DEEP UNDERGROUND NEUTRINO EXPERIMENT

WARWICK
THE UNIVERSITY OF WARWICK

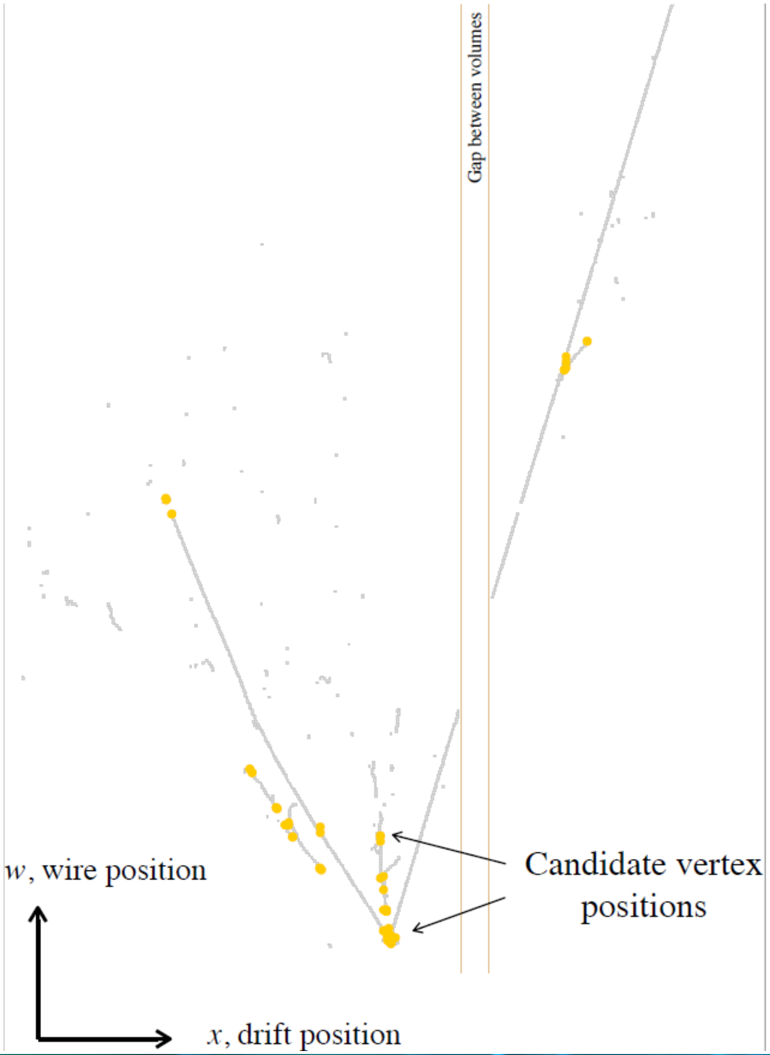
DUNE FD Sim/Reco Meeting
6th of January 2020

Updates on Vertex Reconstruction with Deep Learning in Pandora

J. Ahmed. Second Year PhD Student.
Working on Pandora reconstruction Focusing on DUNE FD and ProtoDUNE.
Supervisor: Dr John Marshall

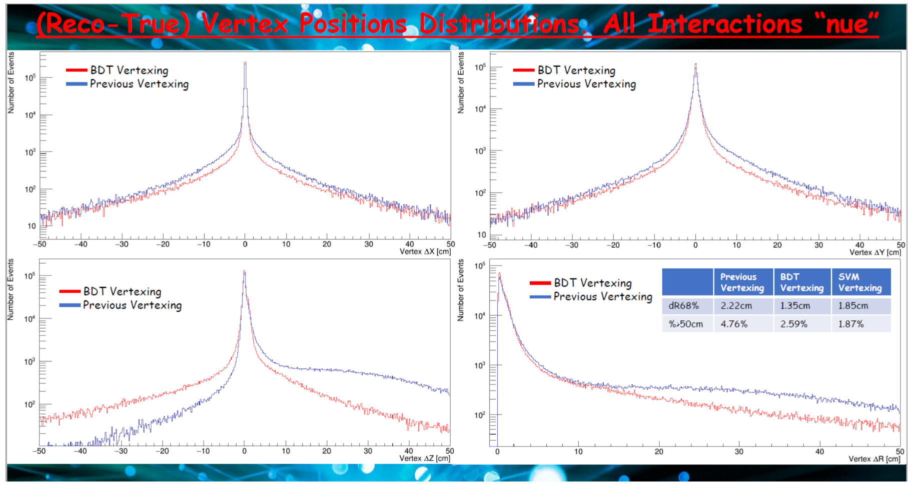
Introduction to Current Vertexing in Pandora

- The vertex is identified in Pandora at the 2D stage of the reconstruction. The process comprises 2 algorithms:

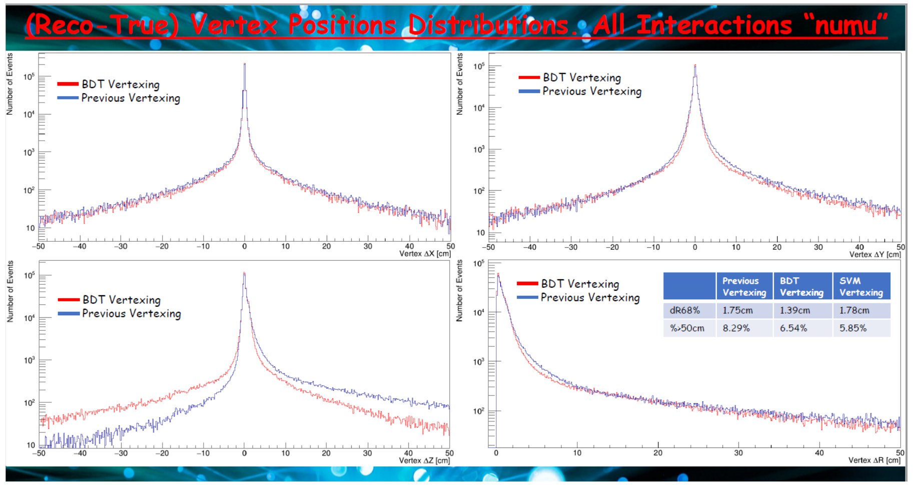


- The first is the **Candidate Vertex Creation Algorithm** which sprinkles the event with lots of 3D candidates.
- The **Vertex Selection Algorithm** tries to find the primary interaction vertex out of these candidates.
- Originally this was the EnergyKickVertexSelectionAlgorithm which worked by assigning each candidate a value for three types of scores.
- The candidate with the highest product of the three scores was chosen as the primary interaction vertex.
- This algorithm was replaced by me earlier this year with a BDT Vertex Selection Algorithm that is now the default for Pandora DUNE FD reconstruction.

BDTVertexSelectionAlgorithm



	Previous Vertexing	BDT Vertexing	SVM Vertexing
dR68%	2.22cm	1.35cm	1.85cm
%>50cm	4.76%	2.59%	1.87%



	Previous Vertexing	BDT Vertexing	SVM Vertexing
dR68%	1.75cm	1.39cm	1.78cm
%>50cm	8.29%	6.54%	5.85%

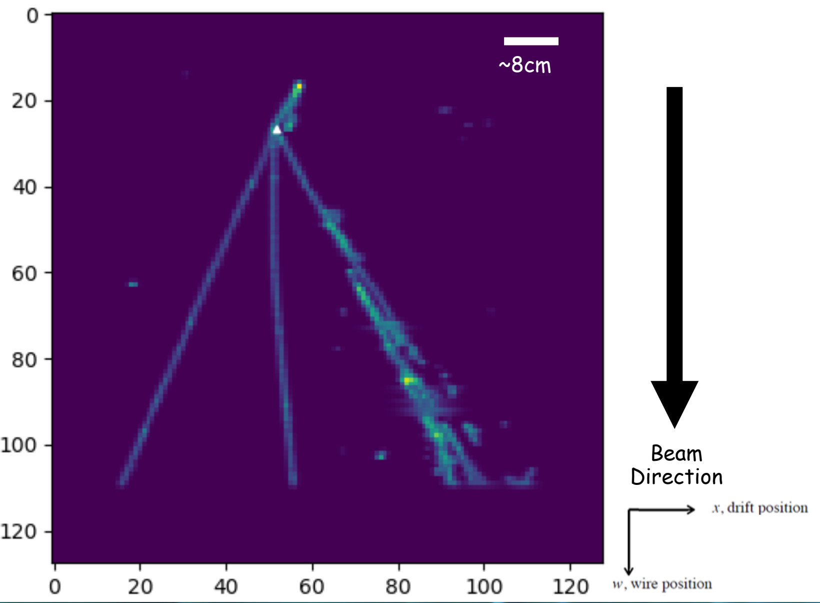
- The problem is split into a **region-finding algorithm, which evaluates different regions in the event**, and a **vertex-finding algorithm, which chooses the most appropriate vertex in a given region.**

- There are two types of BDT variables, Vertex Features (i.e. related to vertex position), plus some Event-Based Features (i.e. related to the hit distribution only).

