LA-UR-24-23812

Approved for public release; distribution is unlimited.

Title: An Improved Boosted Decision Tree Framework for Event Classification in MicroBooNE

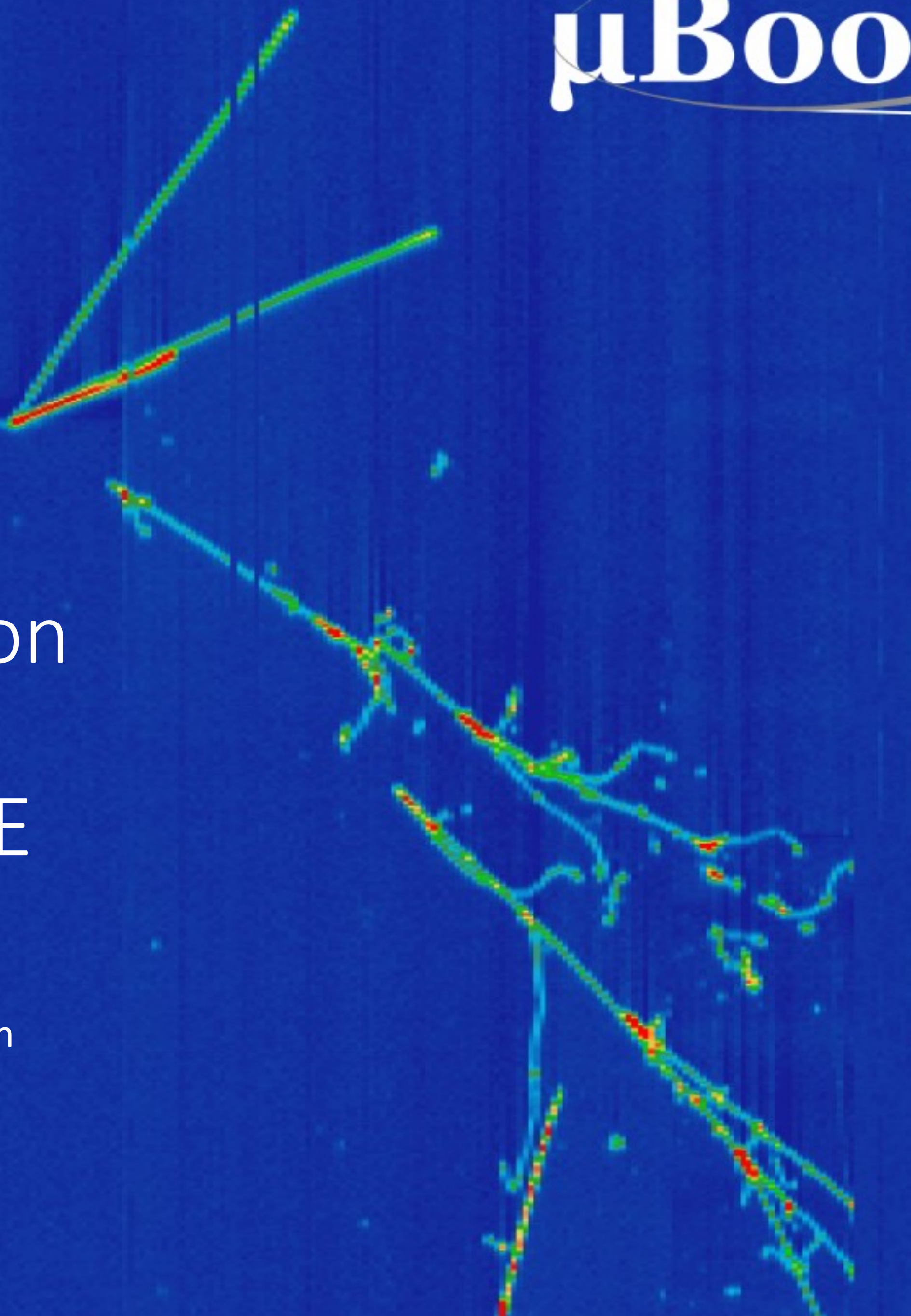
Author(s): Gollapinni, Sowjanya
Xing, Daniel Chris
Ross-Lonergan, Mark Paul

Intended for: Spring 2024 CCI/SULI Presentation and Peer Review Session (LANL internal)

Issued: 2024-04-23

Los Alamos
NATIONAL LABORATORY

μBooNE



An Improved Boosted Decision Tree Framework for Event Classification in MicroBooNE

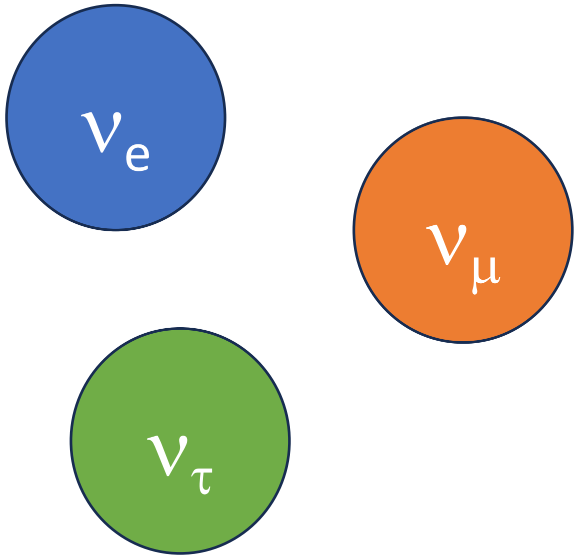
Daniel Xing, on behalf of the MicroBooNE Collaboration

Spring 2024 CCI/SULI Presentation and Peer Review Session

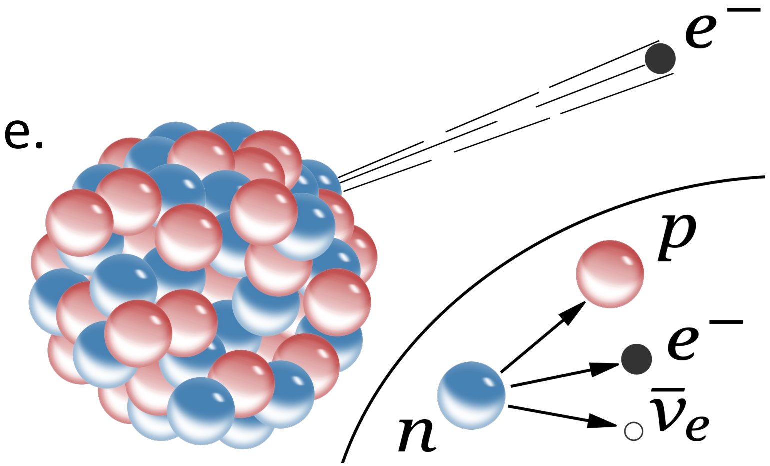
April 24, 2024

Neutrinos

Neutrinos

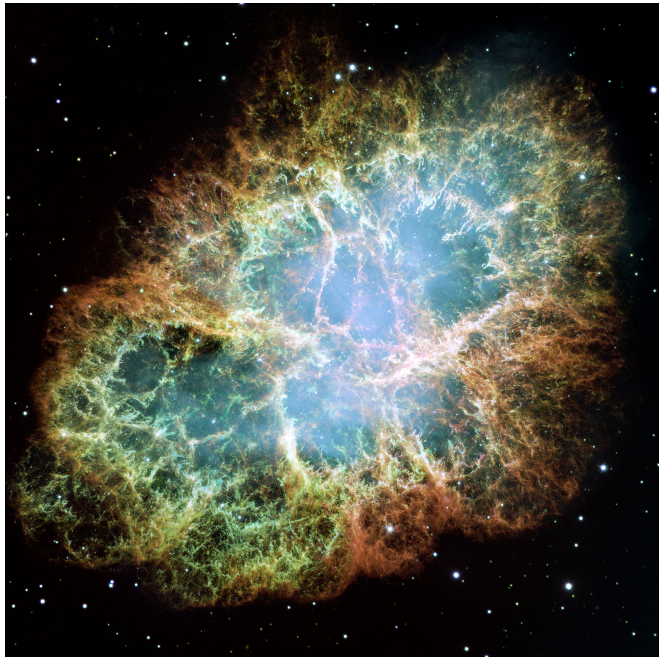


- Neutrinos are neutral particles with very low mass.
- Interact very rarely and are hard to detect.
    - Trillions of neutrinos pass through your body every second and in your lifetime, you have a ¼ chance of one interacting in your body.
- Come in three flavors: electron, muon, and tau.
- Over long distances, a neutrino can mysteriously change its flavor (known as oscillation).
- Are produced by beta decay, nuclear reactions, and in supernovae.



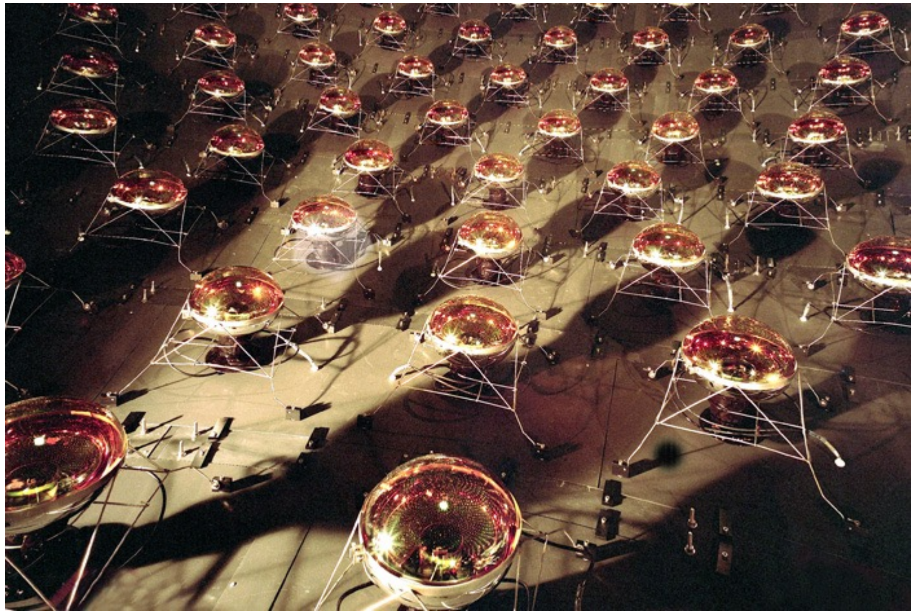
Credit: [Wikimedia Commons, Inductiveload]

Credit: NASA, ESA, J. Hester, A. Loll (ASU)



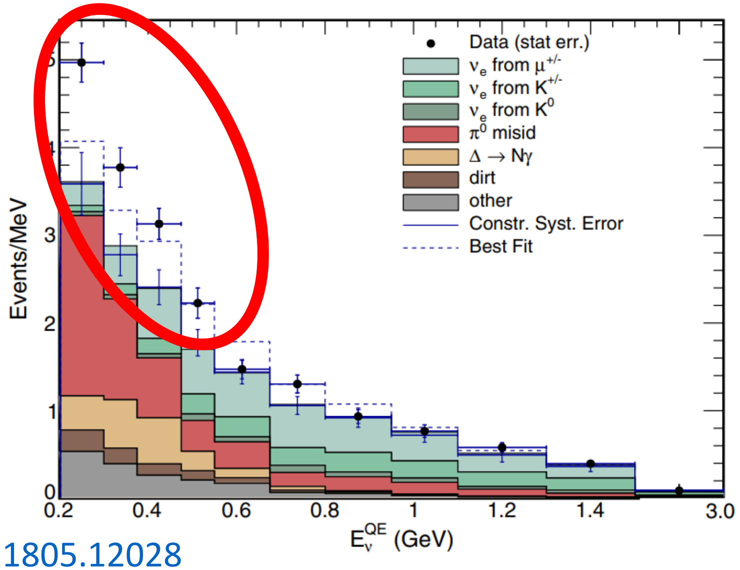
Why Neutrinos?

- The Standard Model of particle physics describes the four fundamental forces and the elementary particles.
- Neutrinos' mass and oscillation are not explained by the Standard Model.
- Processes involving neutrinos could help explain why there is more matter than anti-matter in the universe.
- Neutrinos from astrophysical sources can act as a leg of multi-messenger astronomy and provide information about supernovae and the early universe.
- Experimental anomalies suggest that our 3-flavor picture of neutrinos is incomplete.
- One of these anomalies was first discovered at LANL!

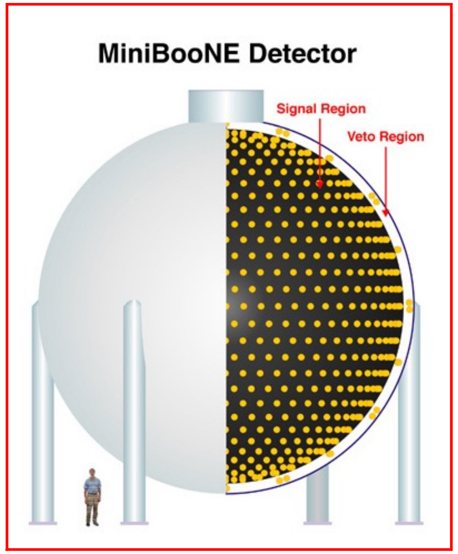


Credit: LANL

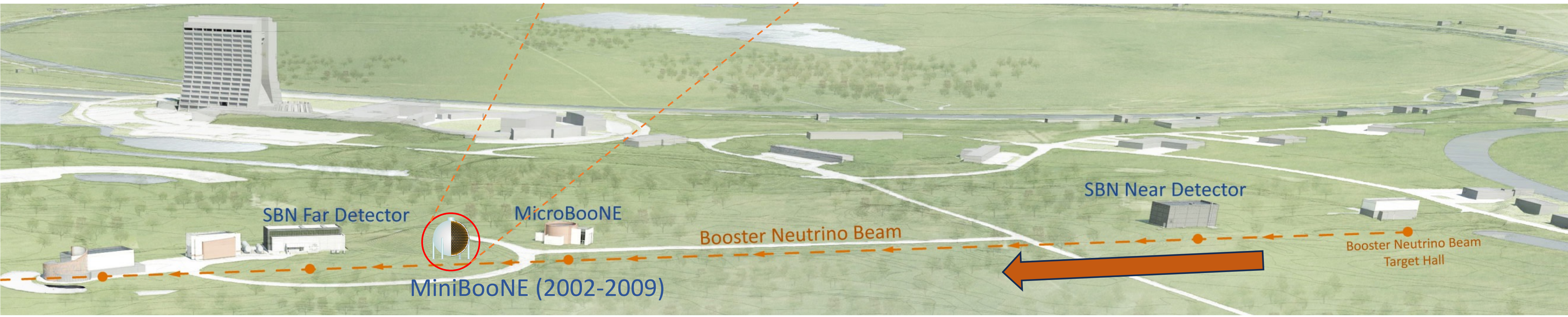
MiniBooNE Low Energy Excess Anomaly



1805.12028



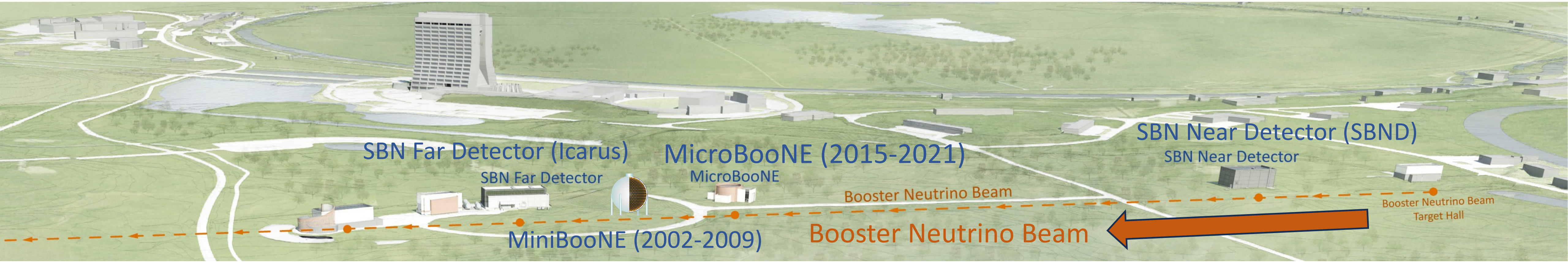
- MiniBooNE was a neutrino detector that found an anomalous excess of electron neutrino like events.
- This could indicate the existence of a fourth, “sterile” neutrino.
- However, MiniBooNE could not separate electrons from photons.



