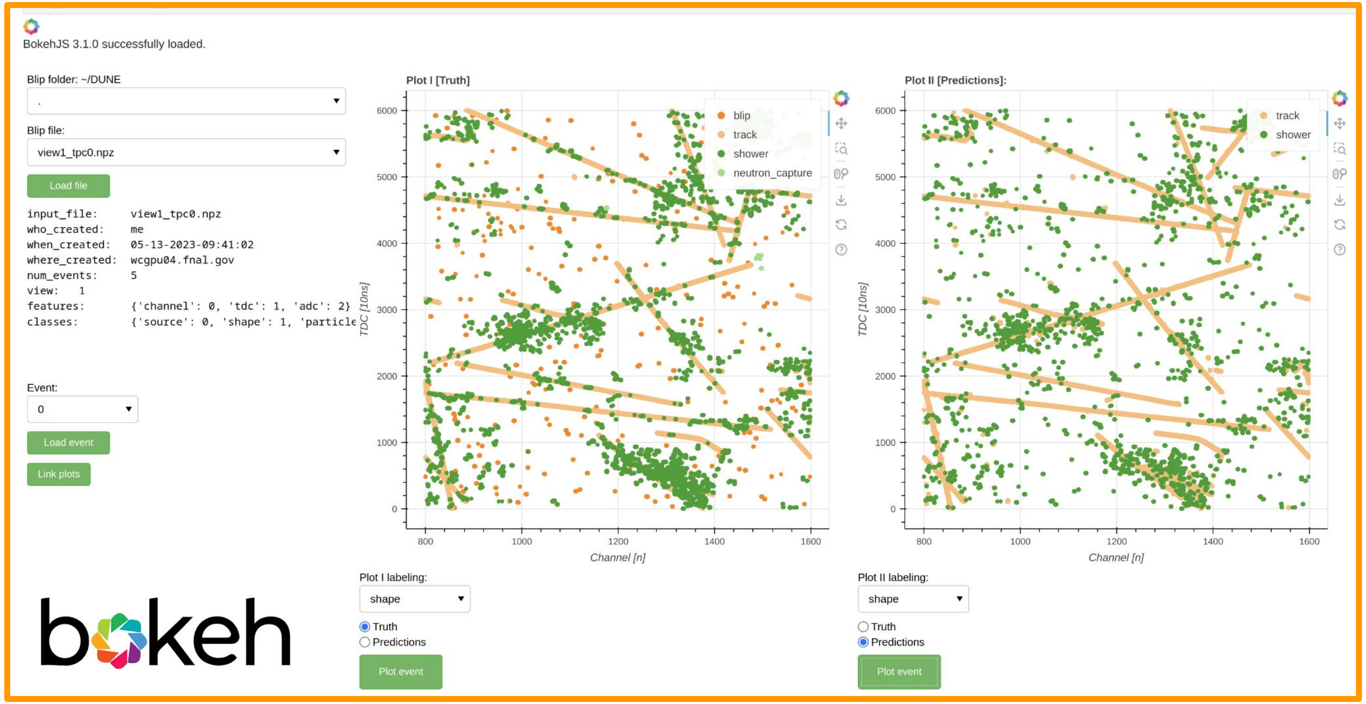

```
training:
  epochs:       150
  checkpoint:   10
  progress_bar: 'all'   # train, validation, test, all
  rewrite_bar:  False   # wether to leave bars after each epoch
  save_predictions: True  # wether to save network outputs in original file
  no_timing:  False     # wether to keep timing/memory info in callback
  gpu:        True
  gpu_device: 0
  seed:       42

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:
  # [ "SparseUNet", "PointNet", "UNet", ... ]
  # or a Composite.
  model_type:       "SparseUResNet"
  in_channels:      1
  classifications:  ["shape"]             # {"source", "shape", "particle"}
  out_channels:     [7]                   # {8, 7, 32}
  filtrations:      [64, 128, 256, 512]   # the number of filters in each downsample
  double_conv_params:
    kernel_size:  3
    stride:       1
    dilation:     1
    activation:   "relu"
    dimension:    2
    batch_norm:   True
  conv_transpose_params:
    kernel_size:  2
    stride:       2
    dilation:     1
    dimension:    2
  max_pooling_params:
    kernel_size:  2
    stride:       2
    dilation:     1
    dimension:    2
```



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.

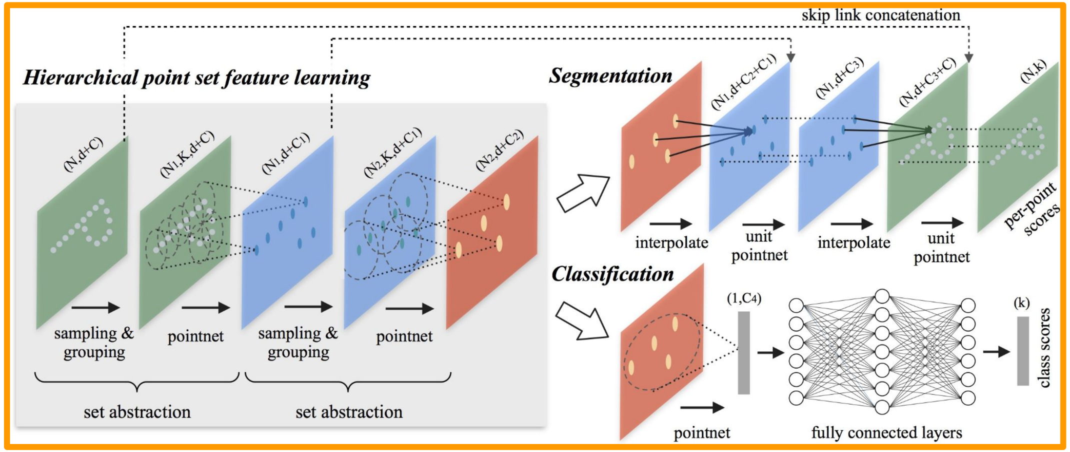


UCDAVIS

BLIP Classification

BLIP Classification will consist of at least two main 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]).