- The current main aim is to improve the BDT vertex selection algorithm by incorporating Deep Learning (DL) to make a new score that would go into the BDT.

Convolutional Neural Network (CNN) Approach

- ~50,000 “nue” and ~50,000 “numu” DUNE FD 1x2x6 MCC11 events are used.
- For each event the true 3D Vertex position is obtained and projected into each of the three views.
- A 128 by 128 pixel image is then created for each of the three views.



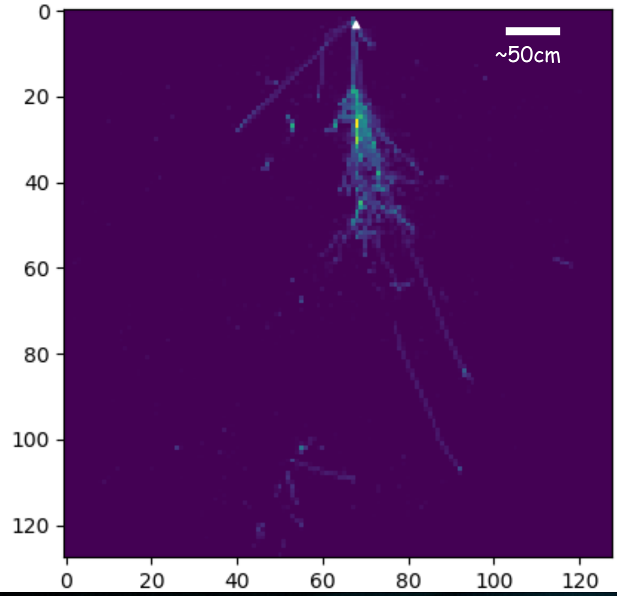
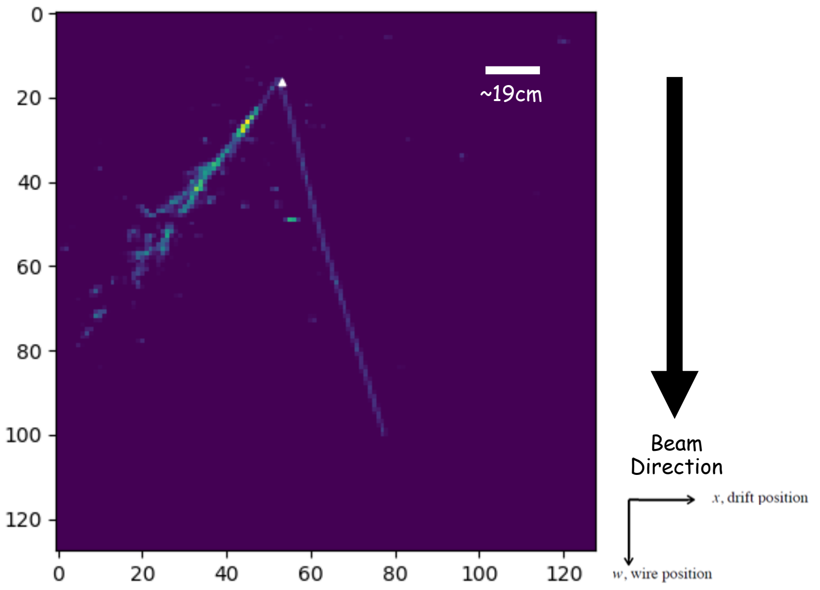


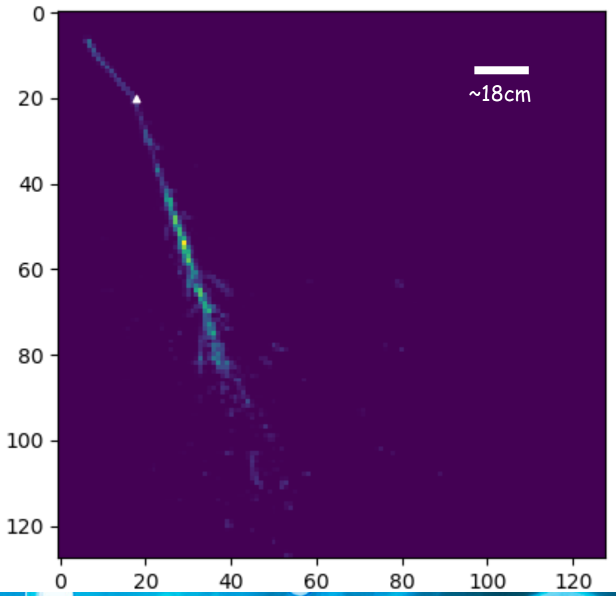
Image Creation Technique

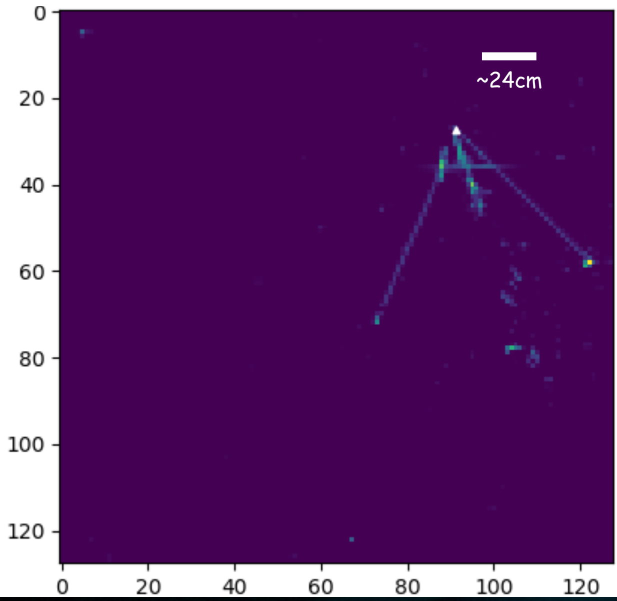
- At the stage vertexing occurs, the reconstructed clusters for each view are used. All clusters with less than 5 hits are removed. The length of the square needed to contain the centers of all hits in the remaining clusters for each view is calculated.

- The image size is then decided to be this square with an extra 10cm in each direction.

- The image is then filled with the hits from all clusters. Examples of how these training images for the W view look like are shown below. The true vertex (white triangle) is marked on the images.



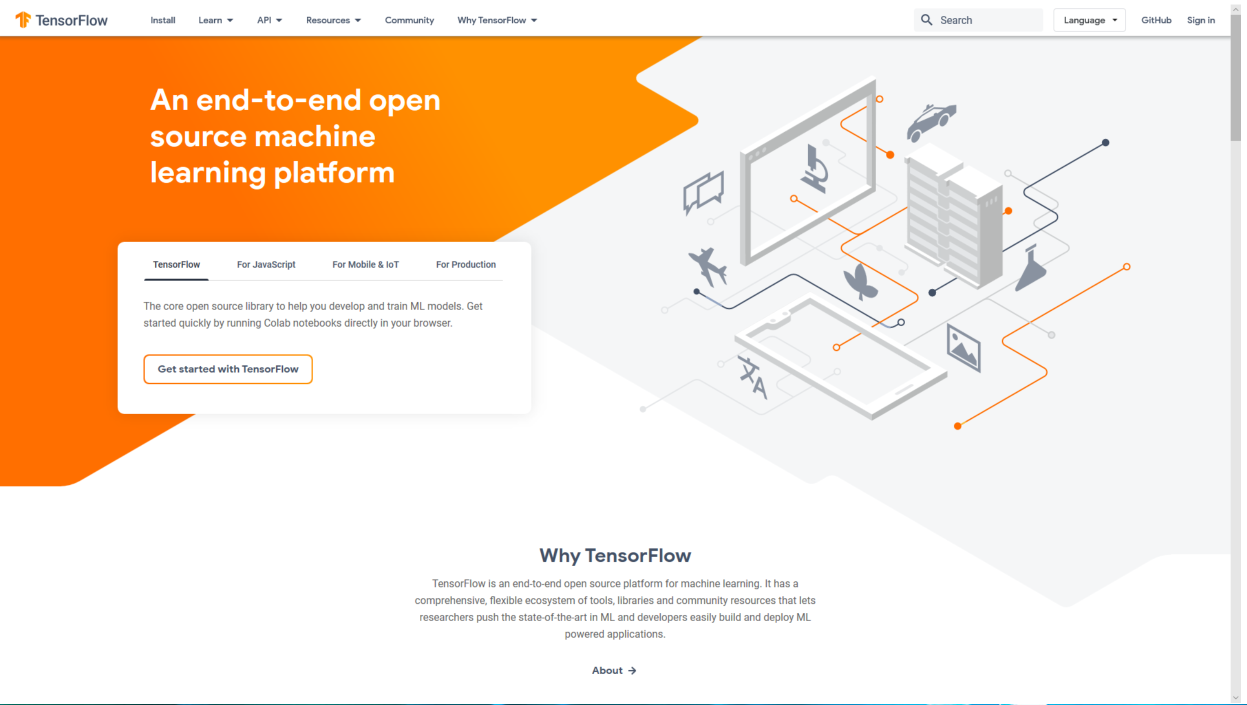




TensorFlow

- These images are then split into three sets. ~45% go into the training set, ~45% go into the testing set and ~10% go into the validation set. Images which don't contain the true vertex are discarded for the training set. Images where the true vertex is not inside the detector volume are discarded.