MicroBooNE

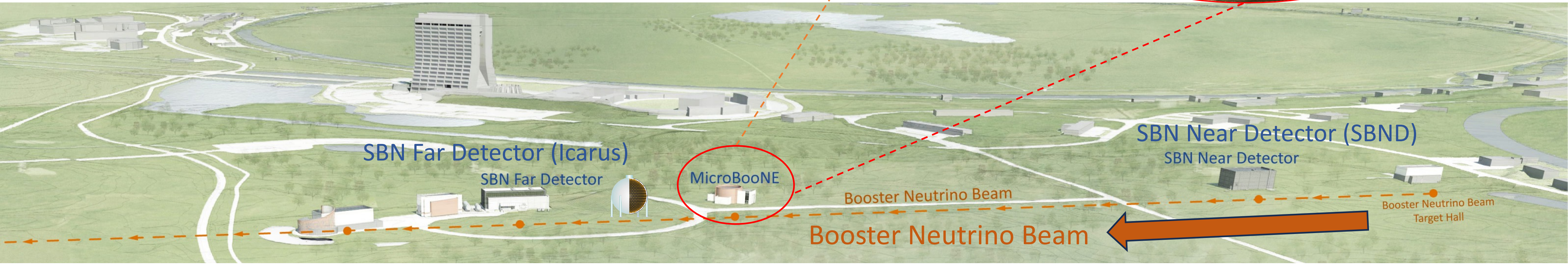
MicroBooNE

- MicroBooNE is a constituent of Fermilab's Short-Baseline Neutrino Program.
- Its primary purpose is to investigate the MiniBooNE Low Energy Excess.
- MicroBooNE collected data 2015 to 2021 and is currently being decommissioned.



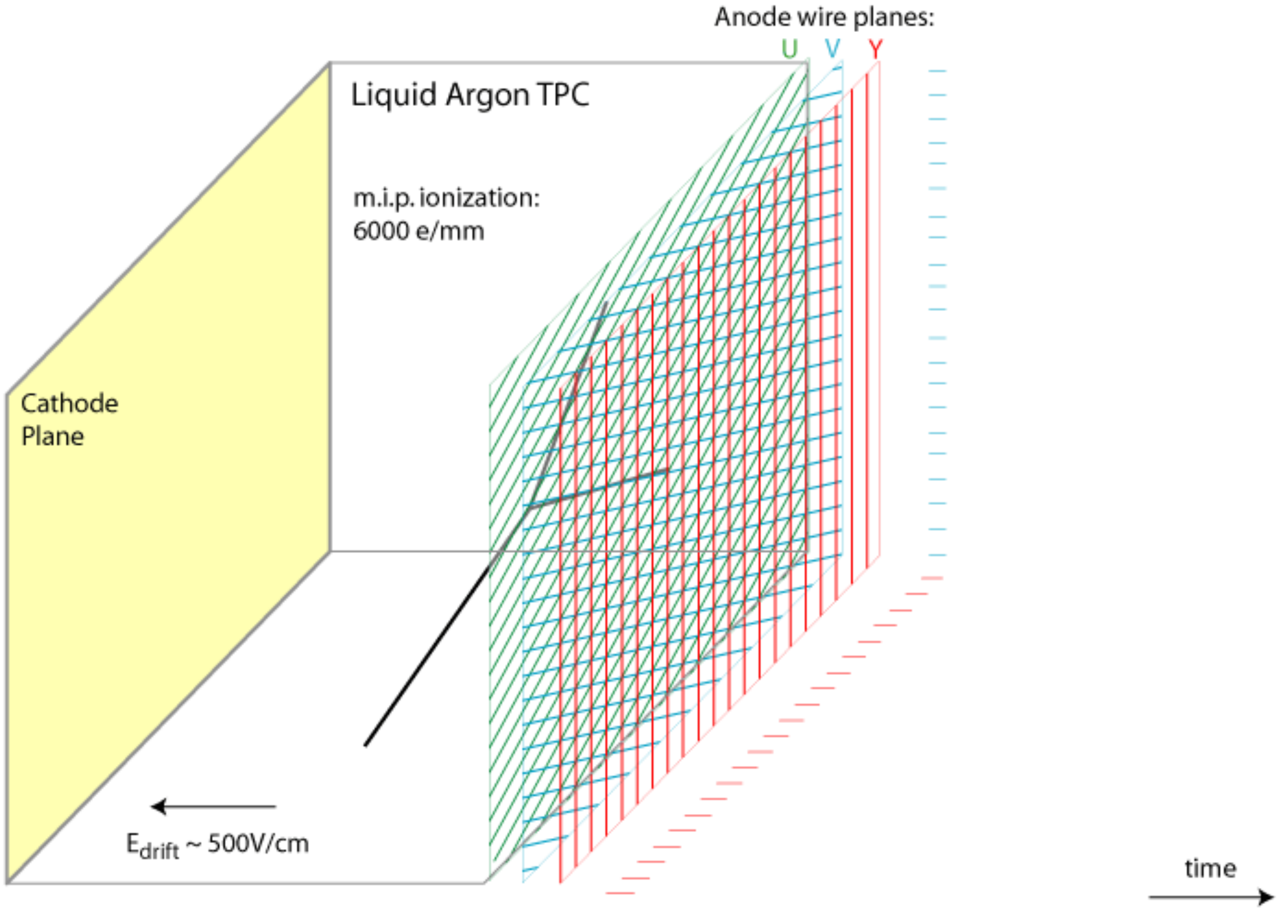
MicroBooNE

- It uses a 170-ton liquid-argon time projection chamber (LArTPC) to detect neutrinos in a neutrino beam

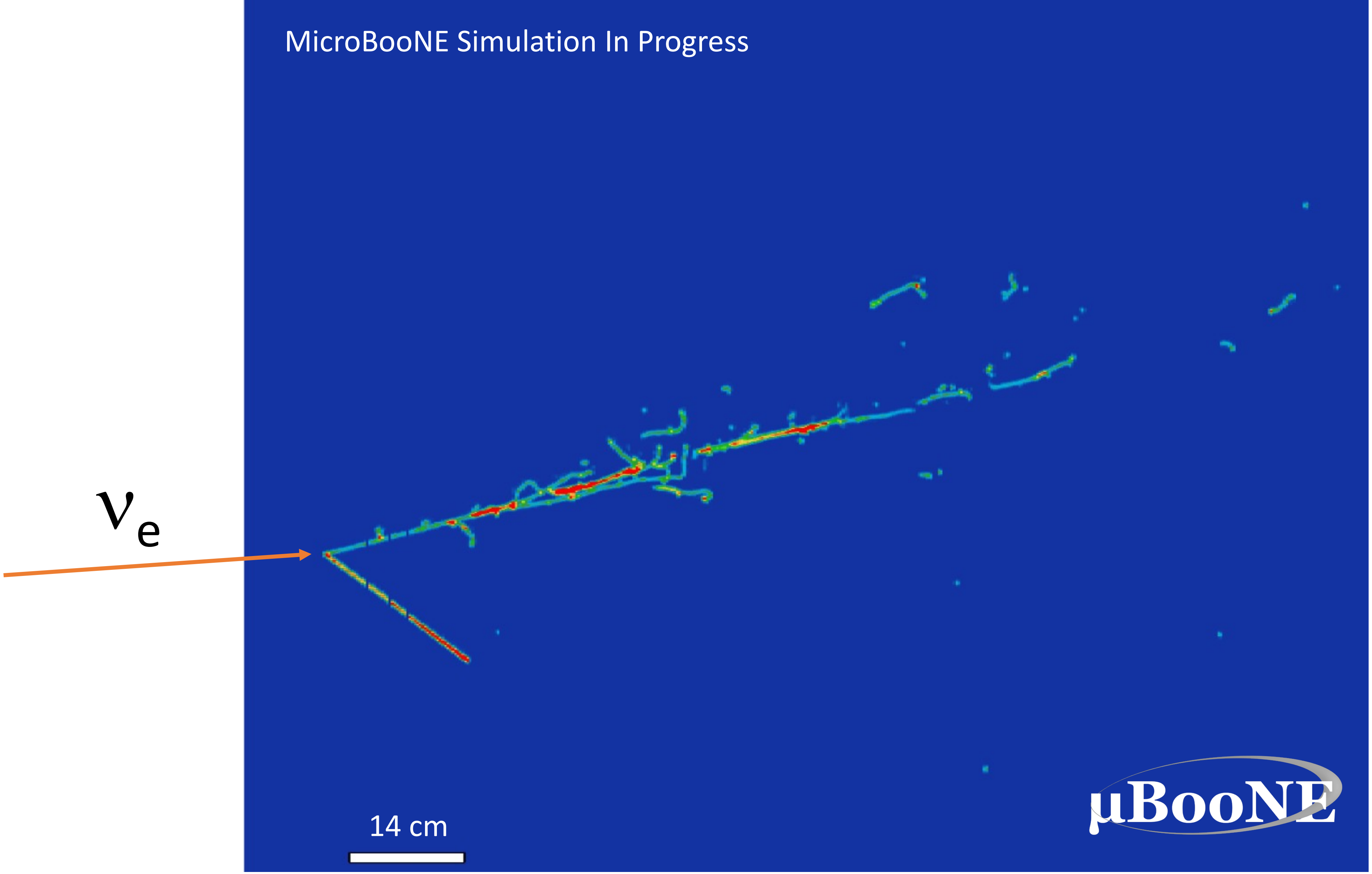


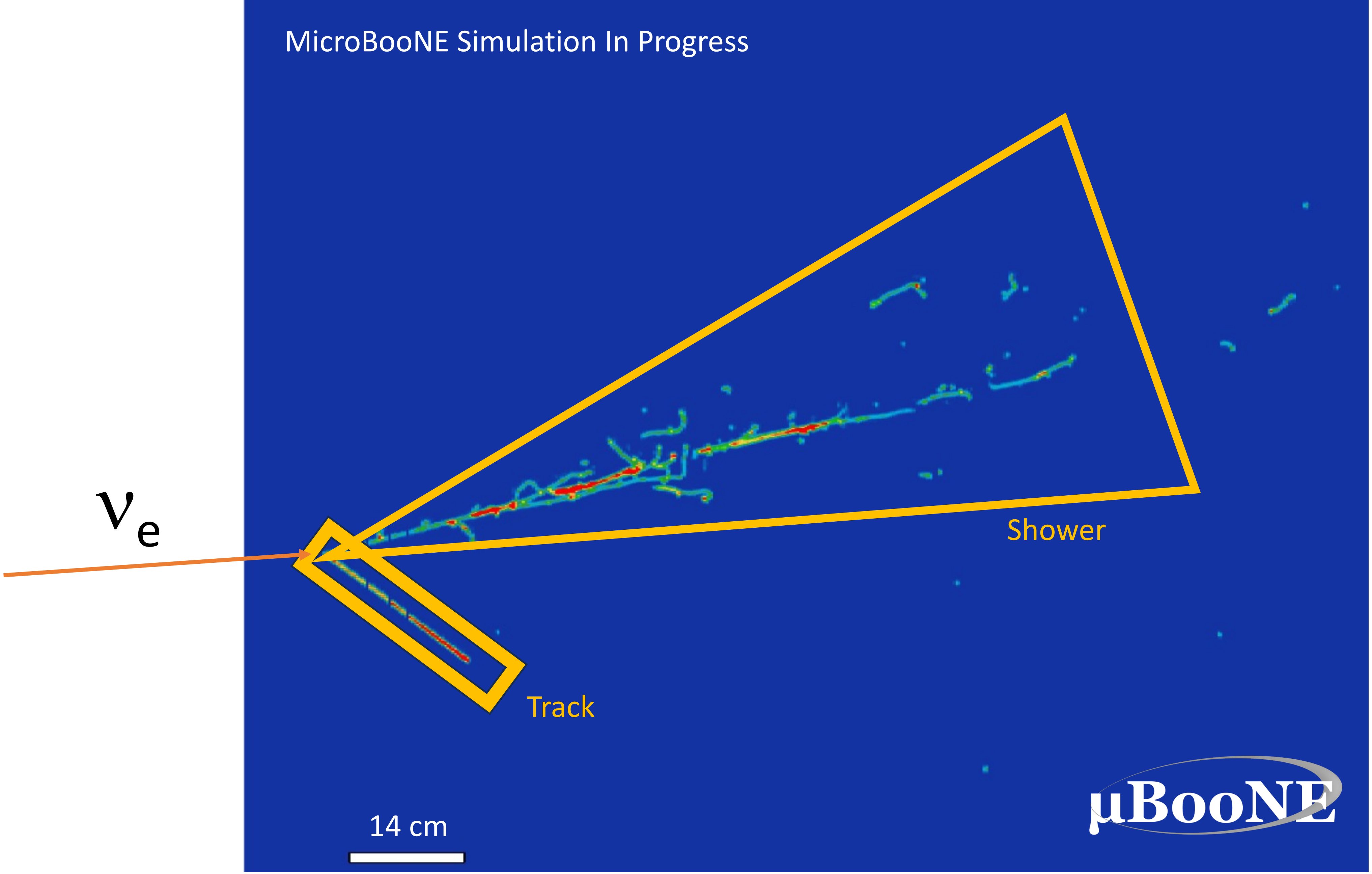
LArTPC Technology

Credit: [WireCell, BNL]



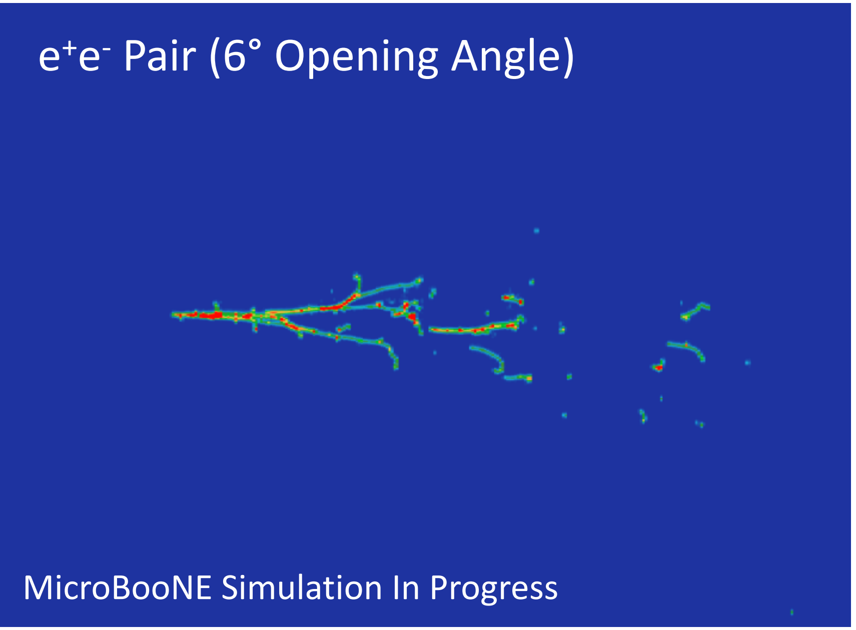
LArTPC technology spatially resolves the trail of ionization electrons produced by charged particles generated in neutrino interactions with Argon nuclei.

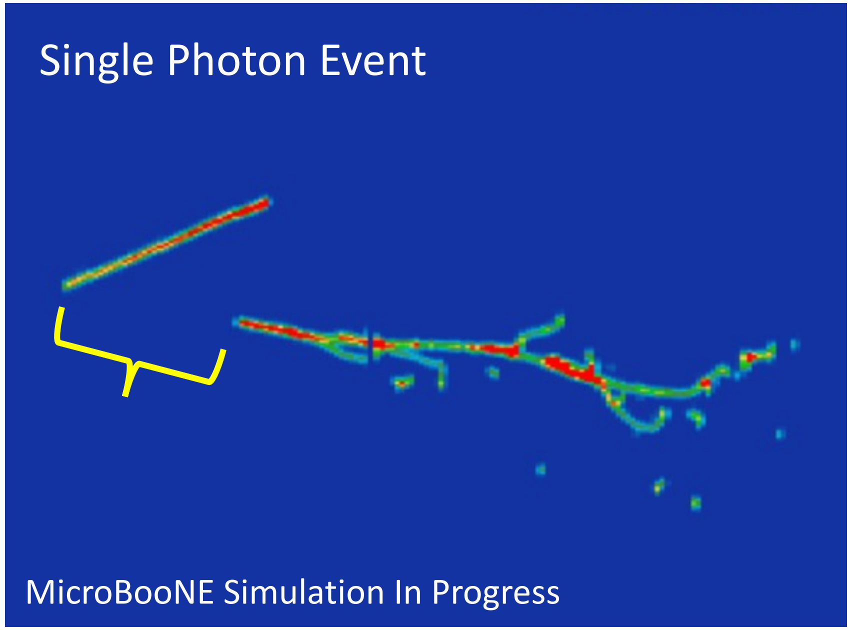

MicroBooNE Simulation In Progress
ν_e
14 cm
μBooNE

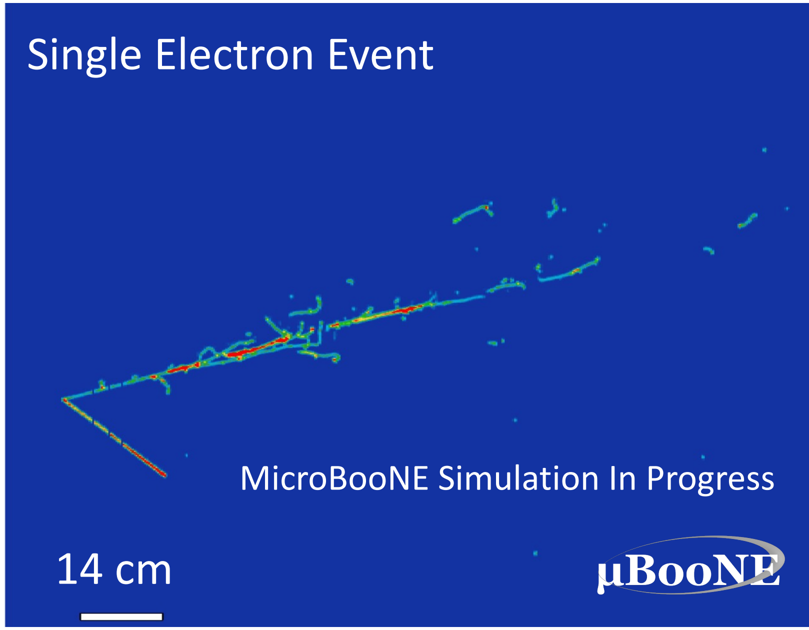

MicroBooNE Simulation In Progress
ν_e
Shower
Track
14 cm
µBooNE

Solving the Low Energy Excess

- Because MiniBooNE did not have LArTPC technology, it could not distinguish between electrons and photons like MicroBooNE.
- Separating photons, single electrons, and e^+e^- pairs is necessary to solve the MiniBooNE Low Energy Excess.
- There are 5 ongoing analyses in MicroBooNE, searching for various explanations for the excess.
- Given a set of data, these analyses want to be able to select most of the type of event they are interested in, with high purity.