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

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

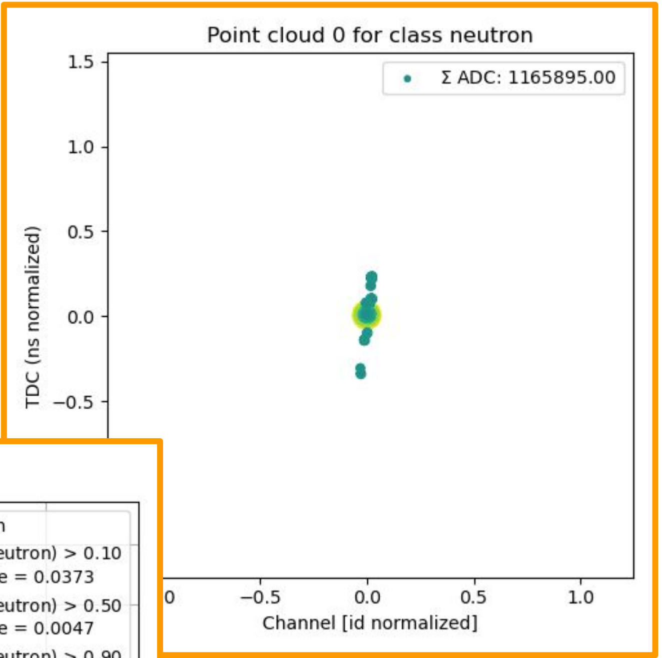
4 *PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation,* C. Qi et. al., CVPR 2017, (https://arxiv.org/abs/1612.00593).
5 *PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space*, C. Qi et. al., (https://arxiv.org/abs/1706.02413)
6 *Dynamic Graph CNN for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

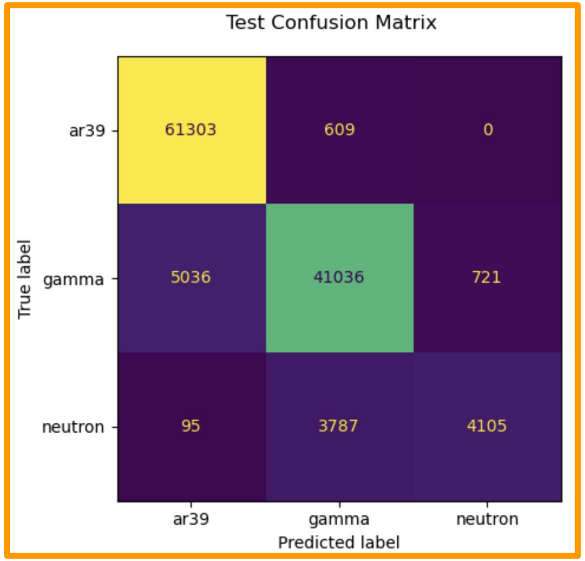
UCDAVIS

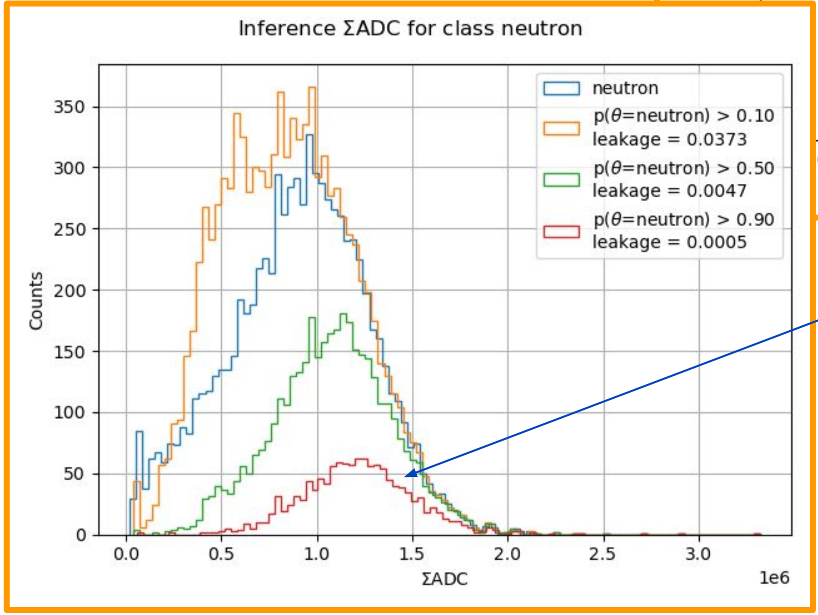
Preliminary PointNet Results

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

Work in Progress

Arrakis [Need a more informative name!]:

- ✓ 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:

- ❑ 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