- The training set is then input to a CNN using TensorFlow. More information on it can be found at www.tensorflow.org.



An end-to-end open source machine learning platform

The core open source library to help you develop and train ML models. Get started quickly by running Colab notebooks directly in your browser.

Why TensorFlow

TensorFlow is an end-to-end open source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries and community resources that lets researchers push the state-of-the-art in ML and developers easily build and deploy ML powered applications.

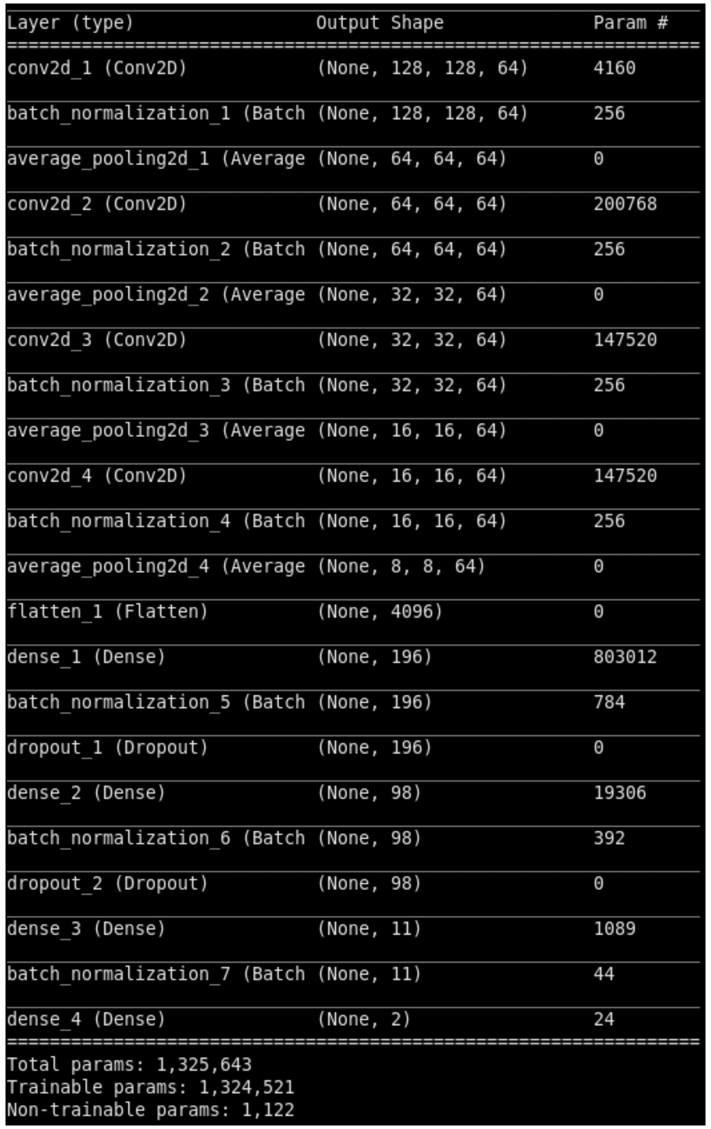
- The network architecture is shown on the next slide. The CNN takes the images as input and outputs the predicted 2D vertex.

- The CNN model once saved is ~15MB. It takes ~2 hour to train each model on the current computers being used.

Network Architecture

Four Convolutional Layers

Four Dense Layers



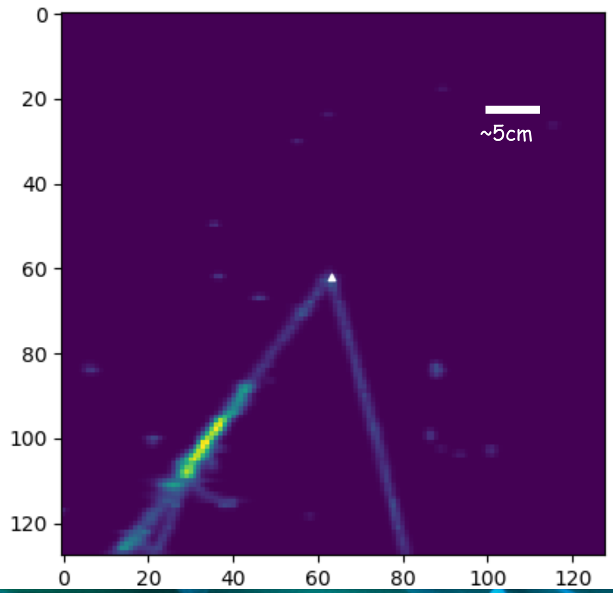
```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 128, 128, 64)      4160
_________________________________________________________________
batch_normalization_1 (Batch (None, 128, 128, 64)      256
_________________________________________________________________
average_pooling2d_1 (Average (None, 64, 64, 64)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 64, 64, 64)        200768
_________________________________________________________________
batch_normalization_2 (Batch (None, 64, 64, 64)        256
_________________________________________________________________
average_pooling2d_2 (Average (None, 32, 32, 64)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 32, 32, 64)        147520
_________________________________________________________________
batch_normalization_3 (Batch (None, 32, 32, 64)        256
_________________________________________________________________
average_pooling2d_3 (Average (None, 16, 16, 64)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 16, 16, 64)        147520
_________________________________________________________________
batch_normalization_4 (Batch (None, 16, 16, 64)        256
_________________________________________________________________
average_pooling2d_4 (Average (None, 8, 8, 64)          0
_________________________________________________________________
flatten_1 (Flatten)          (None, 4096)              0
_________________________________________________________________
dense_1 (Dense)              (None, 196)               803012
_________________________________________________________________
batch_normalization_5 (Batch (None, 196)               784
_________________________________________________________________
dropout_1 (Dropout)          (None, 196)               0
_________________________________________________________________
dense_2 (Dense)              (None, 98)                19306
_________________________________________________________________
batch_normalization_6 (Batch (None, 98)                392
_________________________________________________________________
dropout_2 (Dropout)          (None, 98)                0
_________________________________________________________________
dense_3 (Dense)              (None, 11)                1089
_________________________________________________________________
batch_normalization_7 (Batch (None, 11)                44
_________________________________________________________________
dense_4 (Dense)              (None, 2)                 24
=================================================================
Total params: 1,325,643
Trainable params: 1,324,521
Non-trainable params: 1,122
```

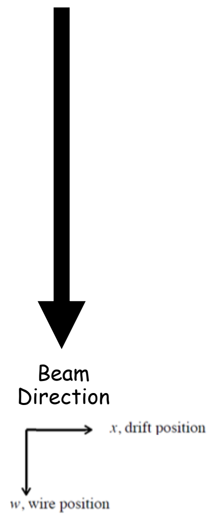
Part 2 of the Algorithm

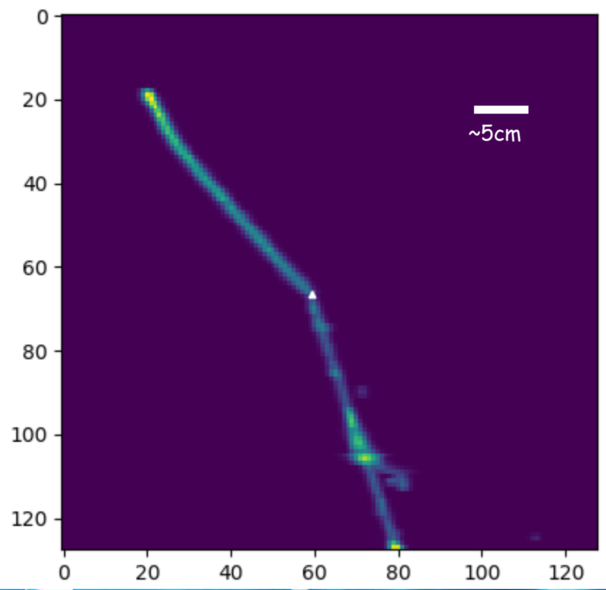
- The three predicted 2D vertices are then combined to form a 3D vertex.