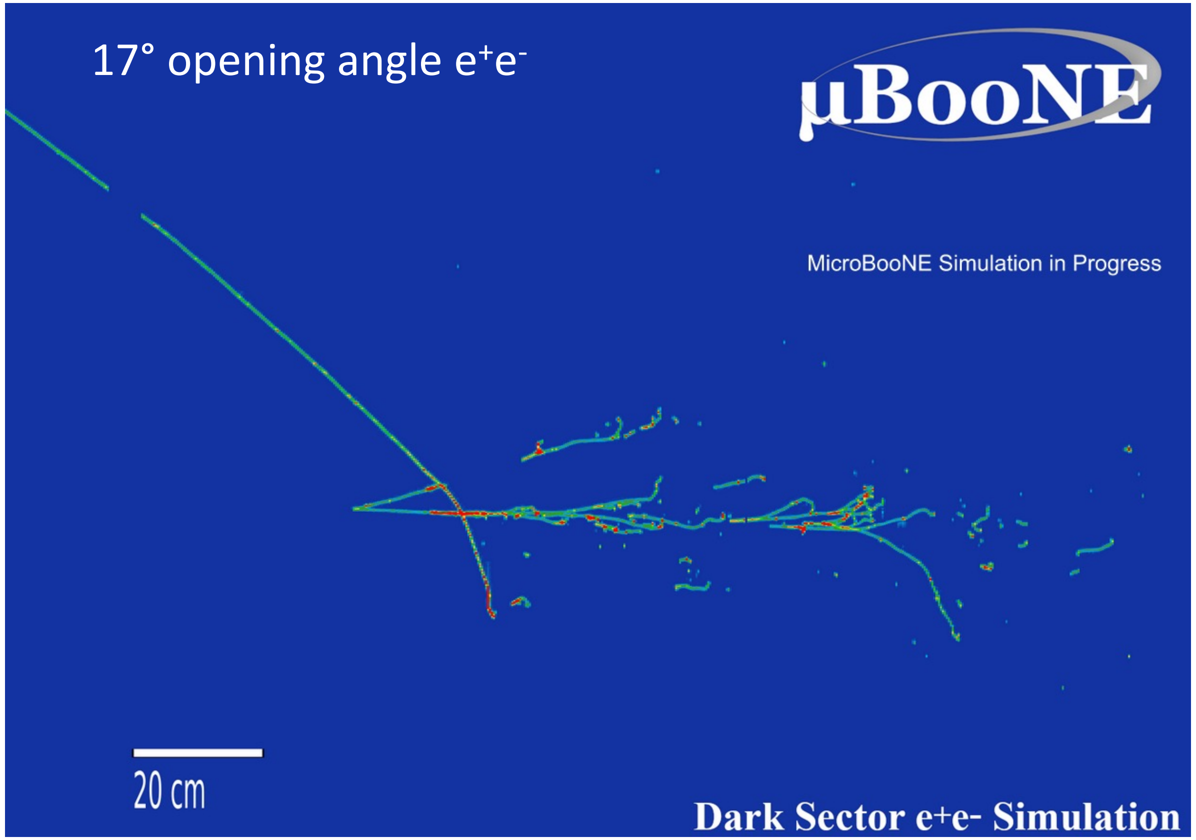


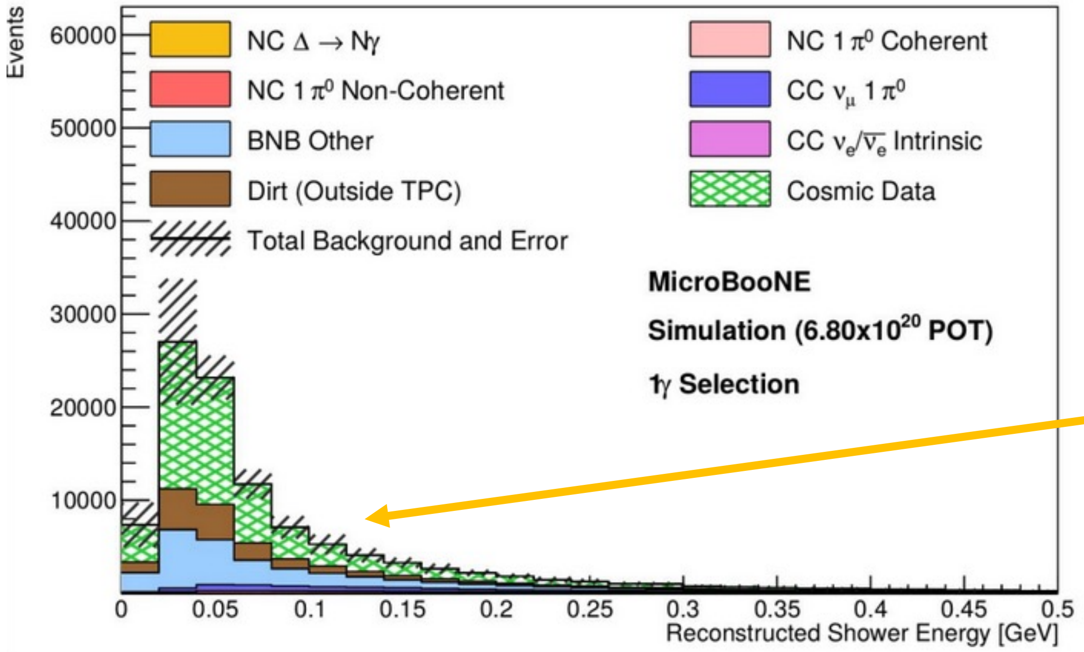




Why BDTs?

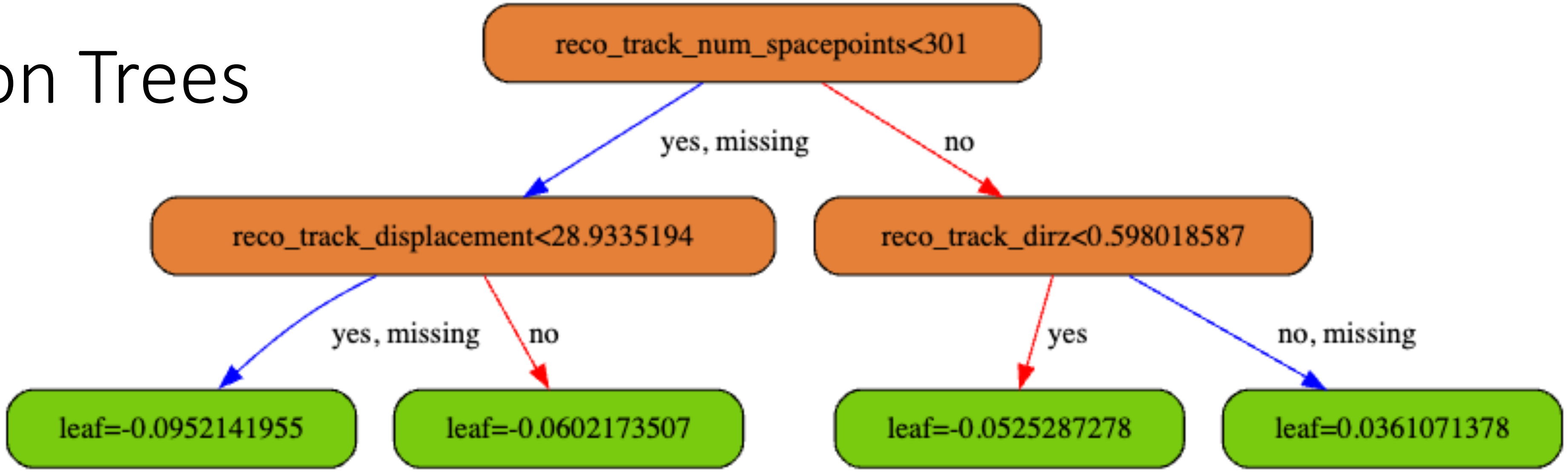
- We want a set of only one type of events like γ or e^+e^-, but we start with all our data which contains many background events.
- BDT are ML algorithms that can separate between different classes (signal and background)





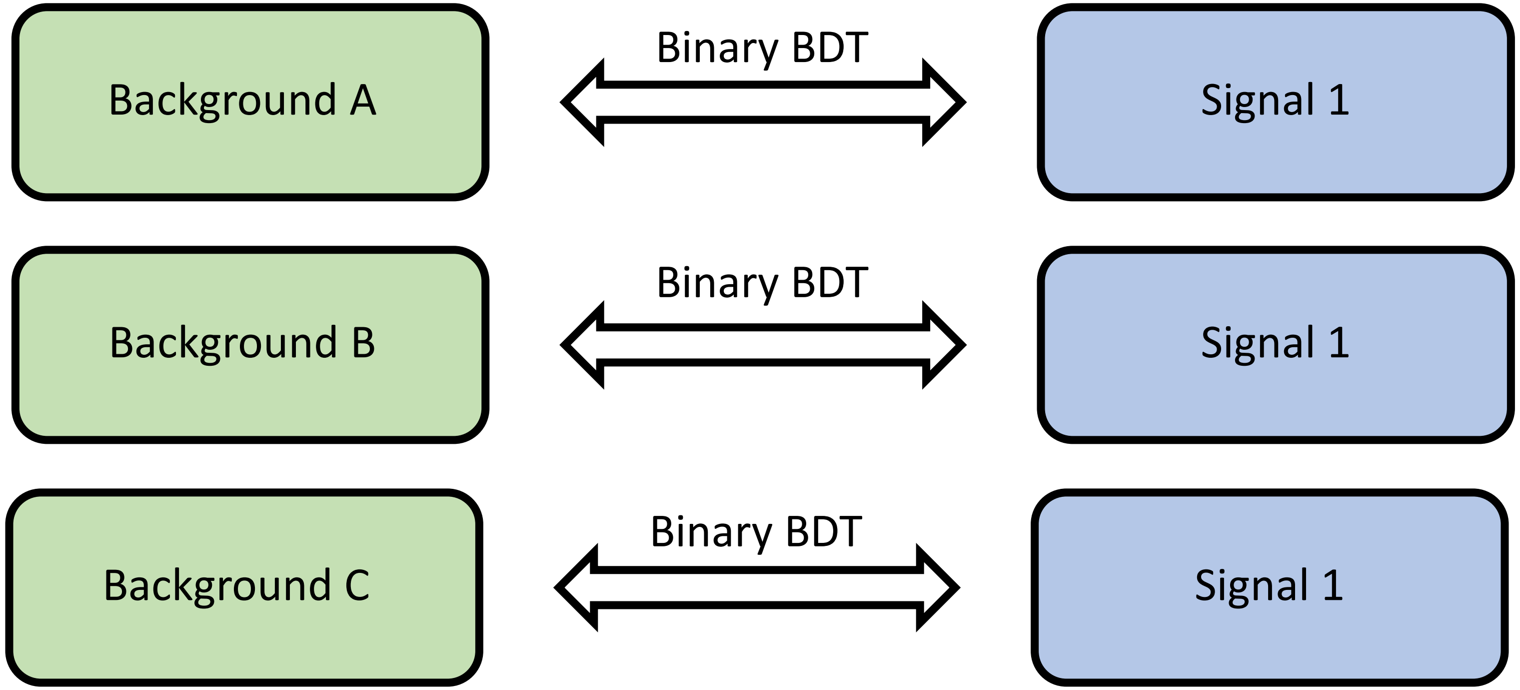
Can not even see the NC Δ -> Nγ signal, O(100) events!

Boosted Decision Trees



- BDTs combine many weak individual trees.
- They use the error of the previous tree to set the weights of the next tree.
- Take less training time than methods like Neural Networks.
- Can be used for regression, classification, or ranking.
- Can handle many data types, including categorical.
- Best suited for tabular data.

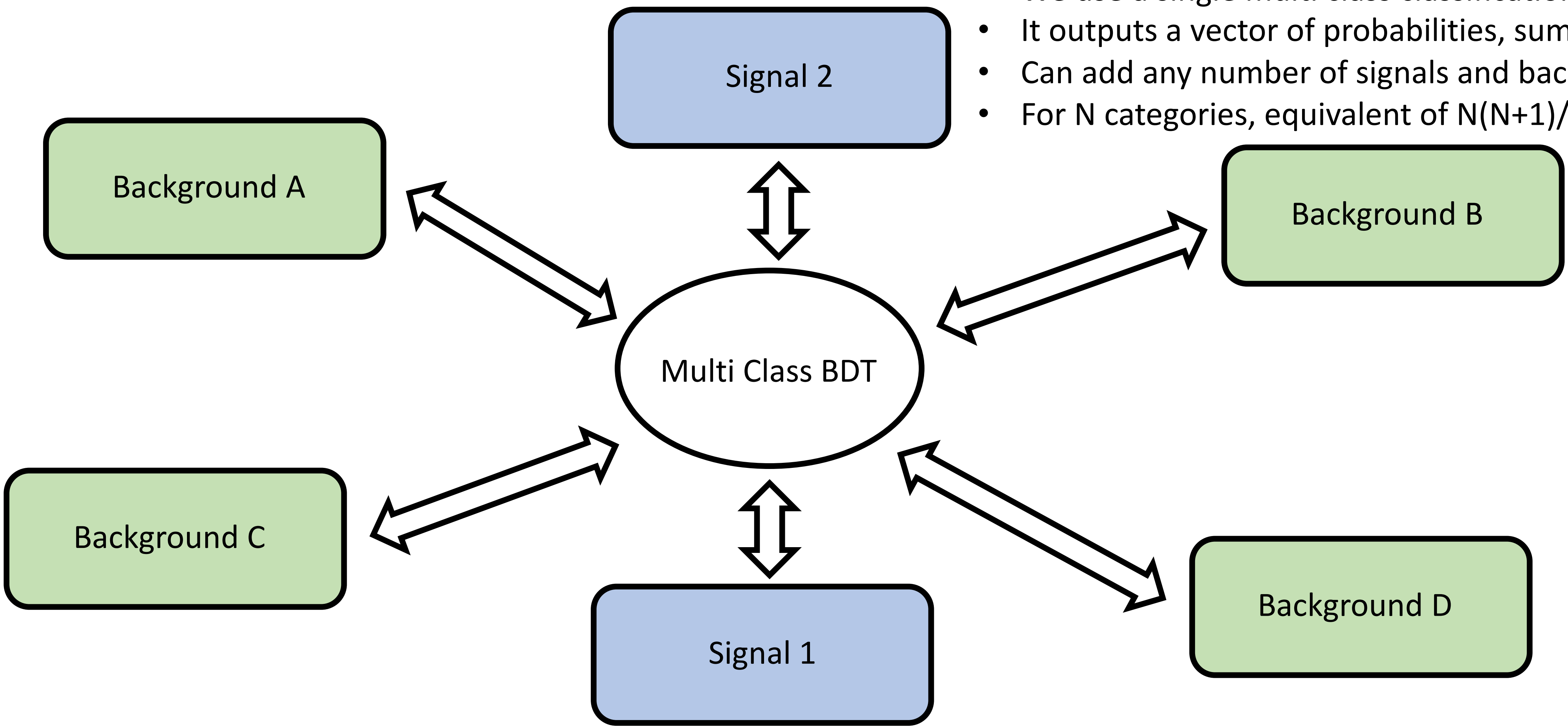
Understanding the Current Tools



- Currently, a binary classification BDT is trained for each background.
- If we add a second signal, it will require double the number of BDTs.
- There is no crosstalk between these BDTs, so there are lots of inefficiencies.