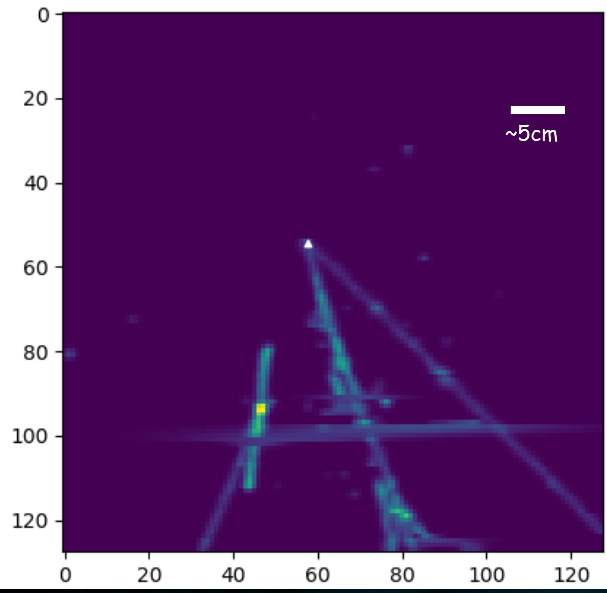
- This 3D vertex was then used to create a new image for each event (for each view) that is centred on this first estimate of the vertex and is a fixed size of 50cm by 50cm and 128 pixels by 128 pixels.

- A new network is then trained on these images.





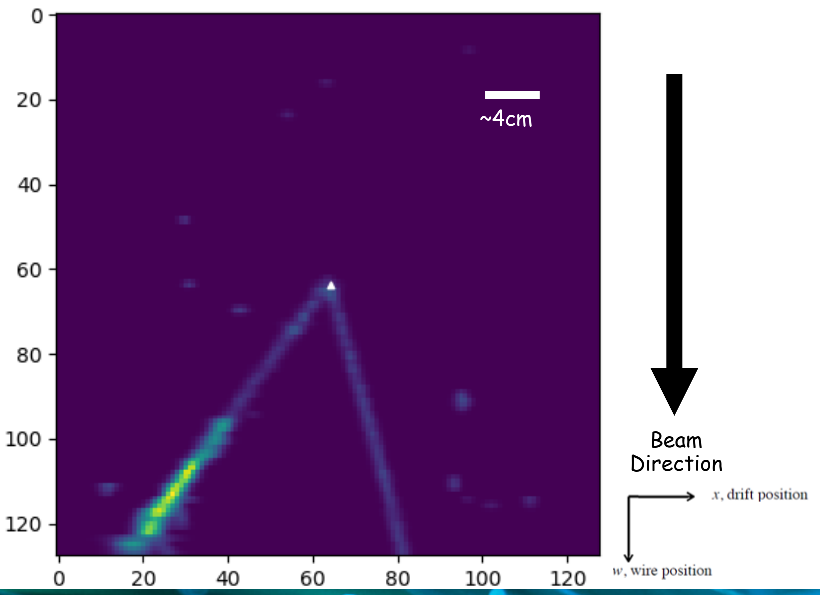


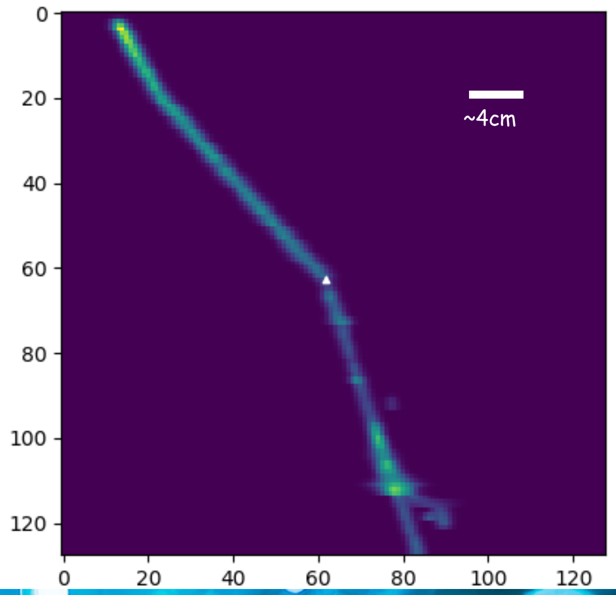


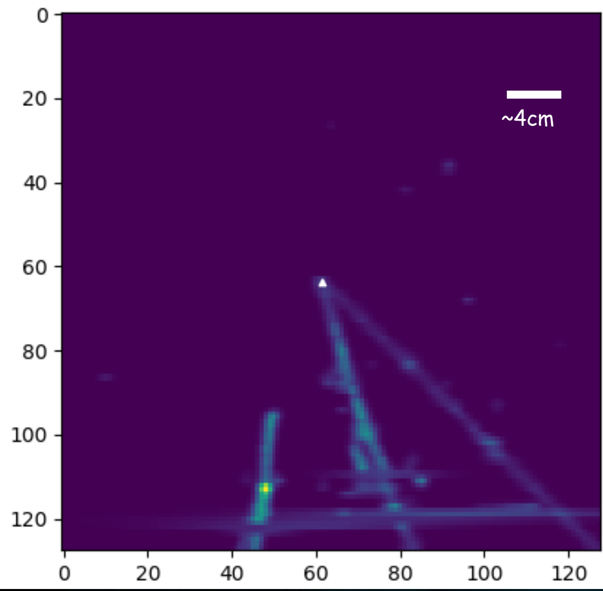
Part 3 of the Algorithm

- The combined output 3D vertex from Part 2 was then used to create a new image for each event (for each view) that is centred on this second estimate of the vertex and is a fixed size of 40cm by 40cm and 128 pixels by 128 pixels. Images which do not contain the true vertex are discarded for the training set but not for the testing set.

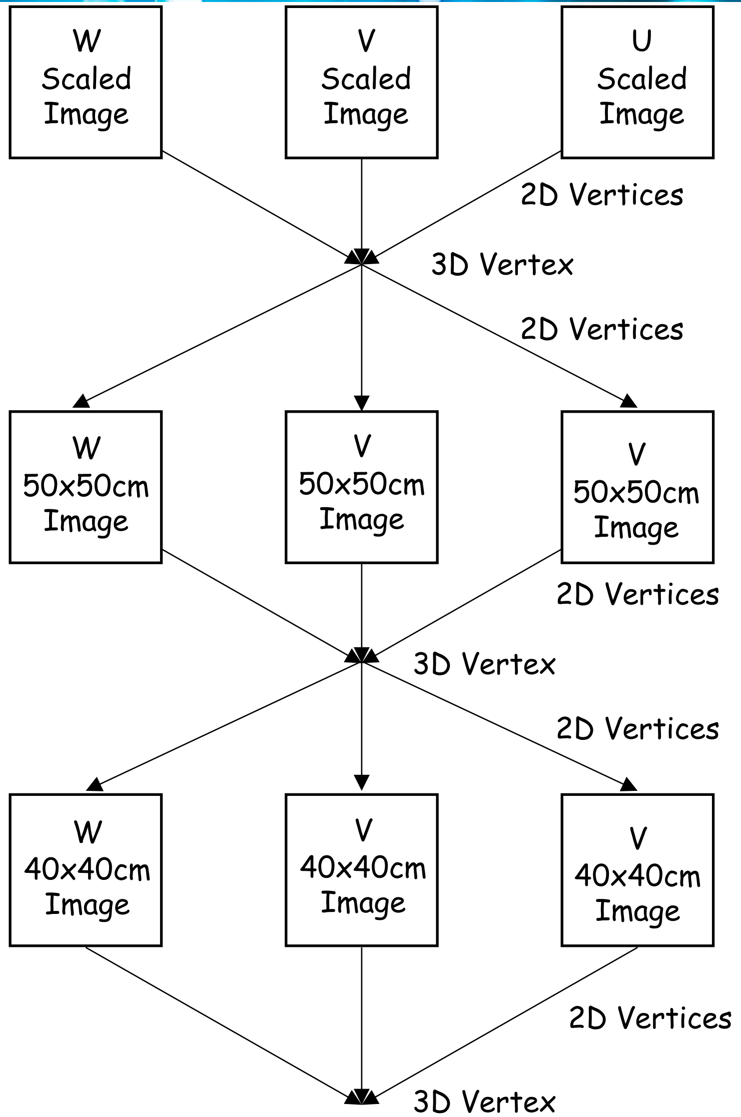
- The reason that these final images are 40cm by 40cm is that with 128 pixels by 128 pixels, each pixel is 0.3125cm.This number is good as it is small but not much smaller than the Wire Pitch (~0.5cm) or the peak hit width (~0.32cm).