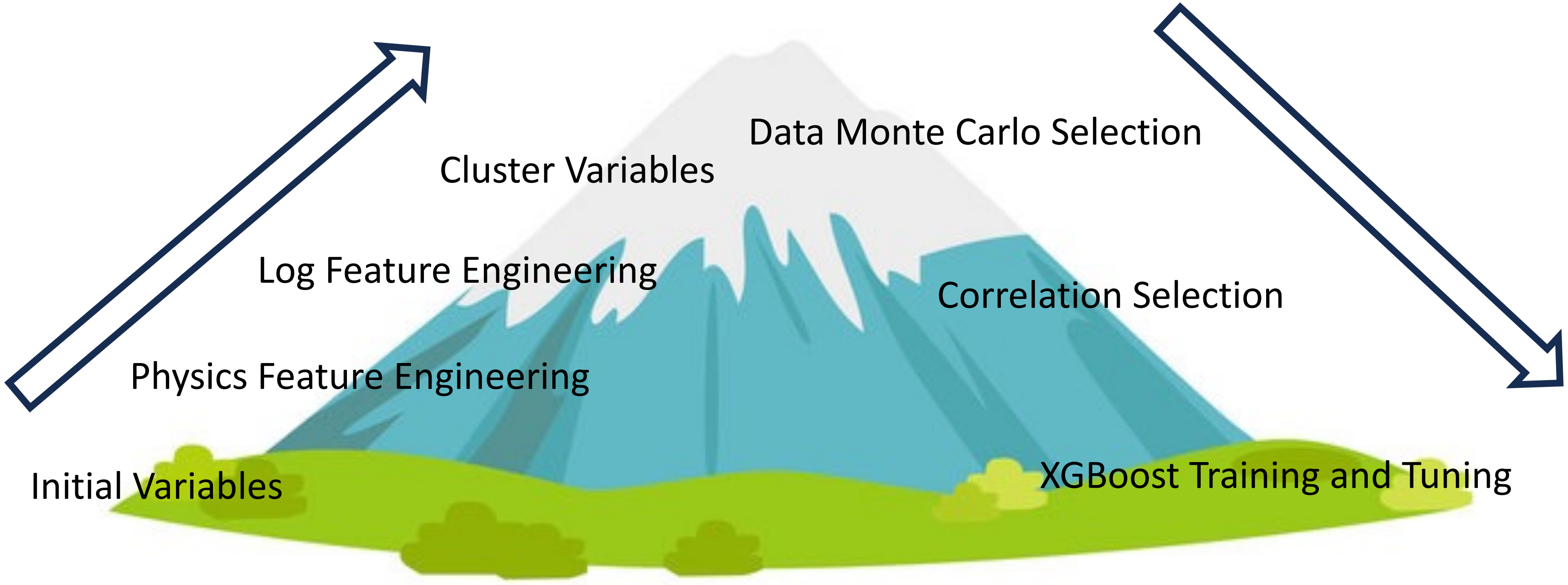
New Boosted Decision Tree Framework

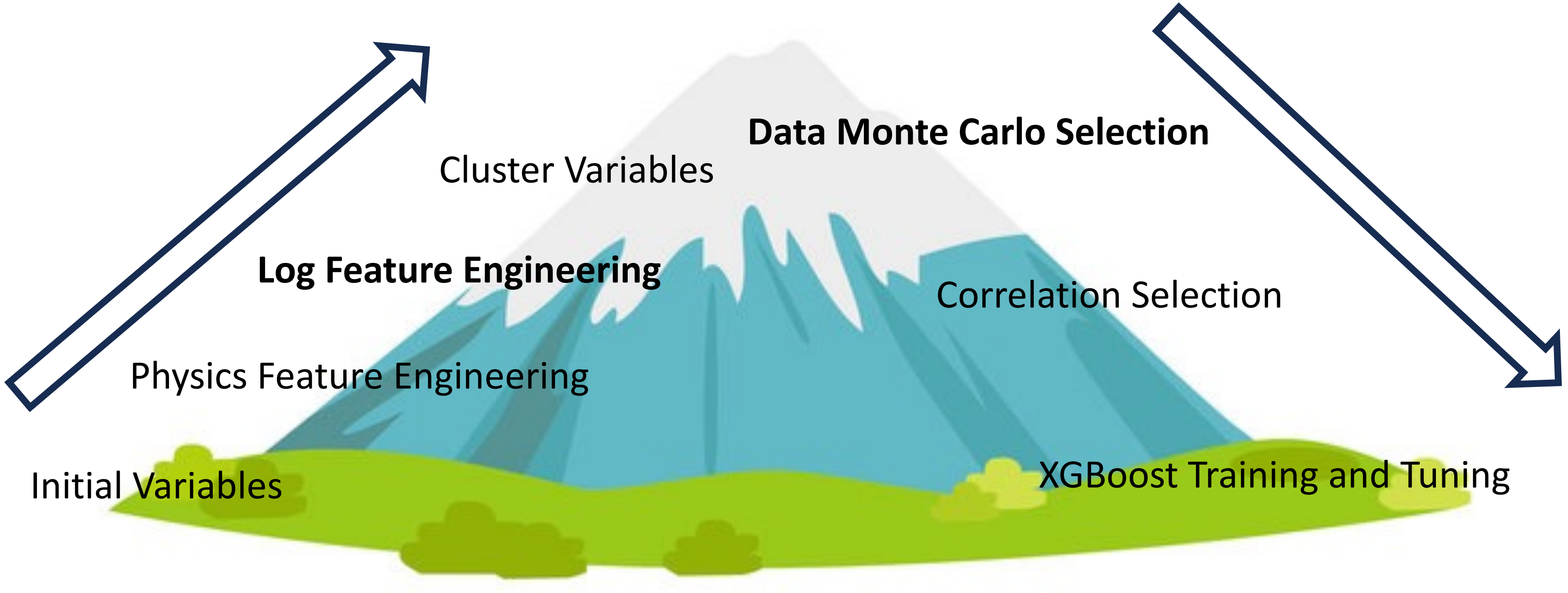
- We use a single multi class classification BDT.
- It outputs a vector of probabilities, summing to 1.
- Can add any number of signals and backgrounds.
- For N categories, equivalent of N(N+1)/2 binary BDTs.



New Boosted Decision Tree Framework

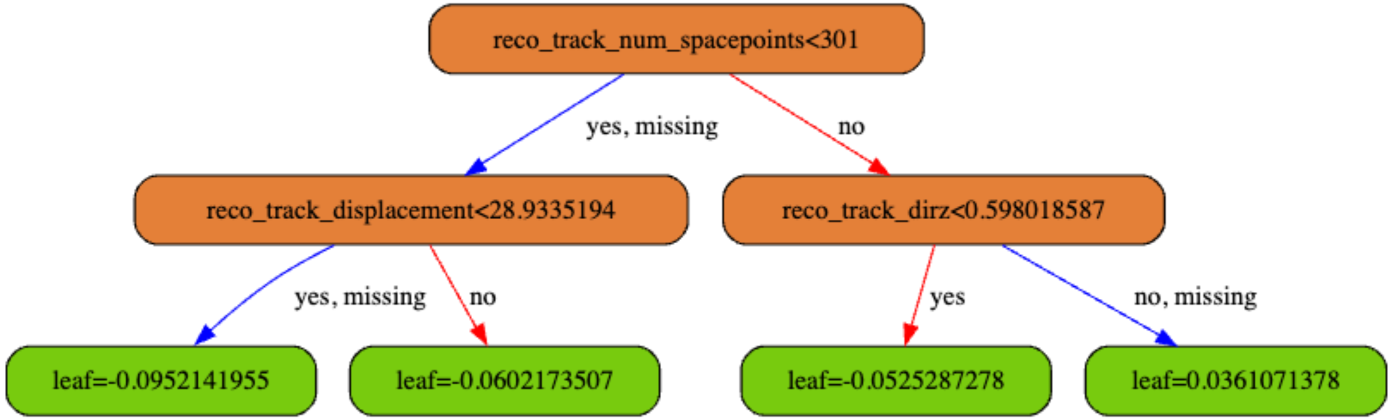
- Uses BDTs in Python with machine learning libraries like XGBoost and scikit-learn.
- Unlike previous BDTs written in C++, it can distinguish between any number of signals and backgrounds simultaneously.
- Feature engineering and feature selection are applied before training the single BDT which uses multi-class classification.
- Training is repeated with hyperparameter tuning, using AUC or balanced accuracy score as a metric.
- Initial Train and Test on Monte Carlo generated simulated data, later test with real data.


Data Monte Carlo Selection
Cluster Variables
Log Feature Engineering
Correlation Selection
Physics Feature Engineering
Initial Variables
XGBoost Training and Tuning


Data Monte Carlo Selection
Cluster Variables
Log Feature Engineering
Correlation Selection
Physics Feature Engineering
Initial Variables
XGBoost Training and Tuning

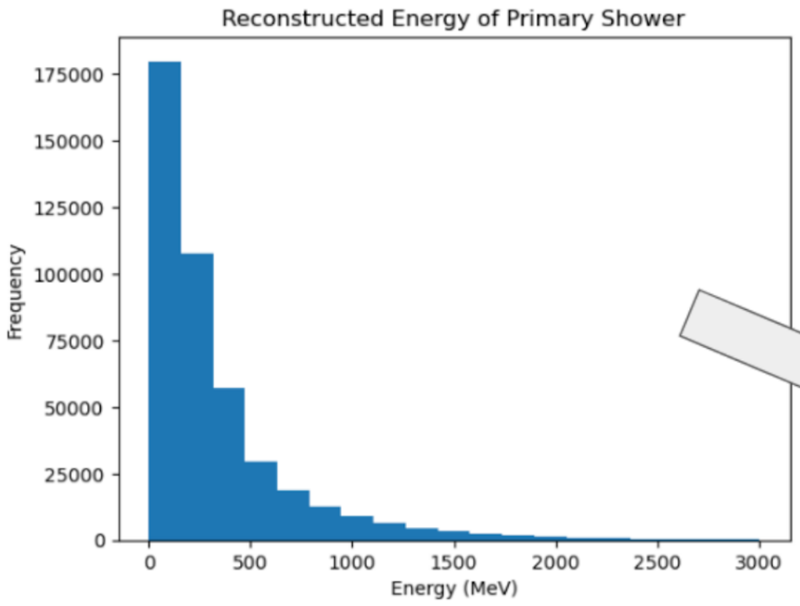
Log Feature Engineering

- A BDT makes successive cuts on data and assigns a weight to each possibility, based on how well that cut partitions the classes.
- XGBoost histograms the data for making the optimal cut
- Heavily skewed data will make XGBoost inefficient.

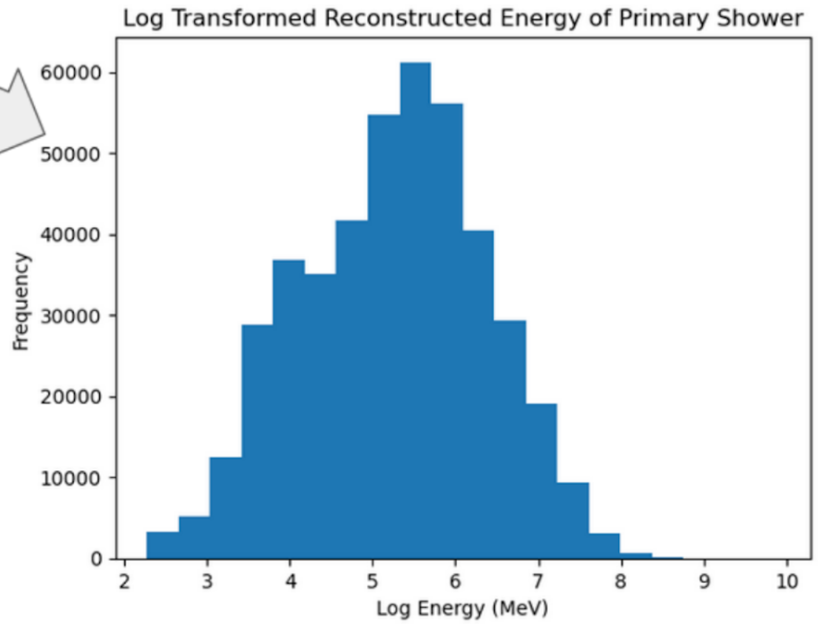


Feature Engineering

- Log transforming heavily skewed data should make it more gaussian and easier for XGBoost to make cuts on.
- Decided which ones to log transform by looking at the first 10 bins out of 60
- With 187 initial variables, the log process added 49

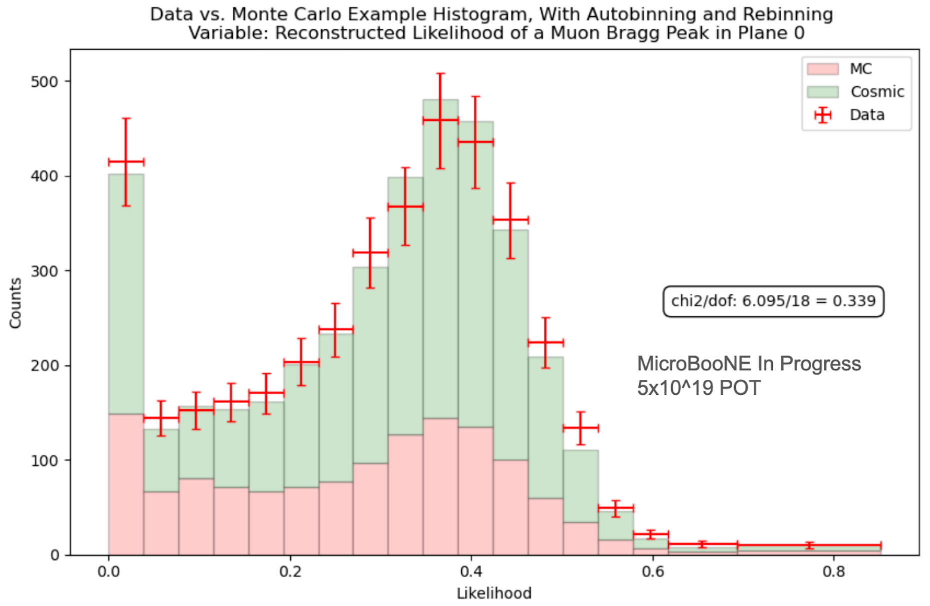


MicroBooNE Simulation in Progress



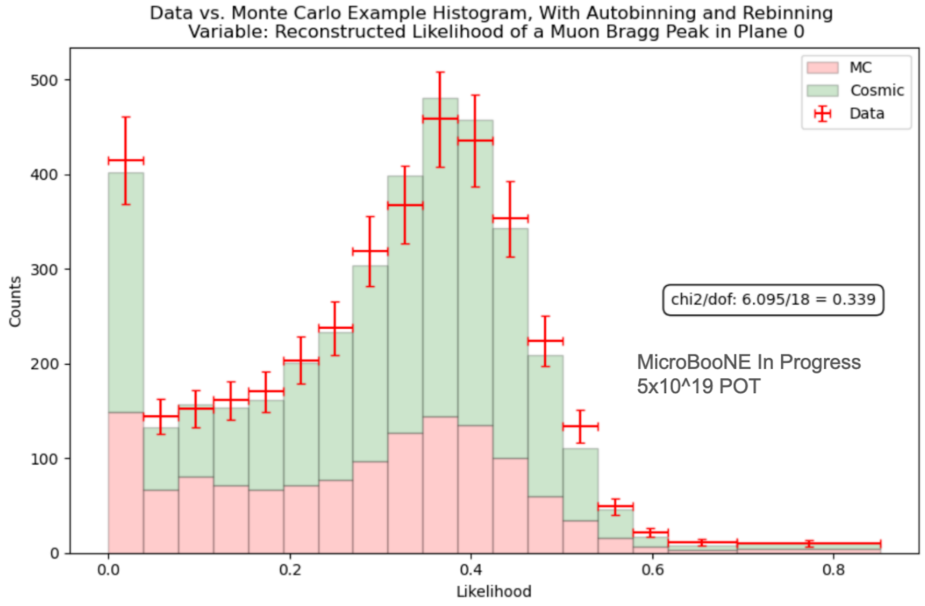
Data Monte Carlo Selection

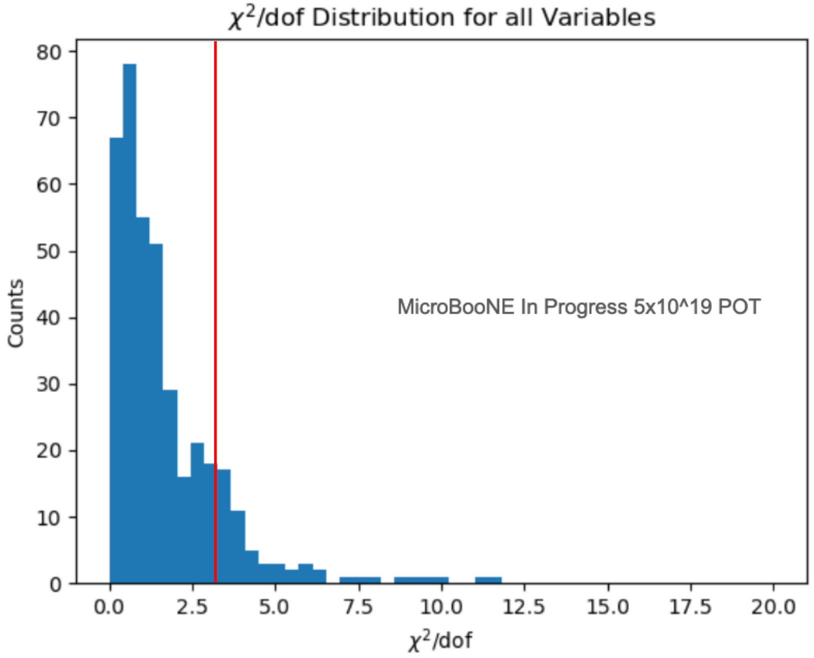
- We have ~10,000 events of real data recorded from MicroBooNE.
- We want to keep variables that are modeled well by the Monte Carlo.
- We created two histograms for each variable, data and Monte Carlo.
- Used Poisson statistical uncertainty and estimated 10% systematic uncertainty as the uncertainty on data, then calculated chi squared for each variable.



Data Monte Carlo Selection

- Variables with too high of a chi squared/dof (>3) were discarded
- In total, 65 were discarded

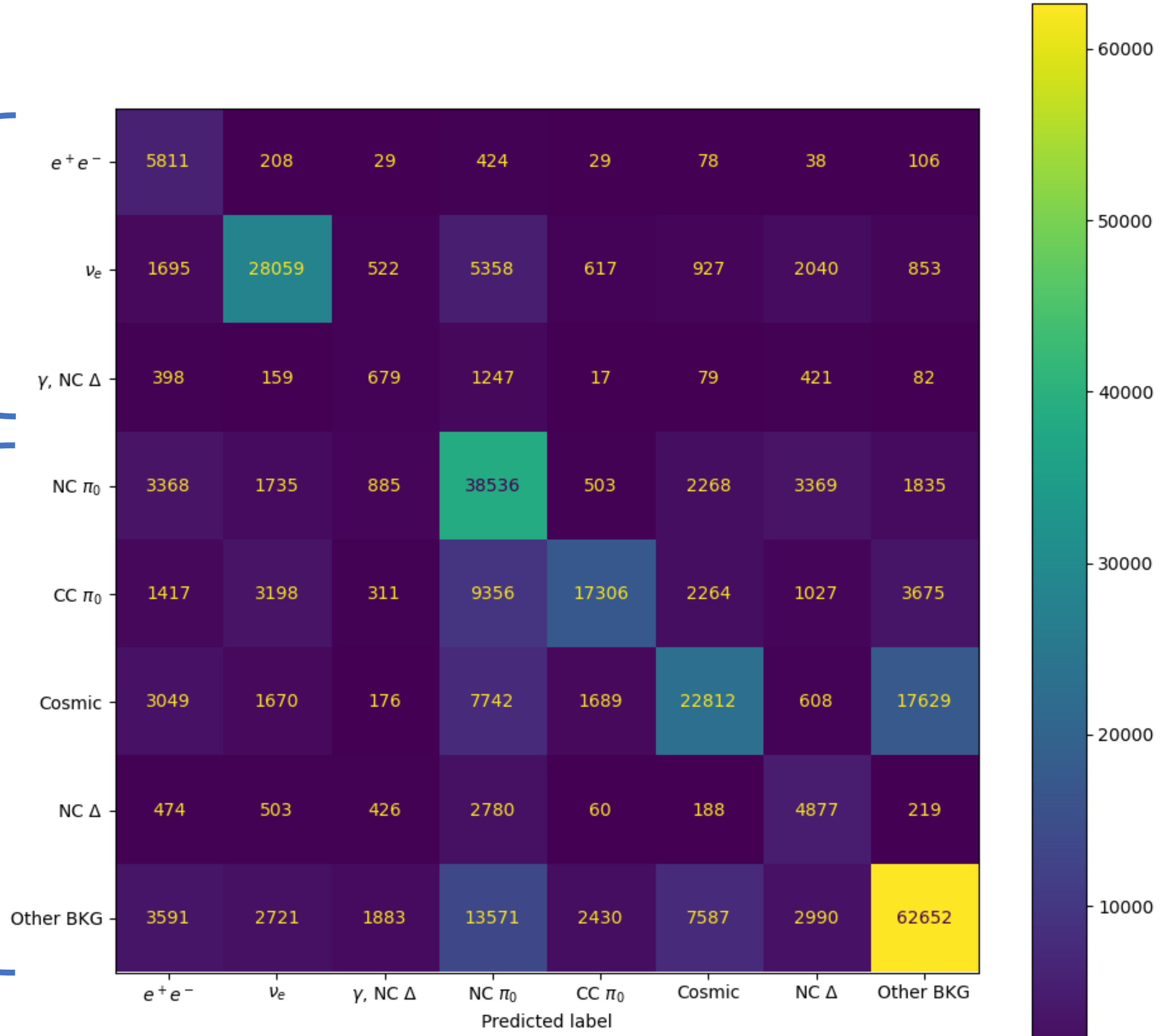




BDT Results

- The current BDT has 8 classes
- 3 Signal: e+e-, ν_e , γ from NC Δ

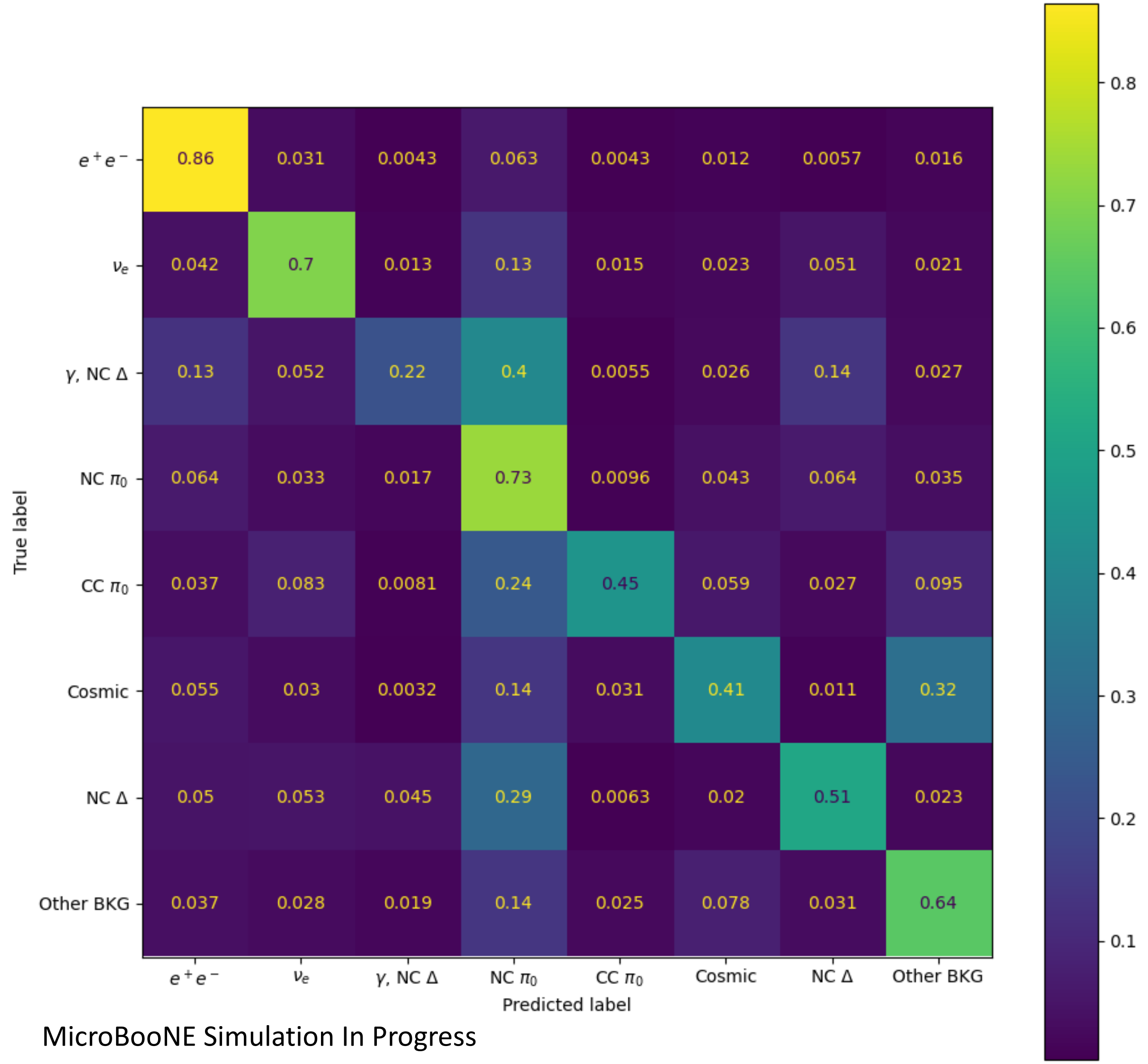
- 5 Backgrounds: NC π^0, CC π^0, Cosmic, NC Δ, and Other



MicroBooNE Simulation In Progress

BDT Results

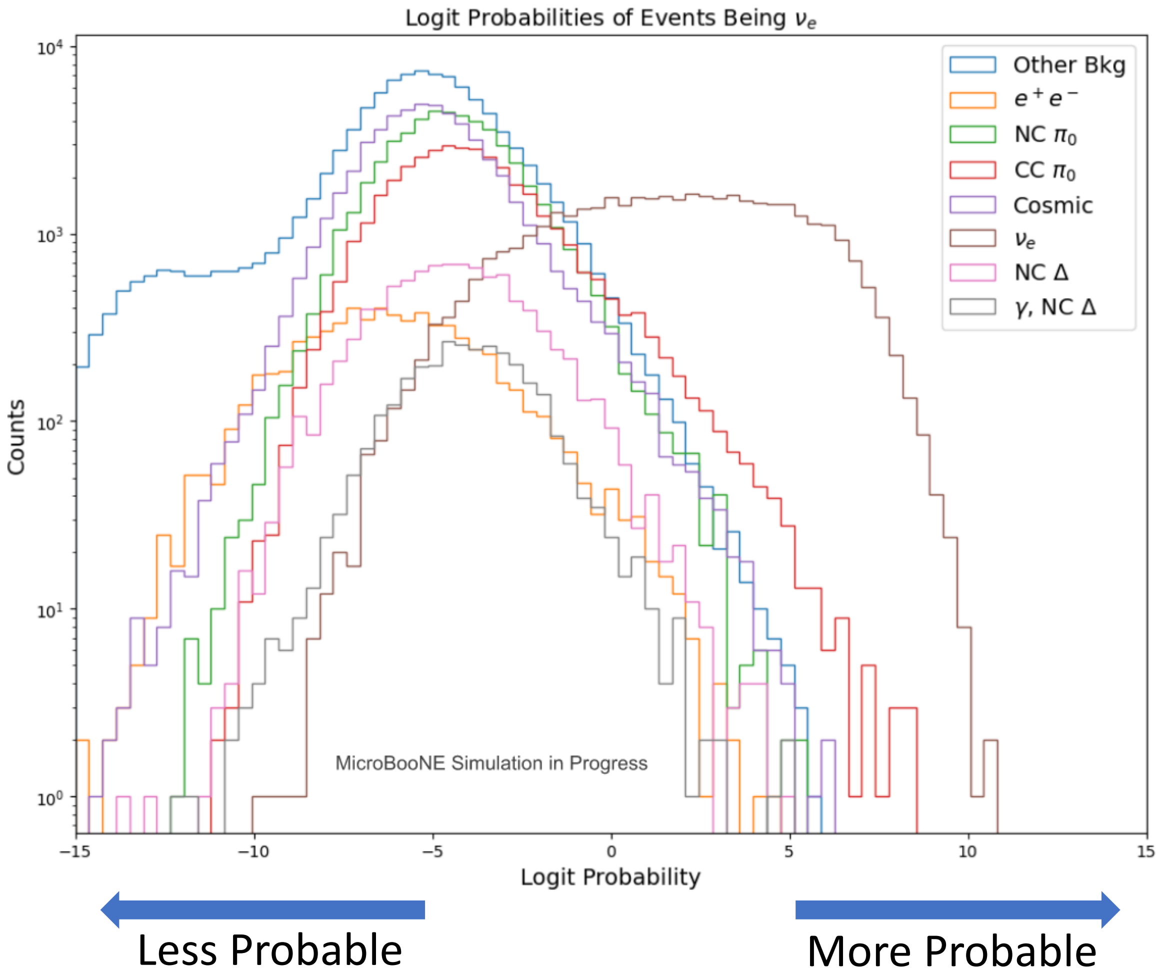
- Same results, but row normalized:
- Each row gives the average probability that a true event is correctly sorted.

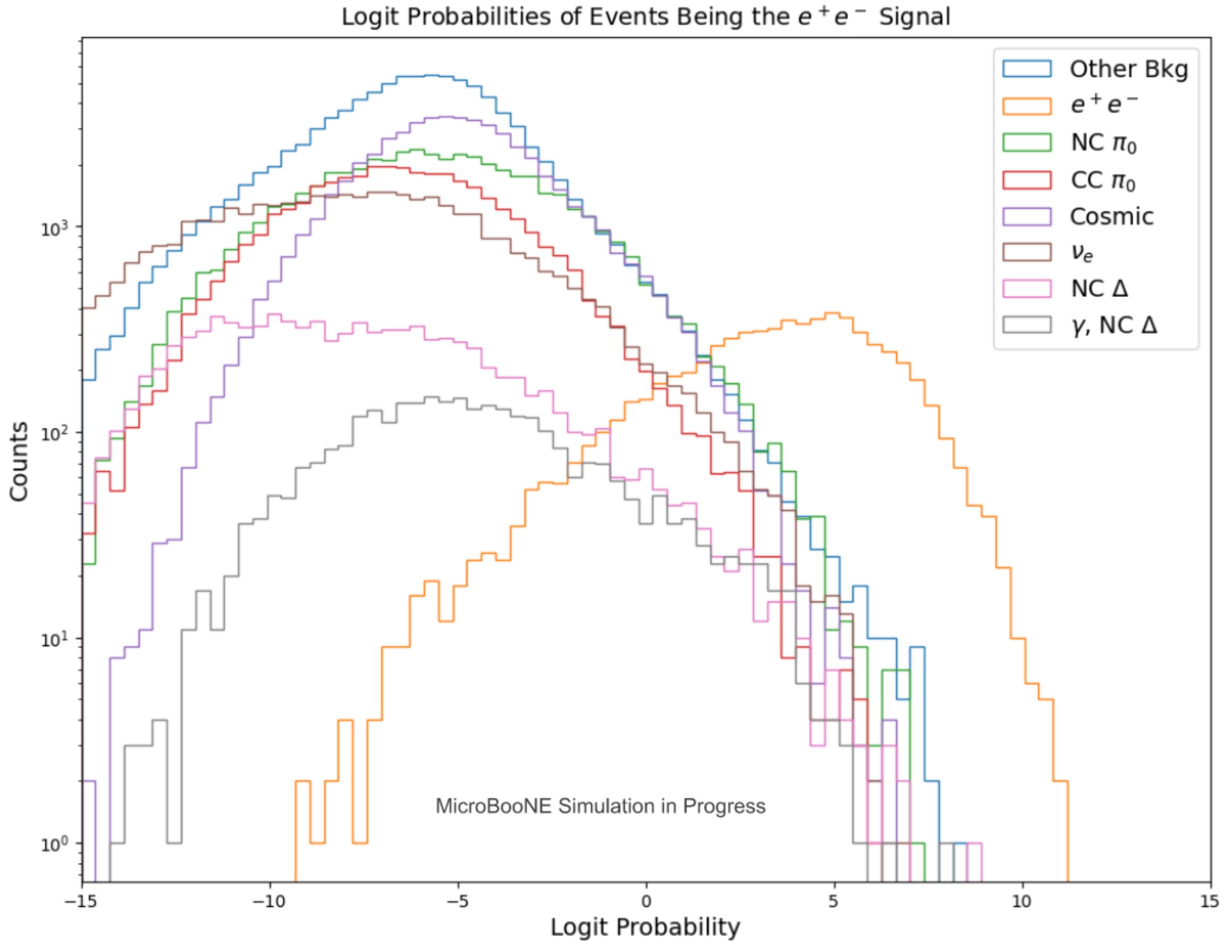


MicroBooNE Simulation In Progress

BDT Results

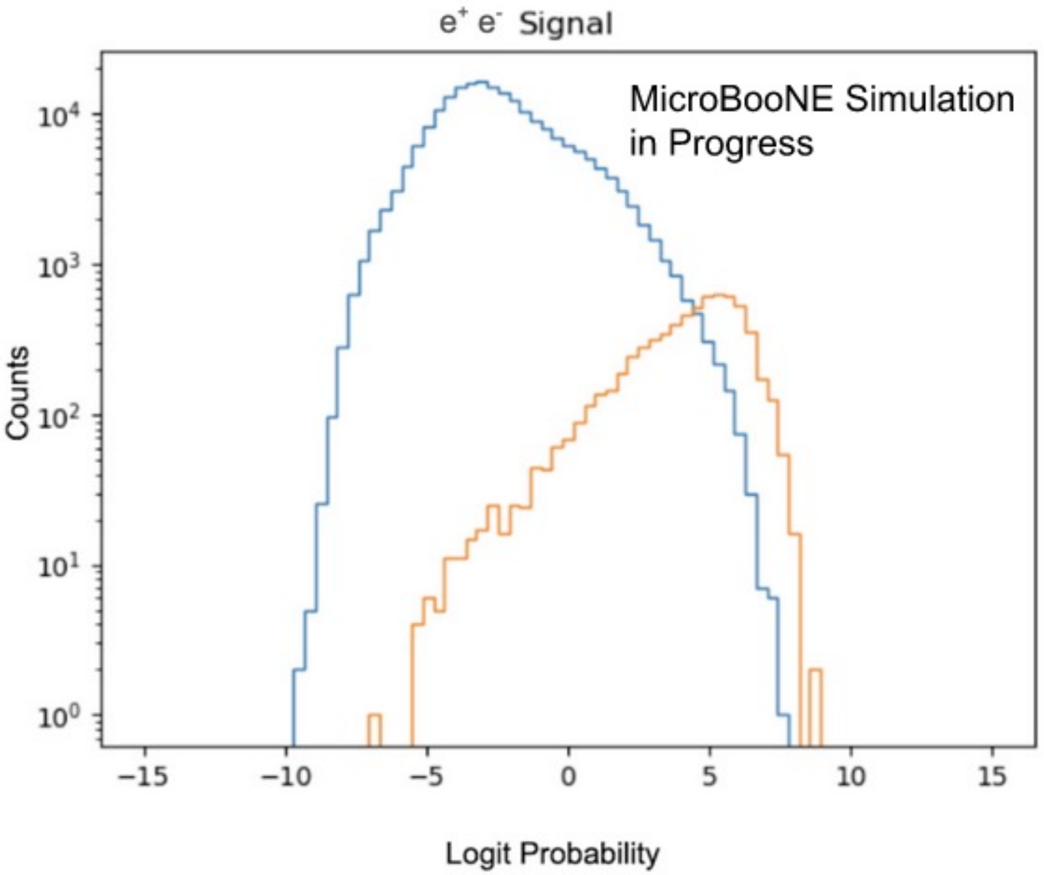
- Below are logit transformed probability outputs of the BDT.
- The type being predicted is the title, and the legend shows the true labels