DL Vertex Algorithm Summary

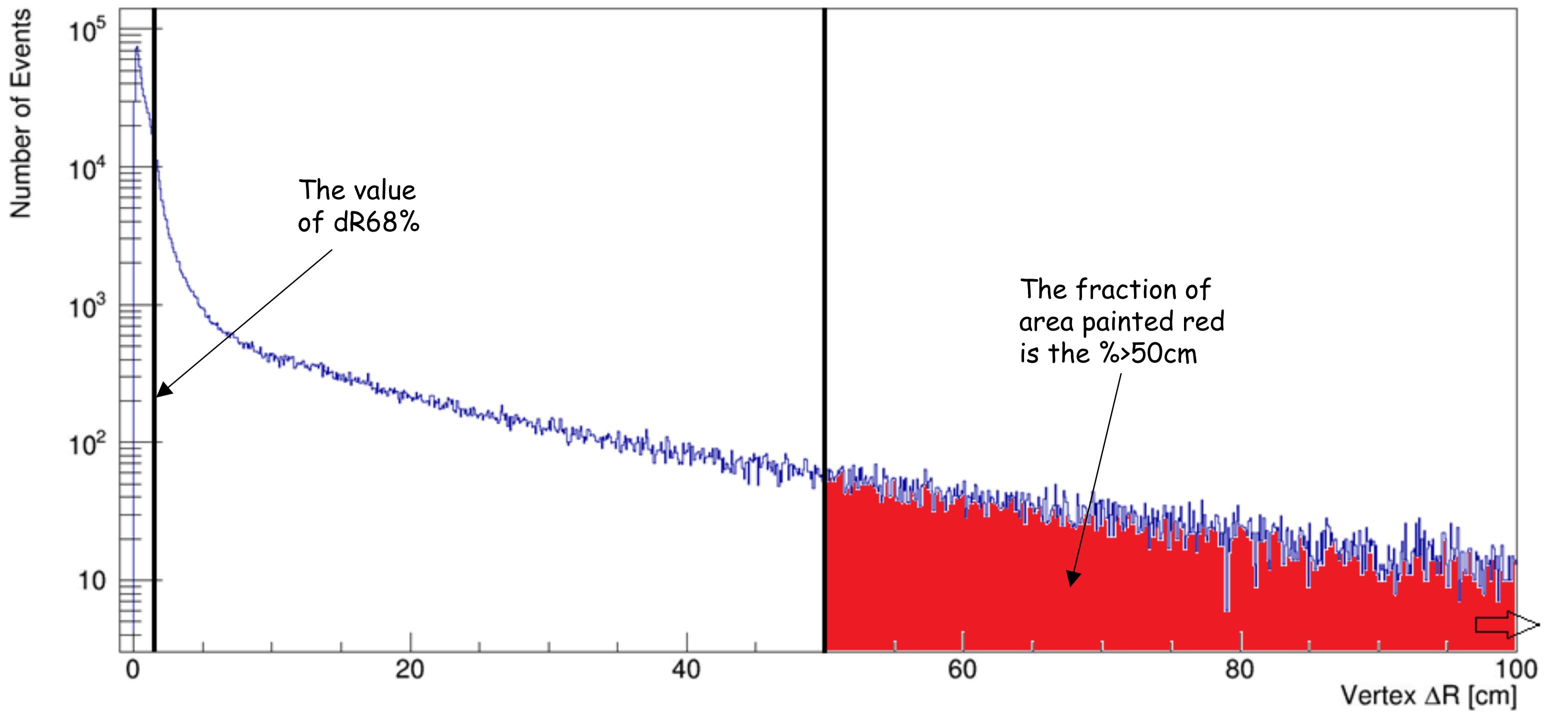


DL Vertex Feature Tool

- A new network is then trained on these images. The three predicted 2D vertices are then combined to form a 3D vertex. The accuracy of this 3D vertex is good compared to the current BDT.
- However, for best performance this 3D vertex output is then used to create a score which is added to the BDT Vertex selection algorithm along with the other standard scores.
- The DL Score is calculated by finding the distance between the DL 3D vertex and each of the candidate vertices, then normalising this number by a measure of the event's span and then scaling it.
- As well as this a candidate vertex is created at the position of the DL 3D vertex.
- A table of the performance metrics for the current BDT vertexing and the BDT vertexing with an added Deep Learning score is shown after slides defining the performance metrics.

Vertex Reconstruction Performance Metrics

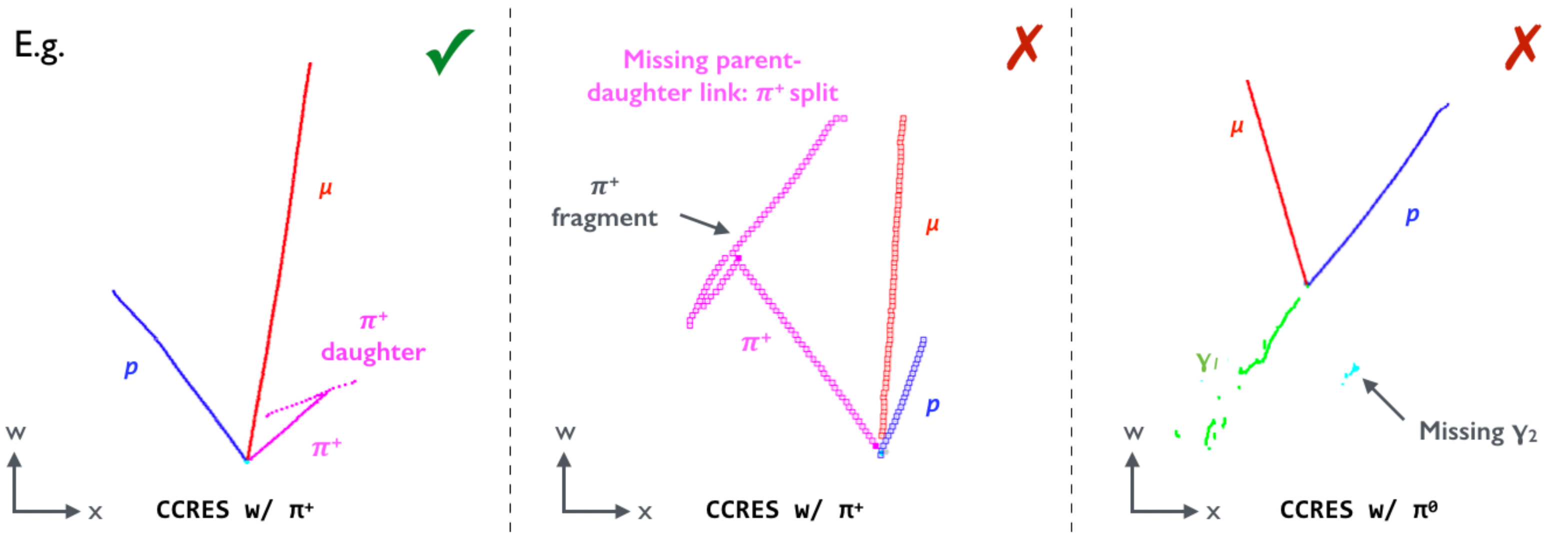
- For vertexing, in specific, main two performance metrics are “dR68%” and “%>50cm”.
  - “dR68%” is the value of deltaR where 68% of events have a deltaR less than that.
  - “%>50cm” is the percentage of events that have a deltaR larger than 50cm



Pandora General Reconstruction Performance Metrics

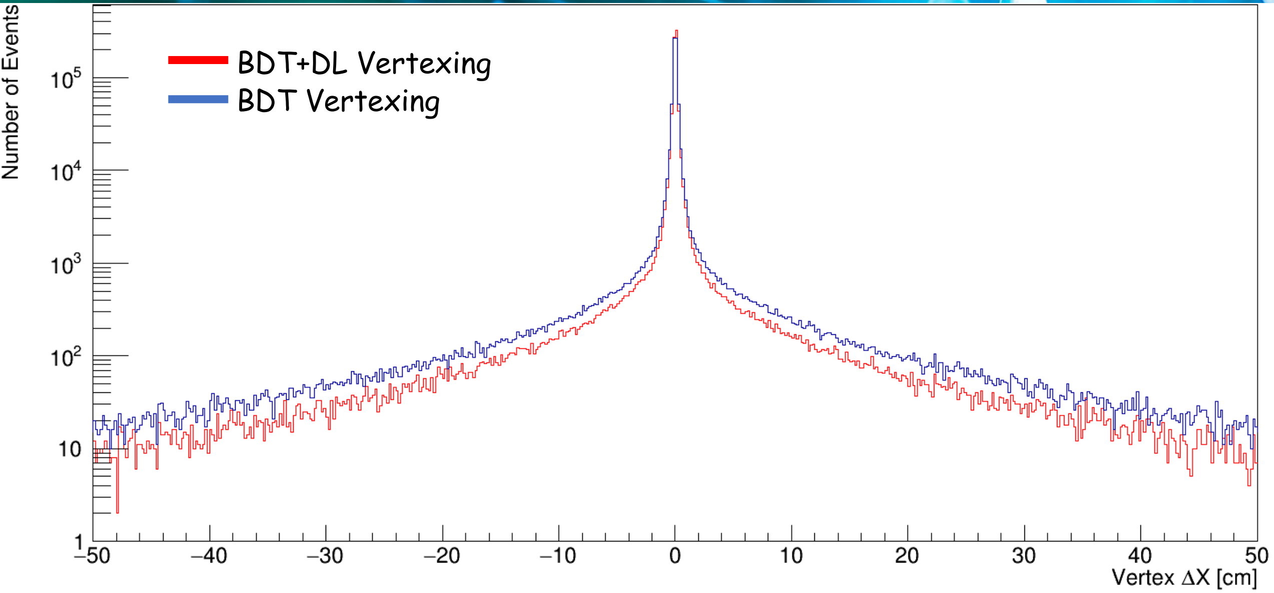
- To assess general reconstruction performance for simulated DUNE events, similar performance metrics were used as at MicroBooNE.