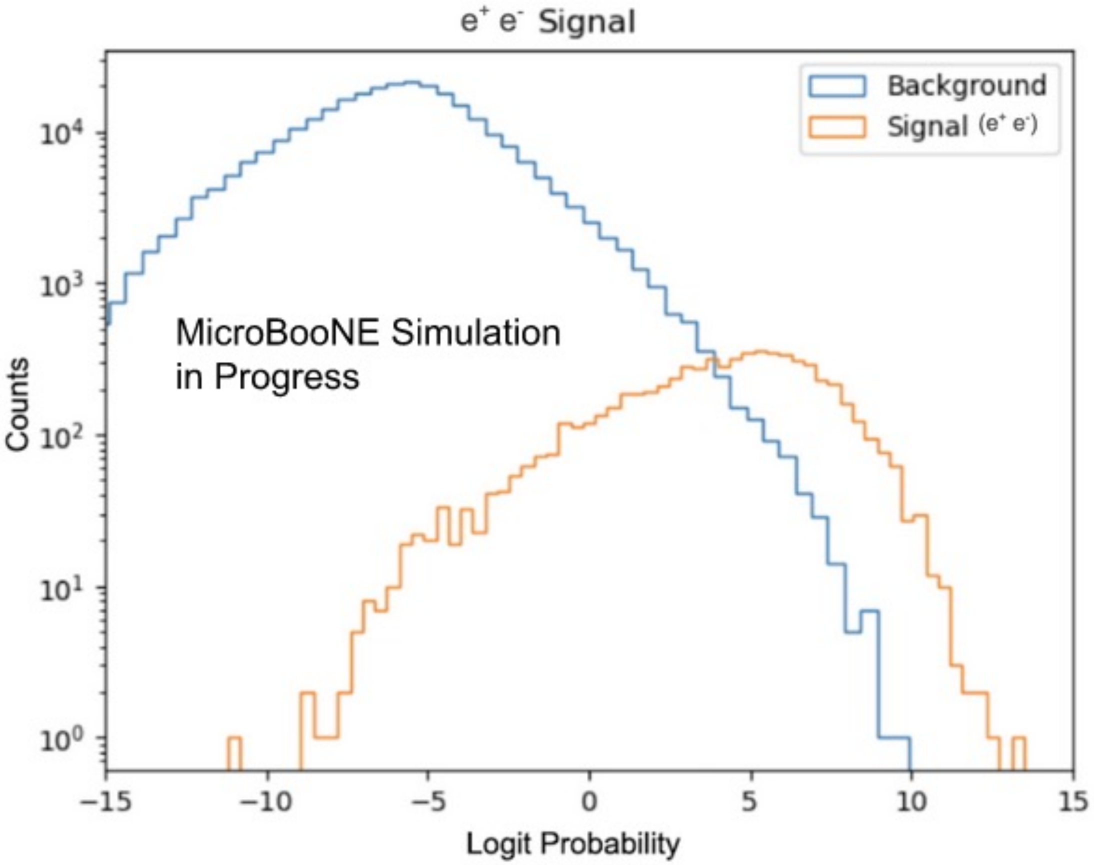


BDT Results

- Signal logit probability distributions of the 1st generation analysis vs the new analysis.
- Signal peaks at about the same probability, but background rejection is better (not right skewed)



1st generation BDTs



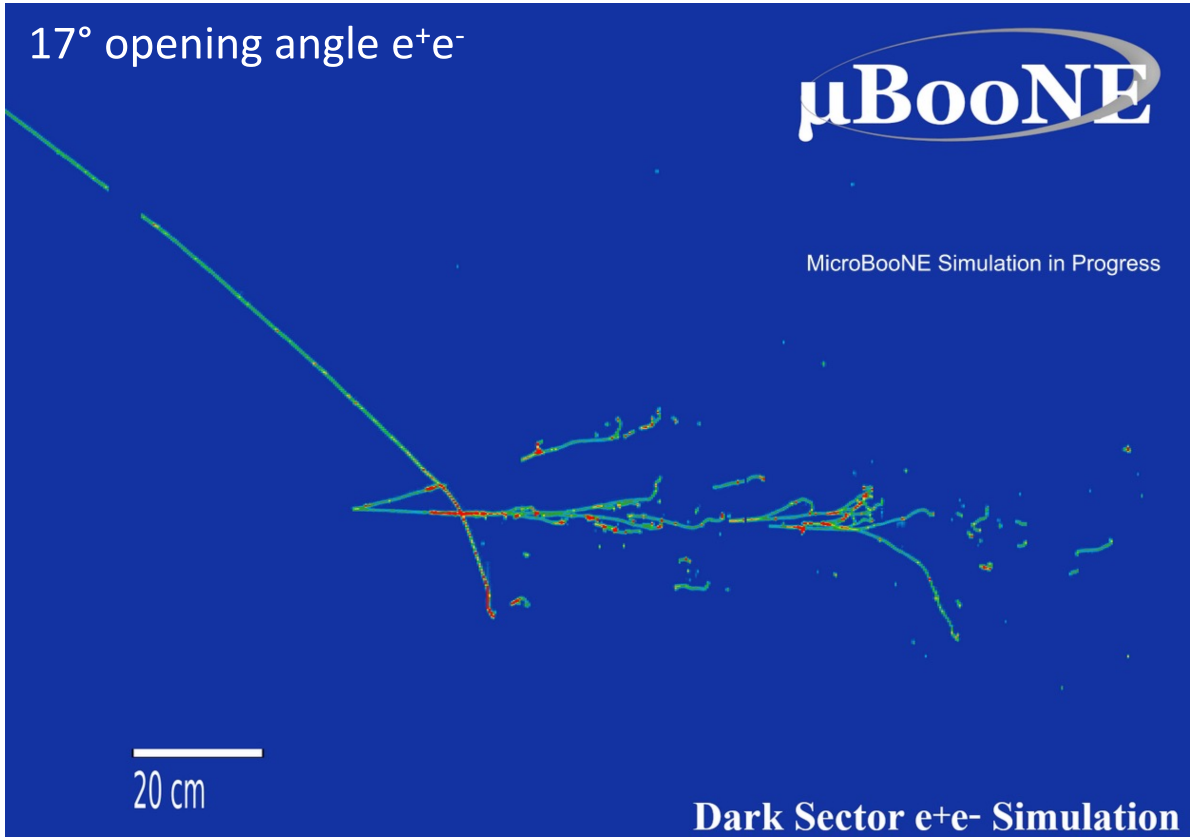
New BDT

Conclusion and Next Steps

- I wrote an improved BDT framework for selecting signals and backgrounds
- Created variable engineering/selection tools
- New BDT outperforms old BDTs

Next:

- Generating more Monte Carlo Data to train on
- Validating data from MicroBooNE's entire dataset, and using the BDT on them
- Finalizing cut choices and hyperparameters
- Integrating with current analyses



Acknowledgements

Fermilab
U.S. DEPARTMENT OF ENERGY | Office of Science
NSF
FNSNF SWISS NATIONAL SCIENCE FOUNDATION
Science & Technology Facilities Council
THE ROYAL SOCIETY
UKRI UK Research and Innovation
Los Alamos NATIONAL LABORATORY

Backup Slides

Further Information

Package versions:

XGBoost - 1.7.4

Scikit-learn - 1.3.0

Scipy - 1.11.3

Numpy - 1.24.4

Pandas - 1.5.3

Hyperopt - 0.2.7

Optuna - 3.4.0

GPU Acceleration:

- XGBoost comes with basic single GPU support, by passing the argument: tree_method = “gpu_hist”
- It is roughly an order of magnitude faster than single thread CPU, but about the same as multi-threaded CPU
- For multi-GPU support, packages like dask, cudf, and cupy will likely be necessary. Unclear how much it would improve performance.

Monte Carlo Data:

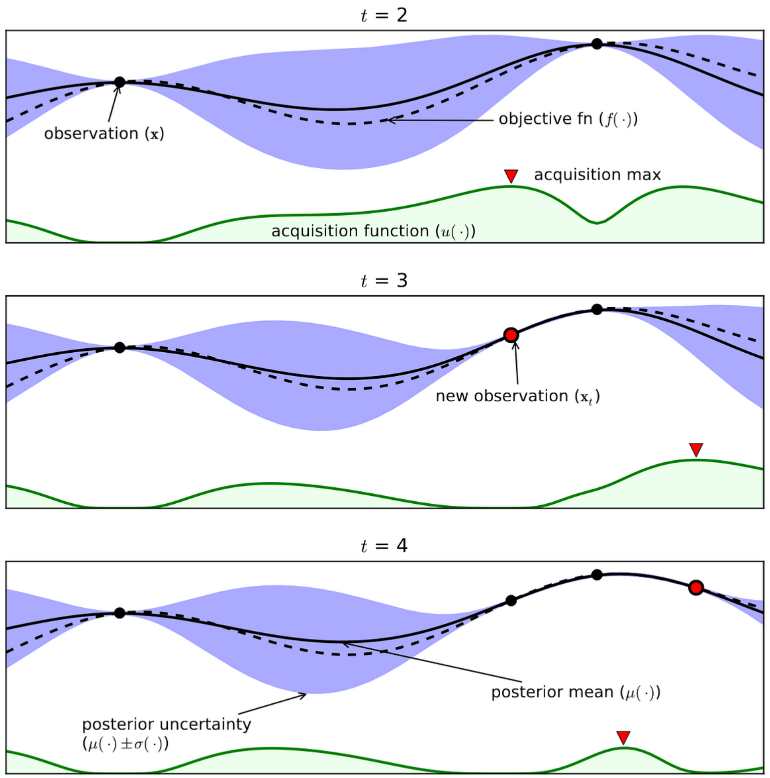
- e+e- signal events were generated with DarkNews. Backgrounds were generated with GENIE
- Events were propagated with Geant4, put through DetSim for detector simulation, and reconstructed with Pandora
- Test and train data was pre-split to avoid the same cosmic shower overlays appearing in both sets

Further Information

Hyperparameter Values:
eta (learning rate) = 0.1932530144602815,
max_depth = 9,
n_estimators = 800,
num_class = 6,
gamma = 0.0016439691781 8055,
reg_lambda = 52.08749037888948,
alpha = 0.0364146254675091,
colsample_bylevel = 0.9909644387051986,
colsample_bynode = 0.7183508385391129,
colsample_bytree = 0.986792749280876,
subsample = 0.6293572669454952

Hyperparameter Tuning

- Hyperparameters of interest:
  Learning rate, max_depth, n_estimators, min_child_weight, alpha, gamma, lambda, subsample rates
- I tested several options for hyperparameter tuning:
  - Grid Search
  - Random Search
  - Bayesian Optimization
- All used K-fold cross validation
  - Splitting training set into k equal sets, then training and averaging over k models, each leaving one of the sets out to use as the test set
- Each option best suited to a certain parameter space sizes and training times



1012.2599

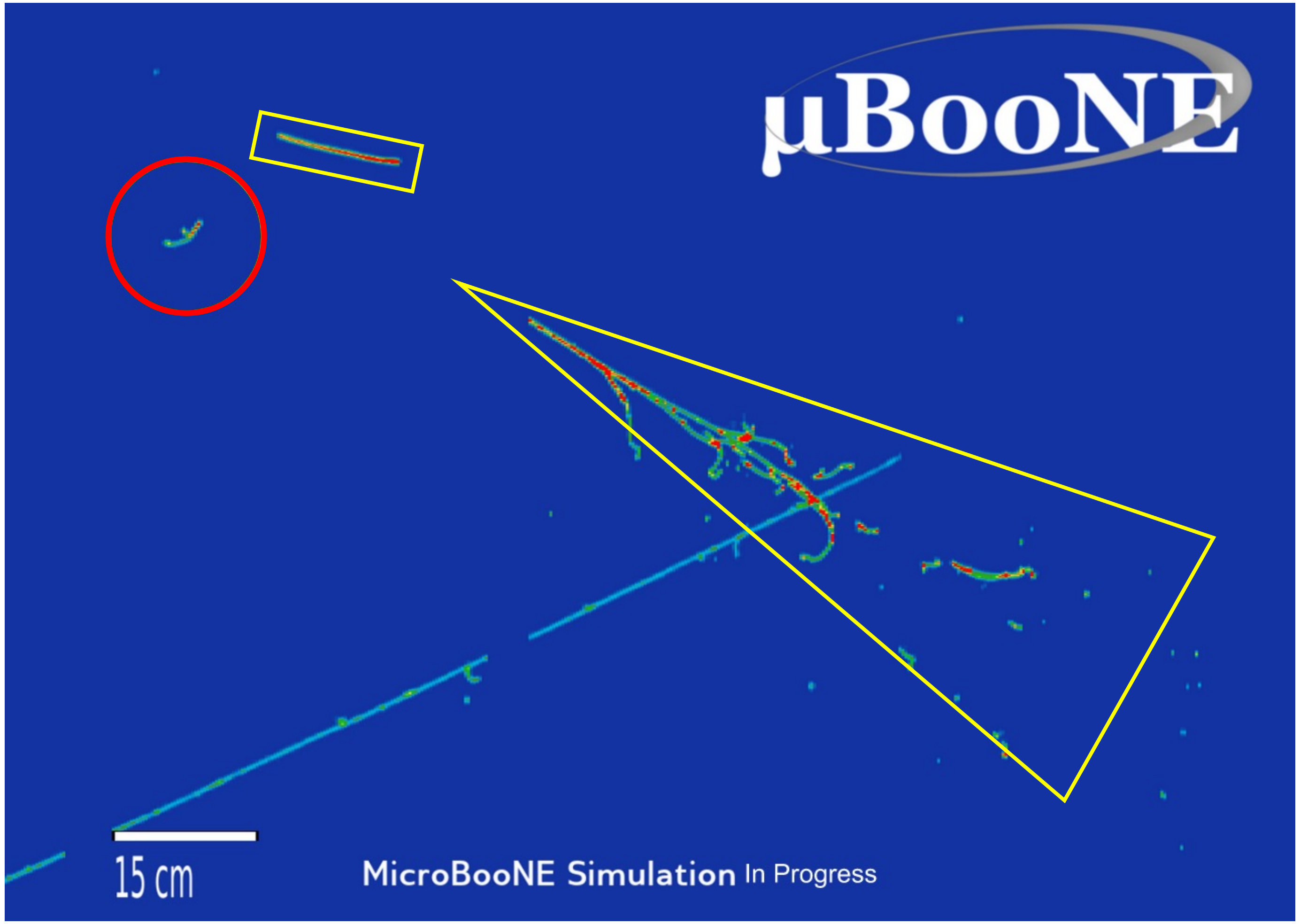
Correlation Matrix Selection

- Highly correlated variables carry the same information and slow the training
- I used the correlation matrix in two ways:
  - Finding highly correlated variables.
  - Removing variables with 0 variance (PCA)
- 45 more variables were removed with this selection.


Training and Test Metrics Over Trees
Log Loss
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
Error Rate
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
AUC
1.0
0.8
0.6
0.4
0.2
0.0
0
200
400
600
800
1000
1200
1400
Number of Trees
MicroBooNE Simulation in Progress
Train Log Loss
Test Log Loss
Train Error Rate
Test Error Rate
Train AUC
Test AUC

Cluster Variables

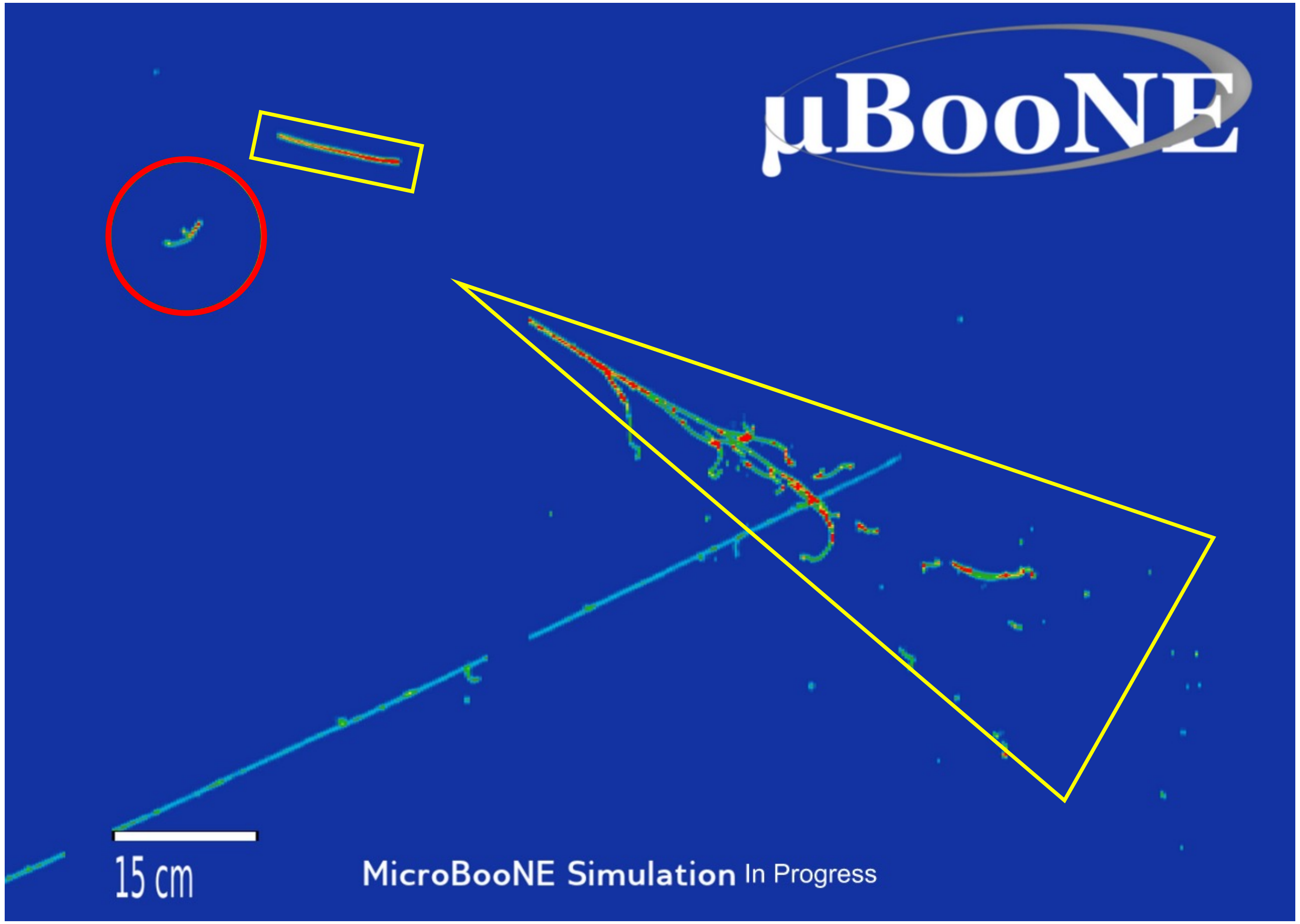
- These variables were developed to search for small showers or tracks that were missed by the initial reconstruction.
  - There are two types: second shower and track stub.



A simulated NC π^0 event that was reconstructed as a 1 track plus 1 shower interaction. The secondary photon from the π^0 is very low in energy, 19 MeV, and can be seen circled to the left of the track.

Cluster Variables

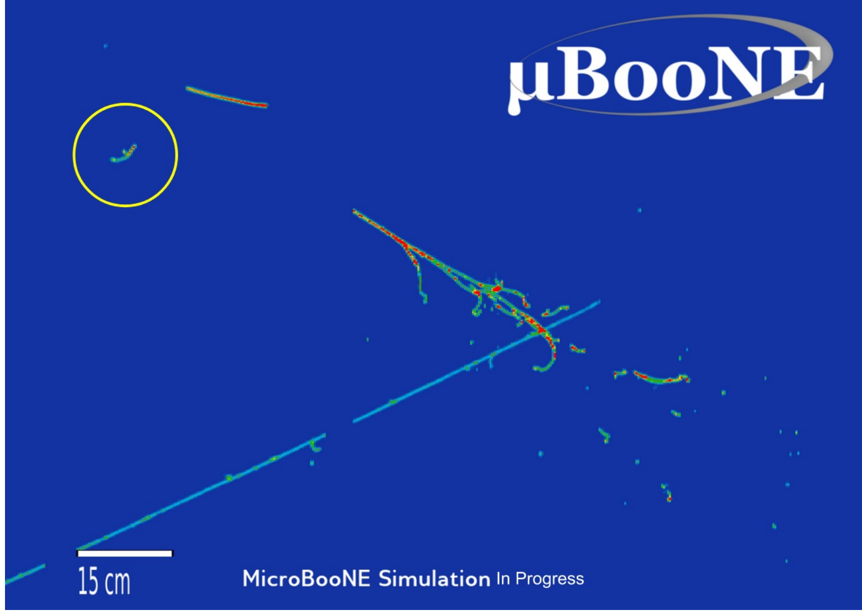
- In each event they are formatted in vectors of data, each entry corresponding to a single detector hit.
- Boosted decision trees do not deal well with vectors of data, especially with varying length (ten to hundreds).



A simulated NC π^0 event that was reconstructed as a 1 track plus 1 shower interaction. The secondary photon from the π^0 is very low in energy, 19 MeV, and can be seen circled to the left of the track.

Cluster Variables

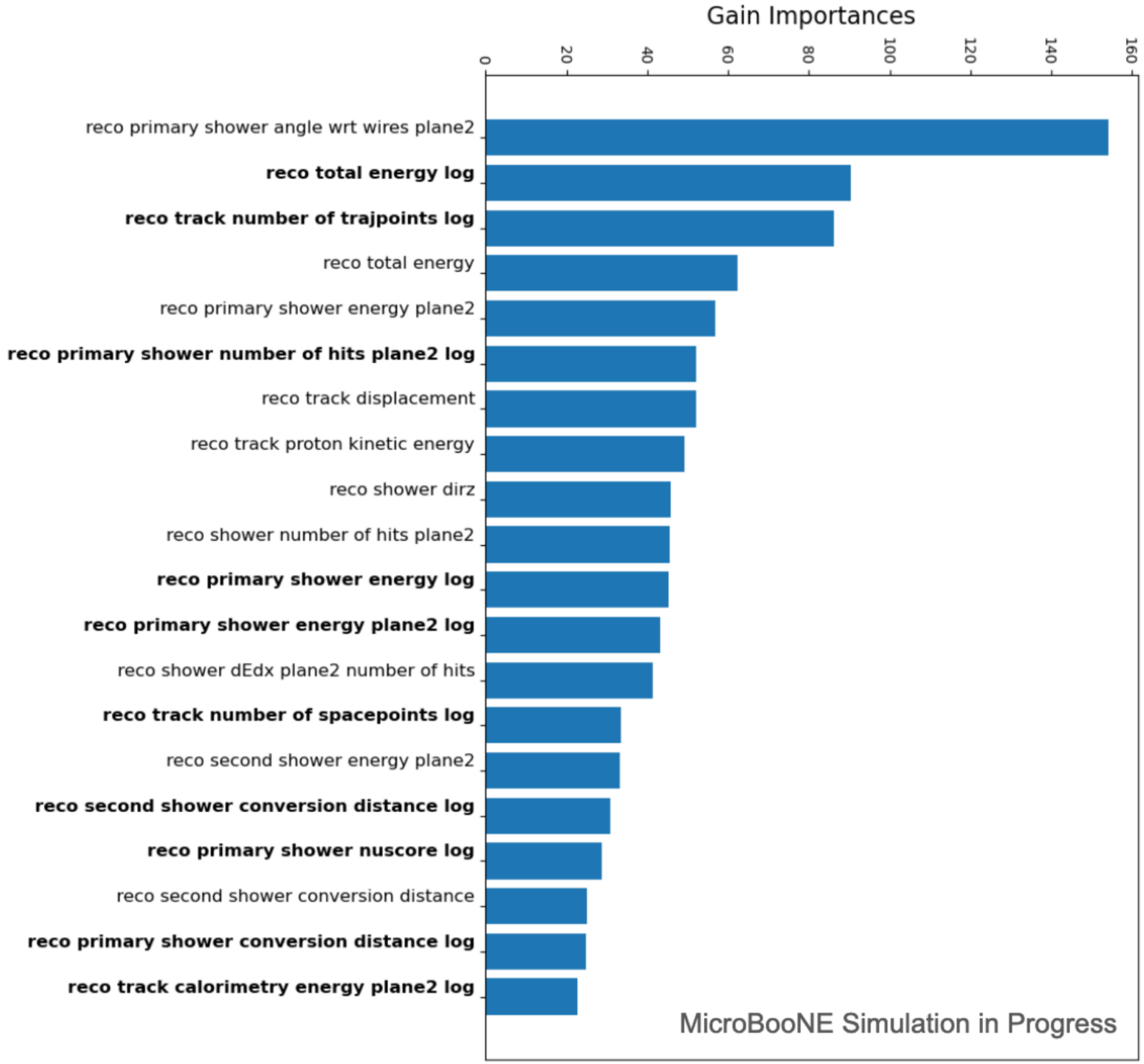
- I needed to condense the information from cluster variables into a reasonable number of variables.
- Solution: grouping the closest cluster events and choosing the 4 most energetic groups
- This way, for each cluster variable, 4 new variables are added.
- In total, 215 new variables are added



A simulated NC π^0 event that was reconstructed as a 1 track plus 1 shower interaction. The secondary photon from the π^0 is very low in energy, 19 MeV, and can be seen circled to the left of the track.

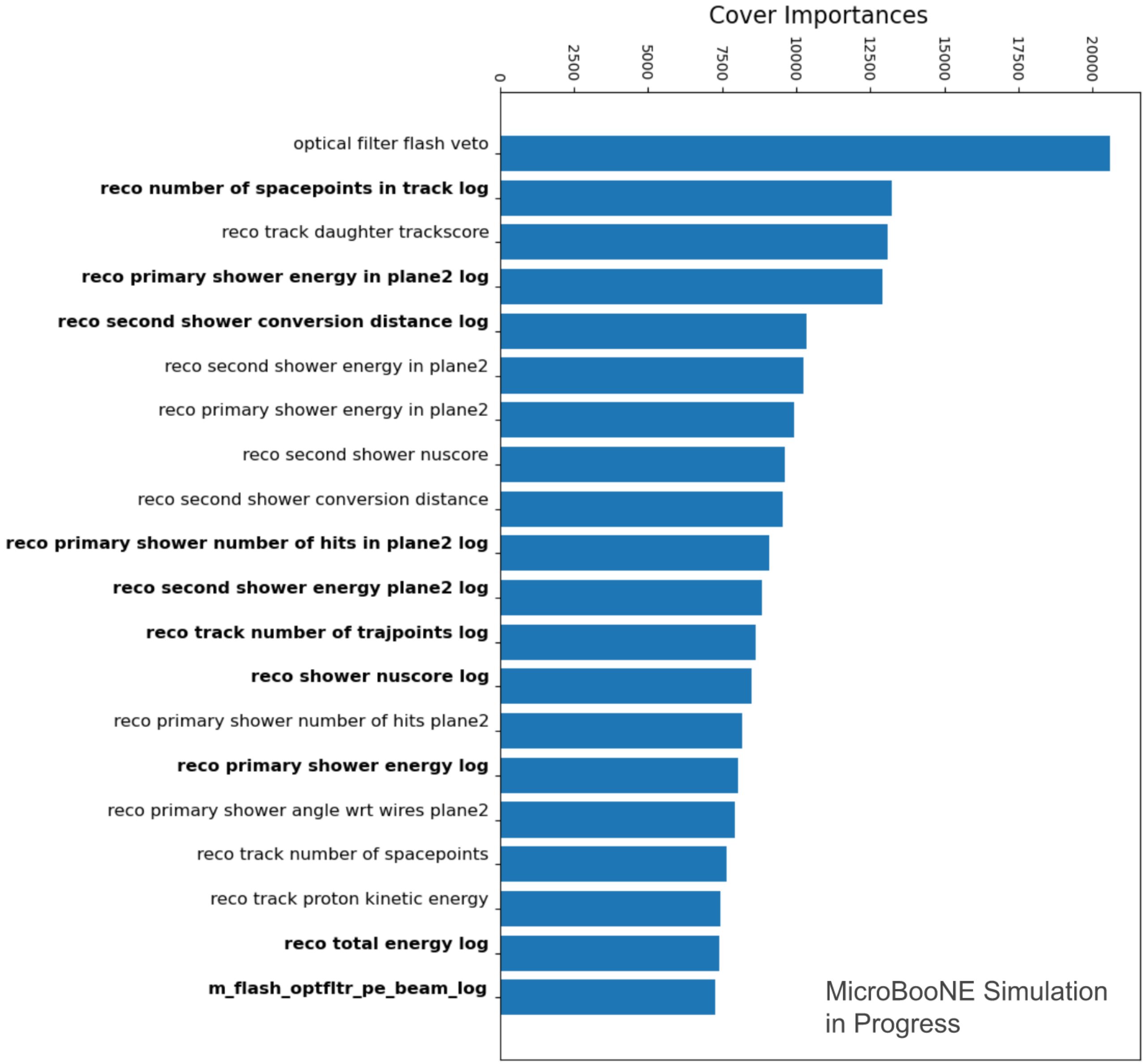
BDT Results

- Feature importances tell us which variables are being used by the model, and how useful they are.
- Gain = Average increase in similarity score.



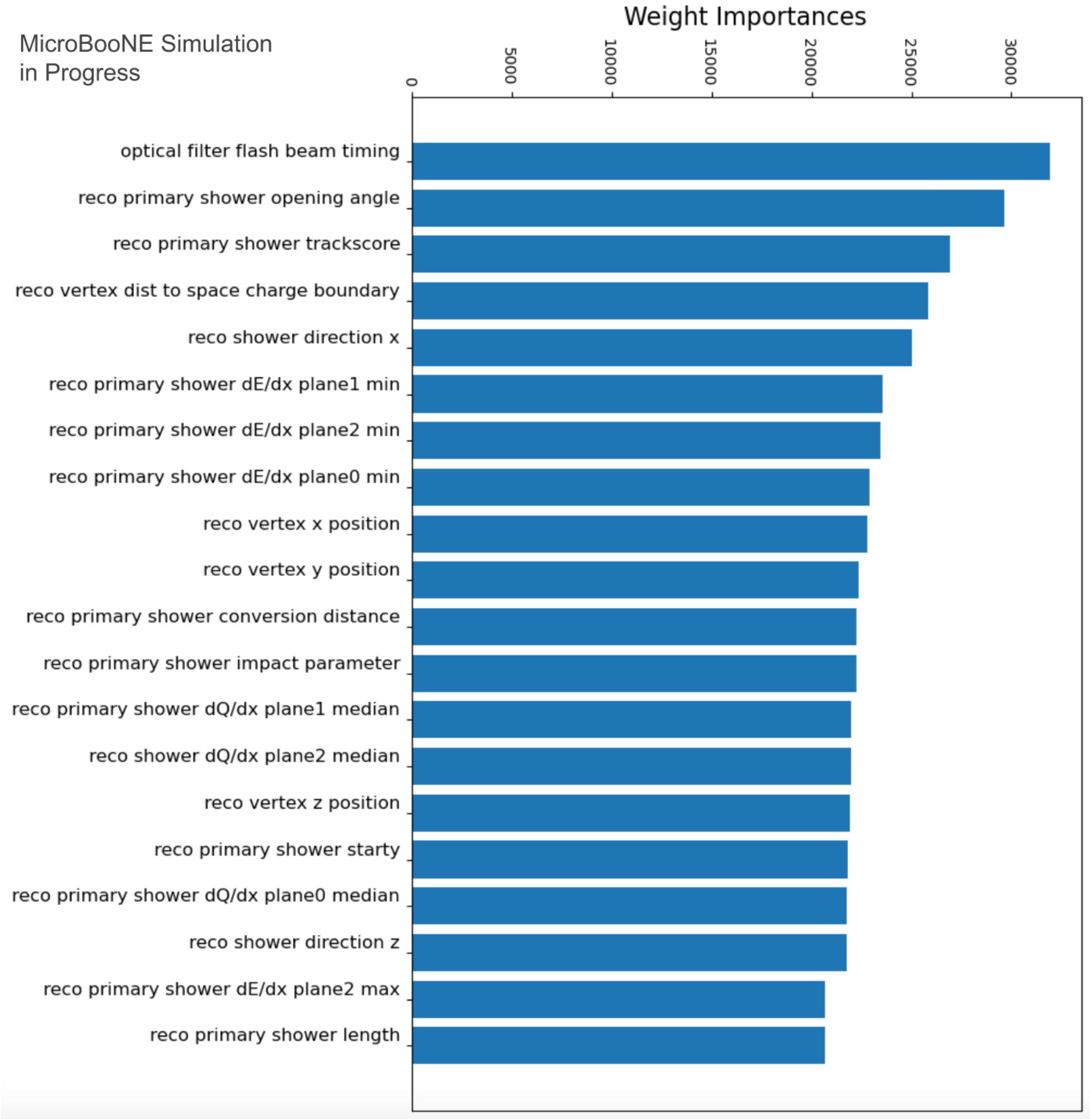
BDT Results

- Feature importances tell us which variables are being used by the model, and how useful they are.
- Gain = Average increase in similarity score.
- Cover = Number of times a variable is used to make a branch.