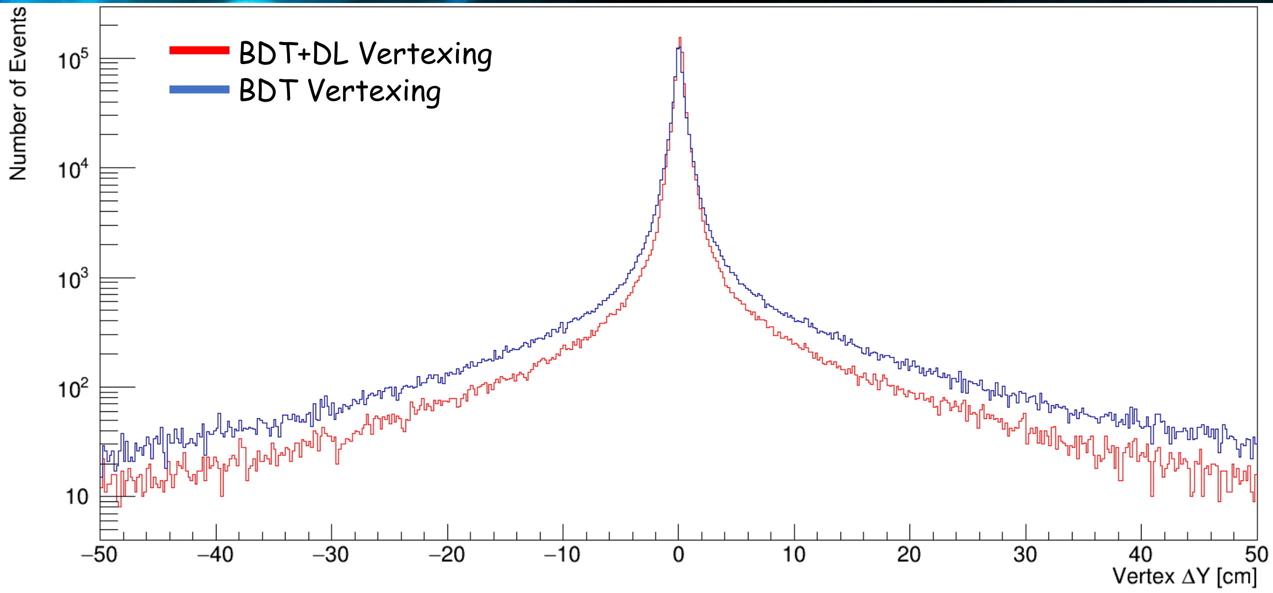
- Examine fraction of events deemed "correct" by very strict pattern-recognition metrics:
  - Consider exclusive final-states where all true particles pass simple quality cuts (e.g. nHits)
  - Correct means exactly one reconstructed primary particle is matched to each true primary particle

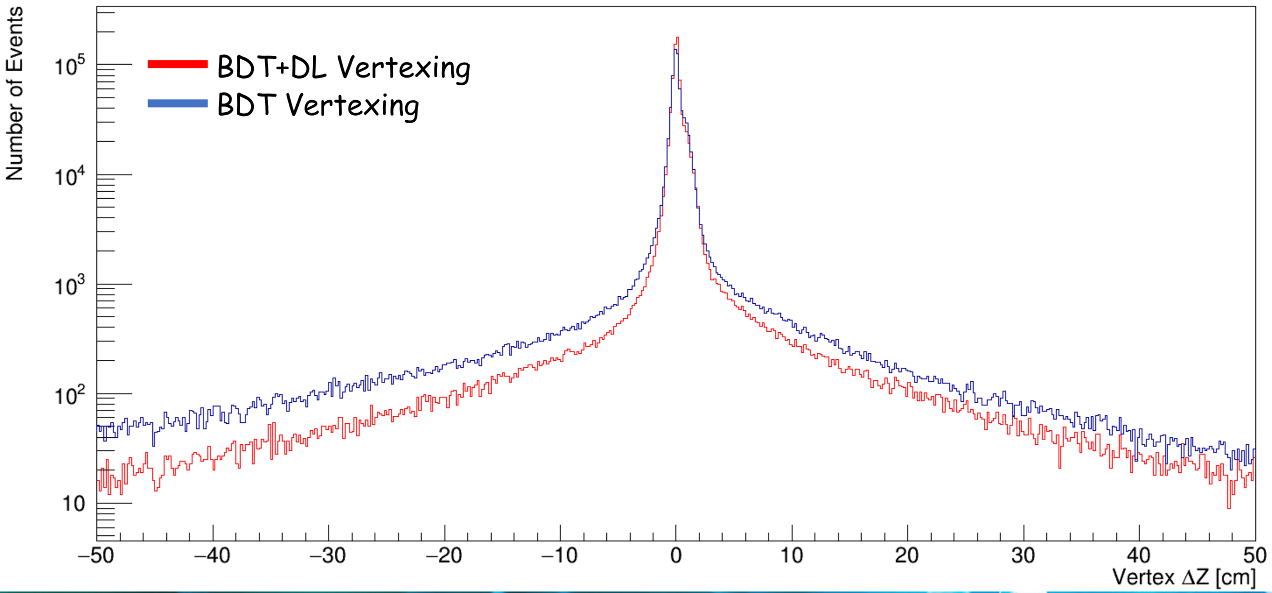


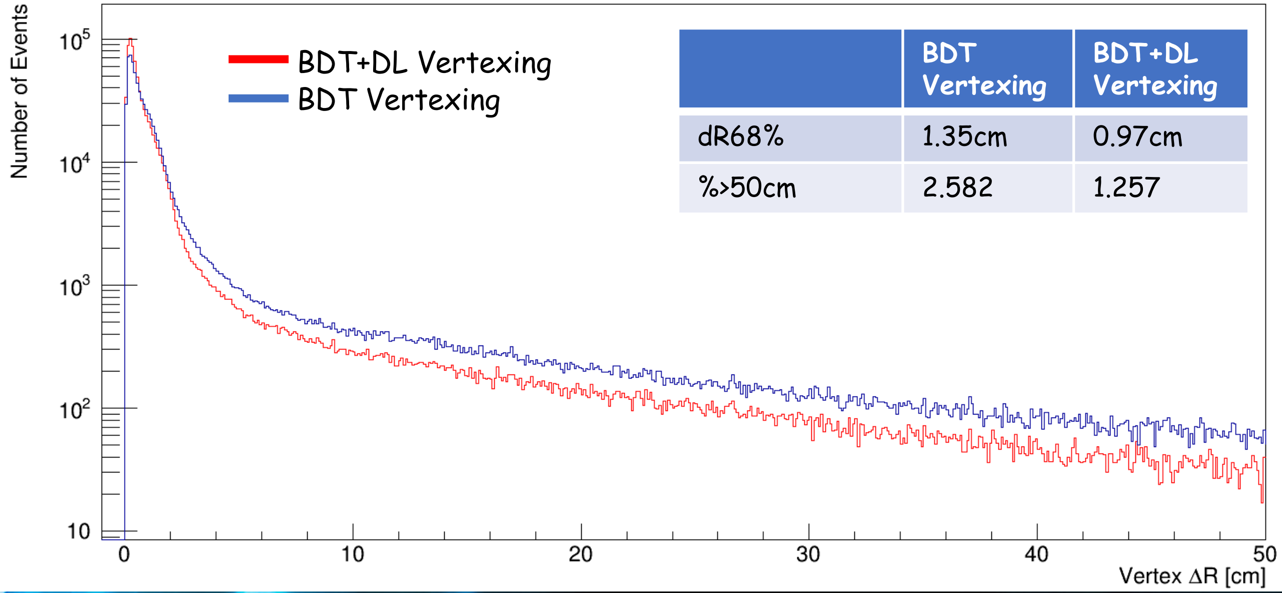
For more details see: Pandora MicroBooNE Paper. Eur. Phys. J. C. (2018) 78: 82

(Reco-True) Vertex Positions Distributions: All Interactions "nue"



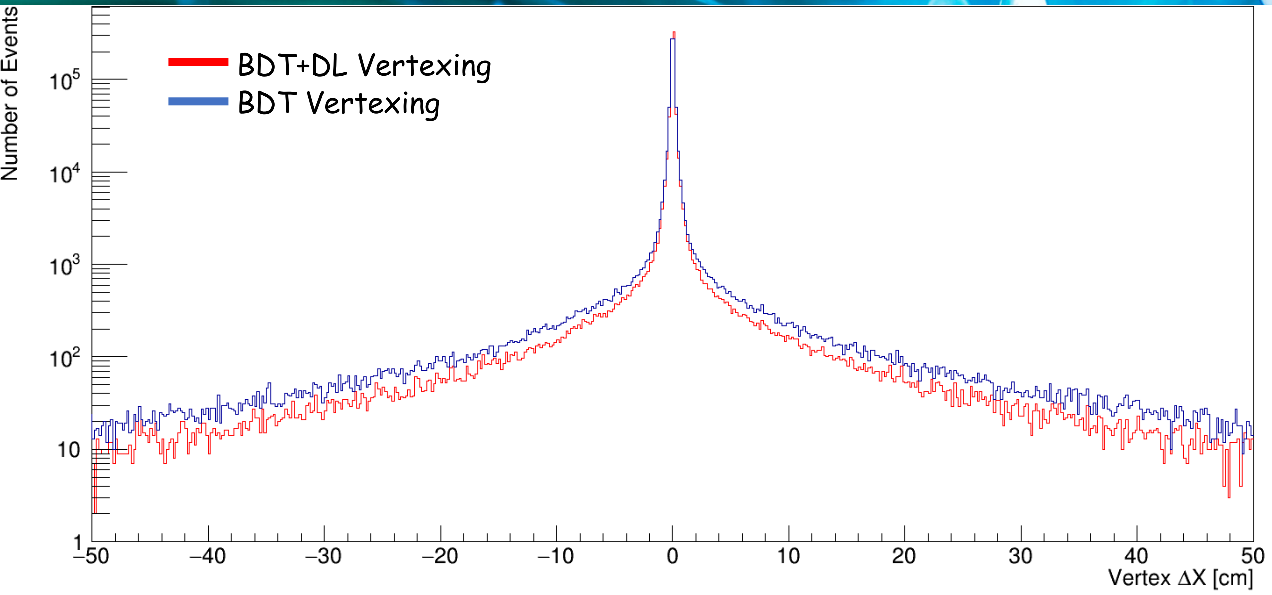


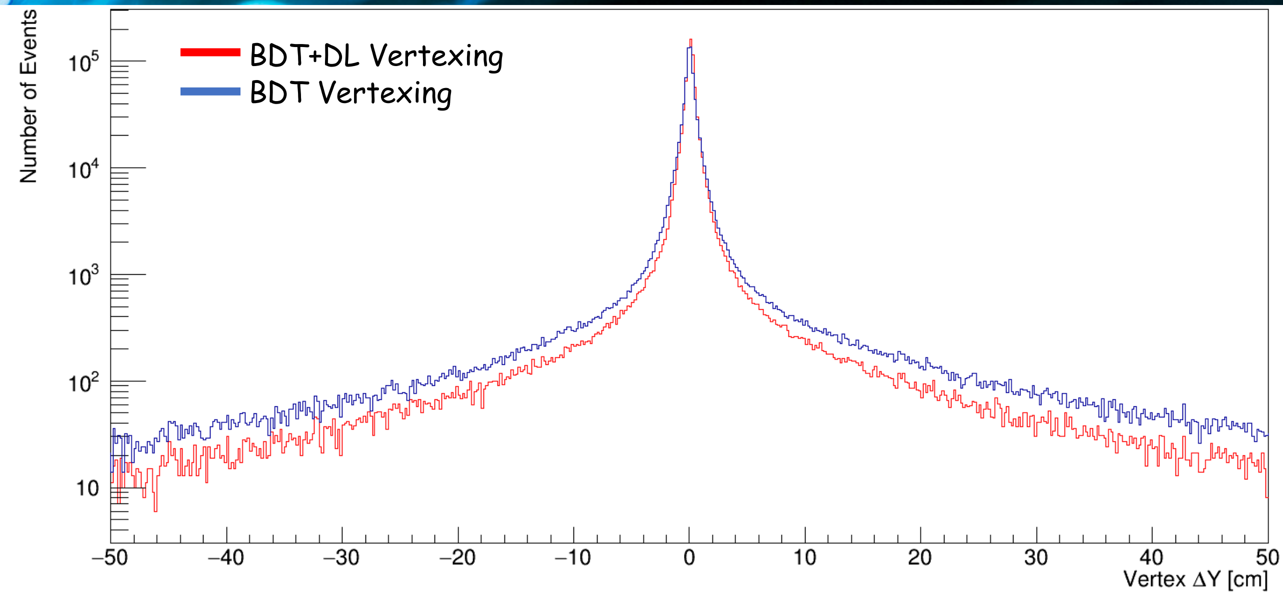


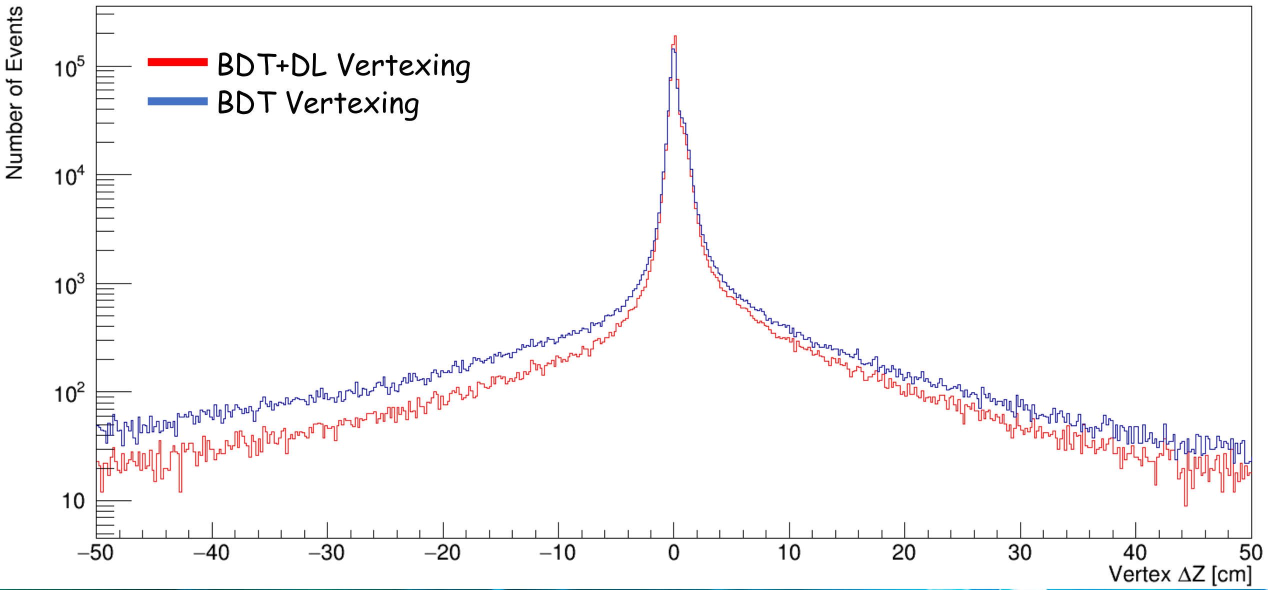


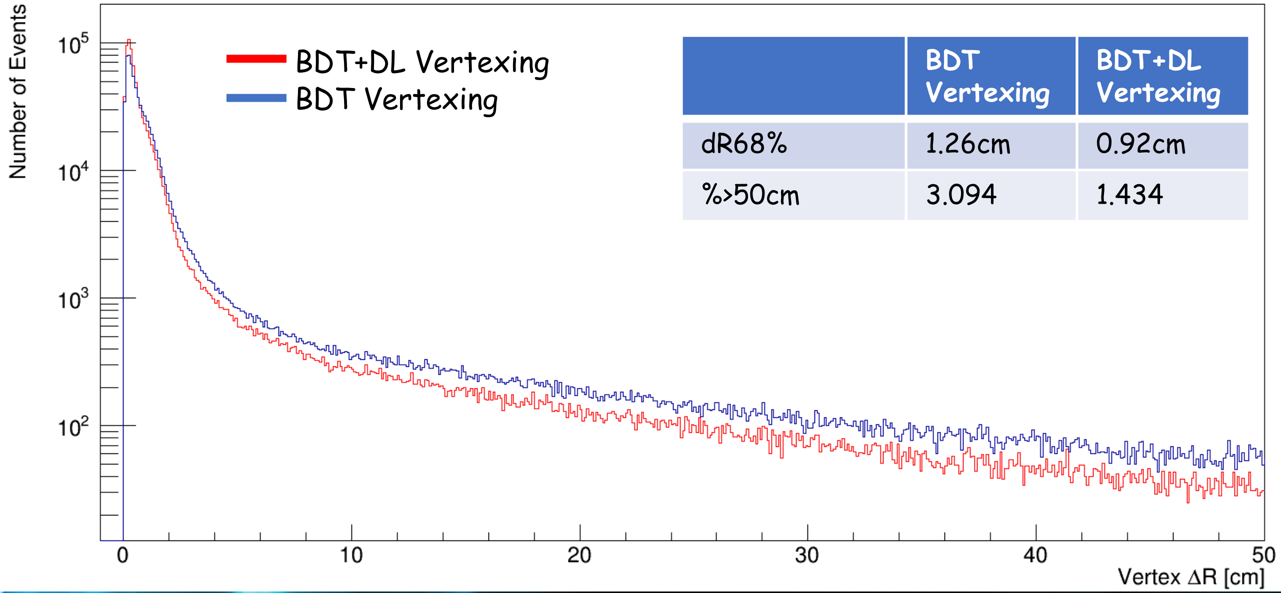
	BDT Vertexing	BDT+DL Vertexing
dR68%	1.35cm	0.97cm
%>50cm	2.582	1.257