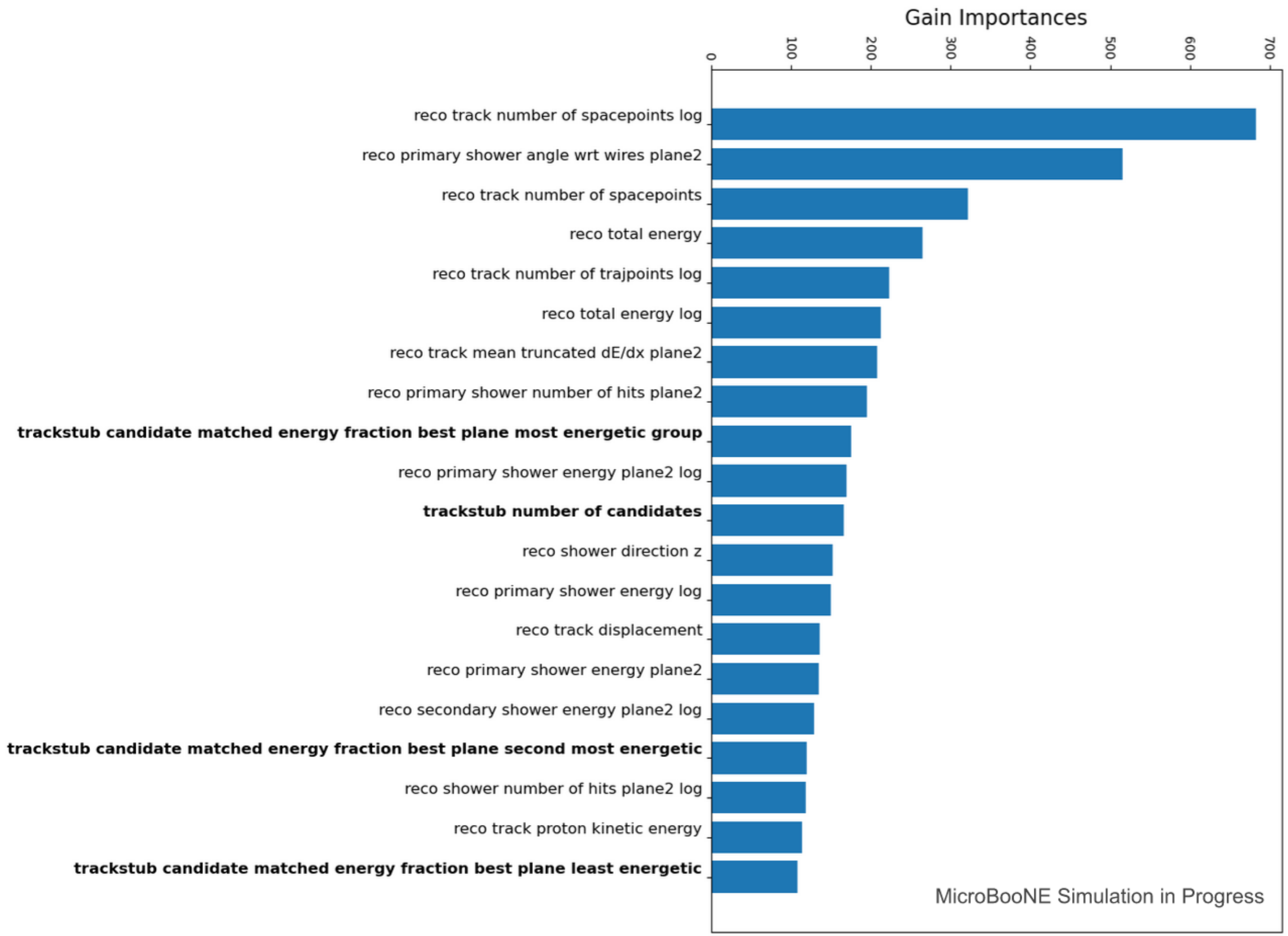
BDT Results

- Feature importances tell us which variables are being used by the model, and how useful they are.
- Gain = Average increase in similarity score.
- Cover = Number of times a variable is used to make a branch.
- Weight = number of data points which a variable affects.



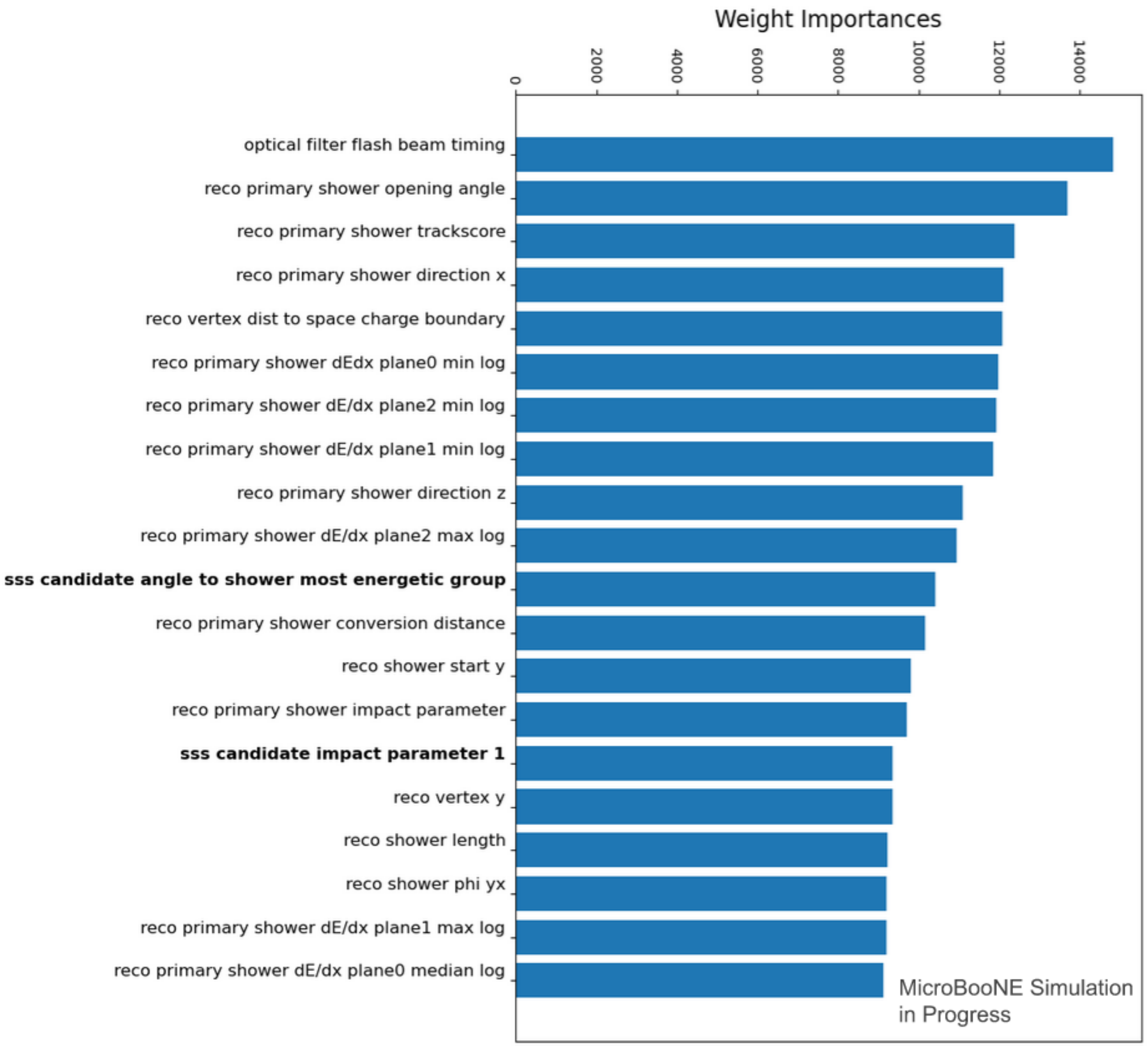
BDT Results - Feature Importances

- Feature importances tell us which variables are being used by the model, and how useful they are.
- Gain = Average increase in similarity score.
- Weight = number of data points which a variable affects.



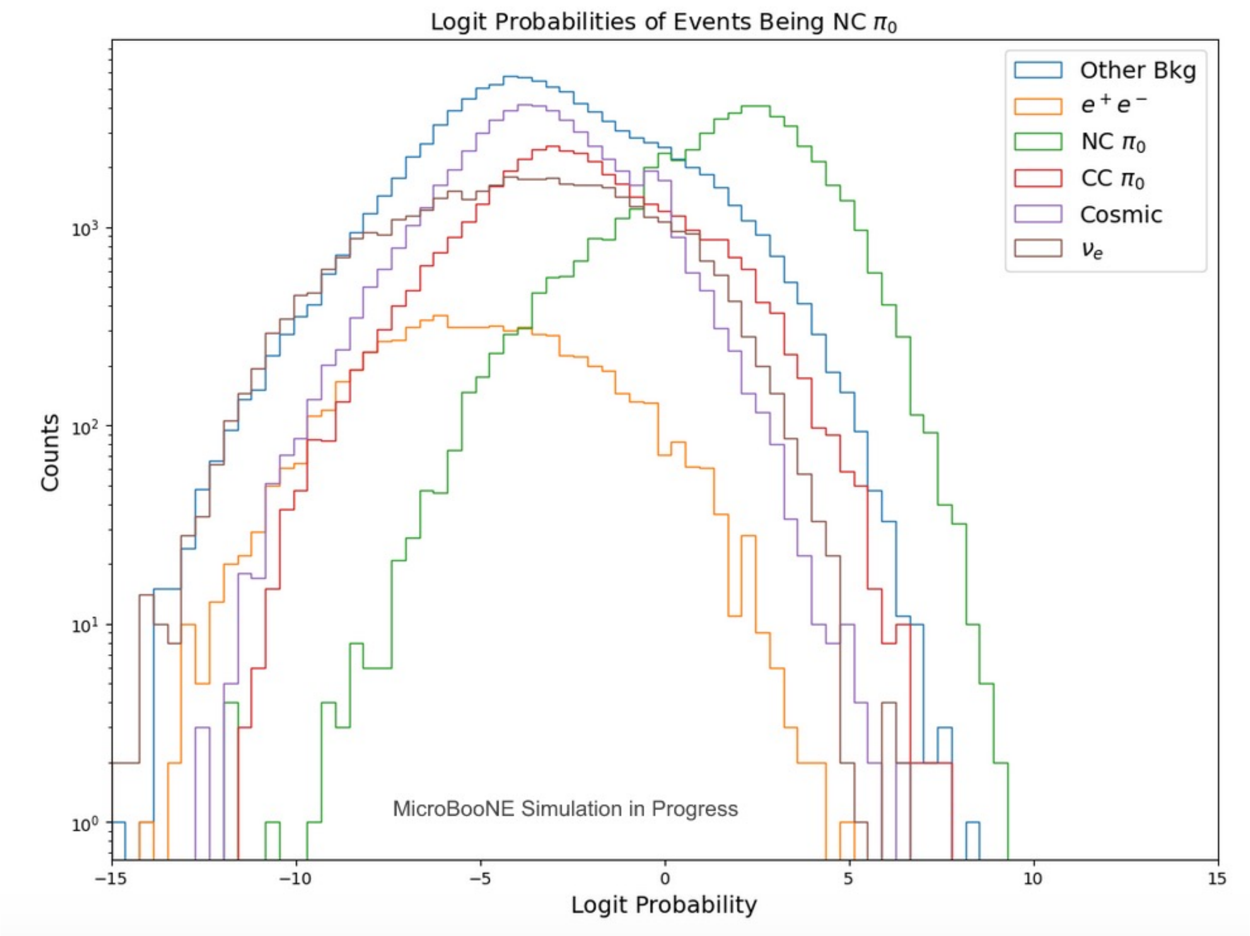
BDT Results - Feature Importances

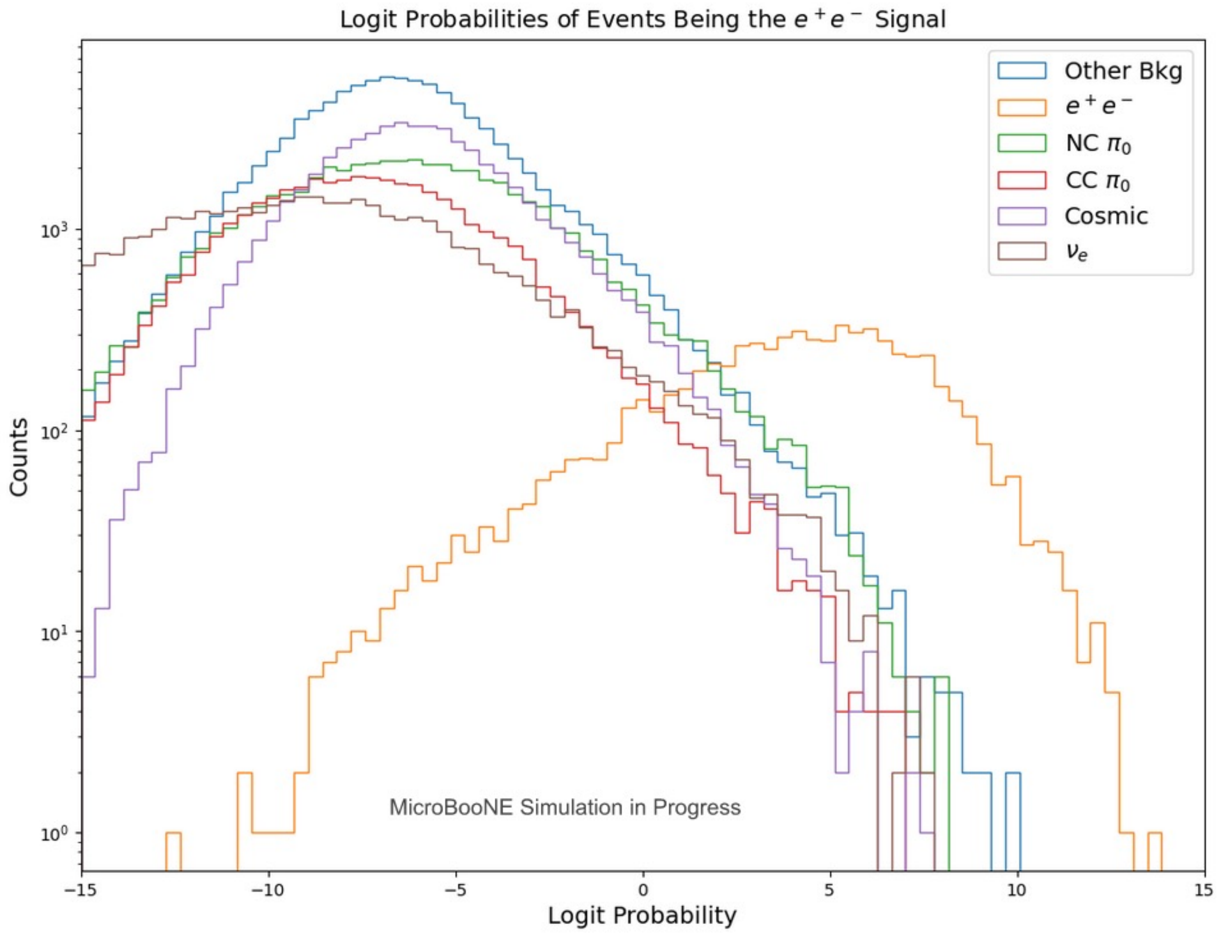
- Cluster variables found in the top 20.
- Quite a few log transformed variables too.



BDT Results

- Below are logit transformed probability outputs of the BDT.
- The type being predicted is the title, and the legend shows the true labels





BDT Results

- To the right are POT weighted confusion matrices and metrics, with the backgrounds summed, for the old BDTs and for this talk's BDTs
- Precision aka Purity:
  - True signal predicted as signal / (All predictions of signal)
- Recall aka Efficiency:
  - True signal predicted as signal / (All true signals)
- F_1 = 2*precision*recall/(precision+recall)

Old:
Precision: 0.3178
Recall: 0.3837
precision*recall: 0.1220
F_1 score: 0.3477

Old	Predicted Background	Predicted Signal
True Background	0.998511	0.000673
True Signal	0.000504	0.000314

New	Predicted Background	Predicted Signal
True Background	0.998891	0.000435
True Signal	0.000345	0.000329

New:
Precision: 0.4303
Recall: 0.4877
precision*recall: 0.2099
F_1 score: 0.4572