(Reco-True) Vertex Positions Distributions. All Interactions "numu"



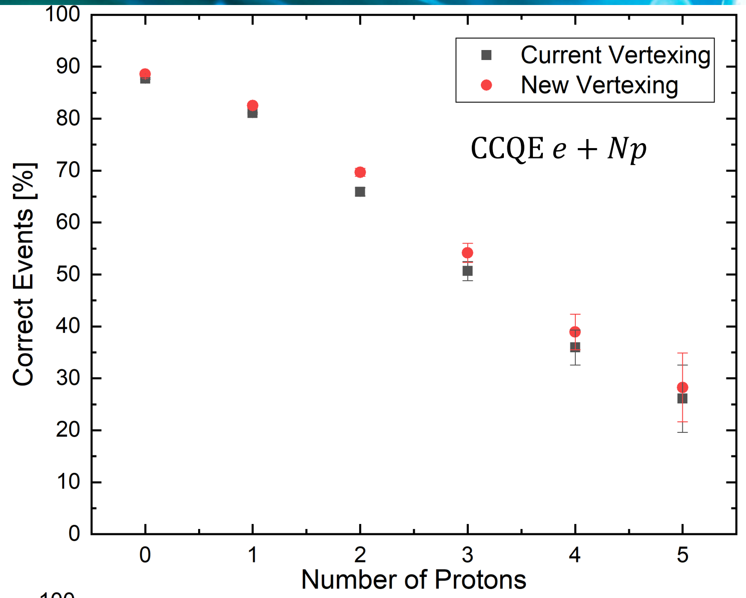


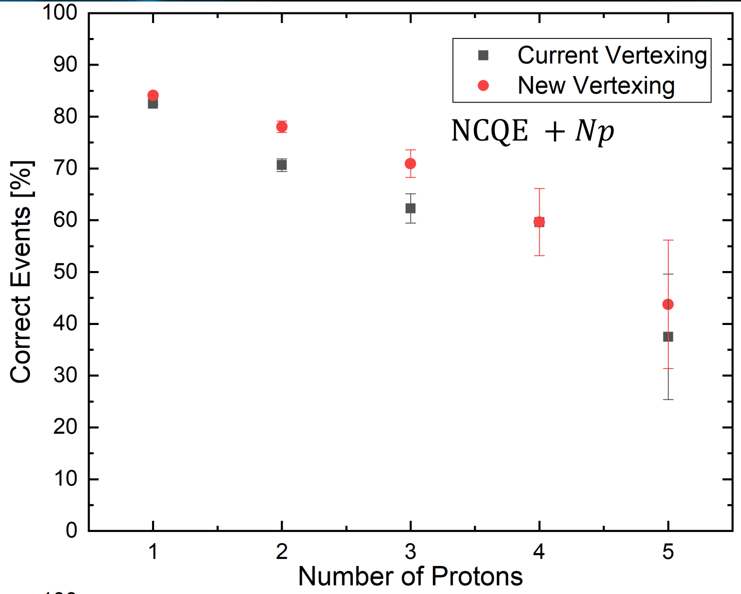


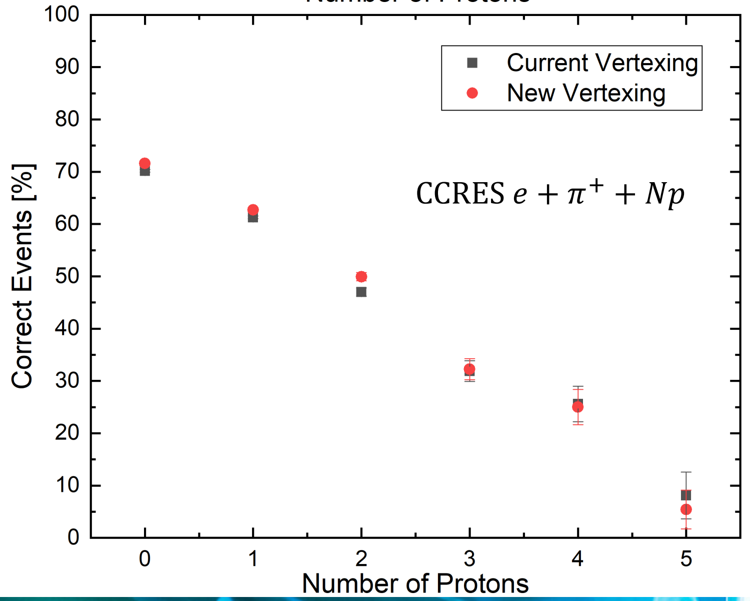


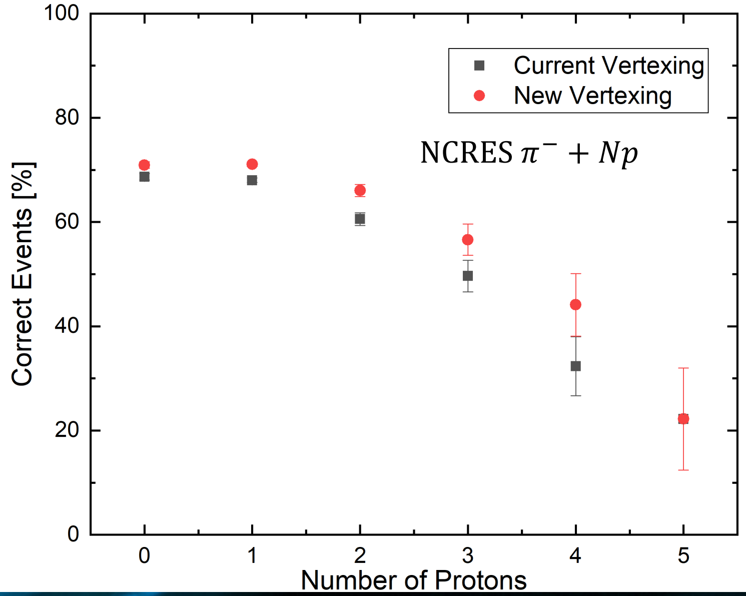
	BDT Vertexing	BDT+DL Vertexing
dR68%	1.26cm	0.92cm
%>50cm	3.094	1.434

Correct Event Fraction Plots

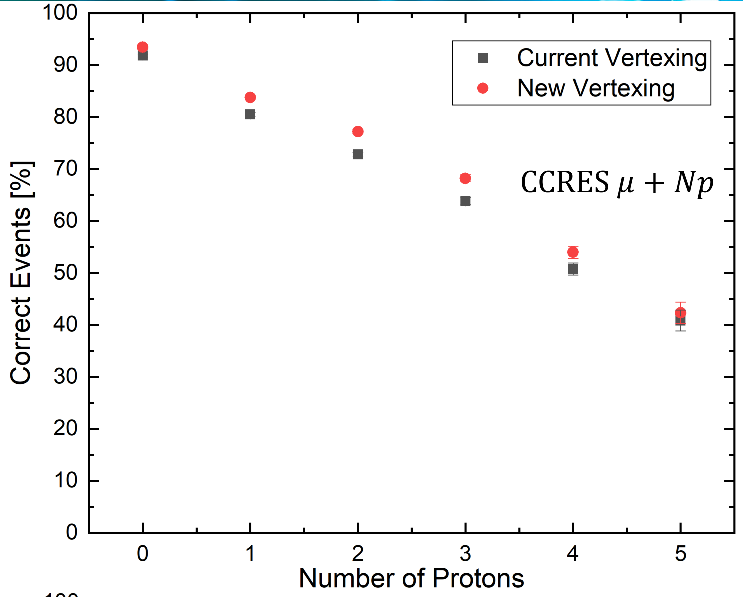


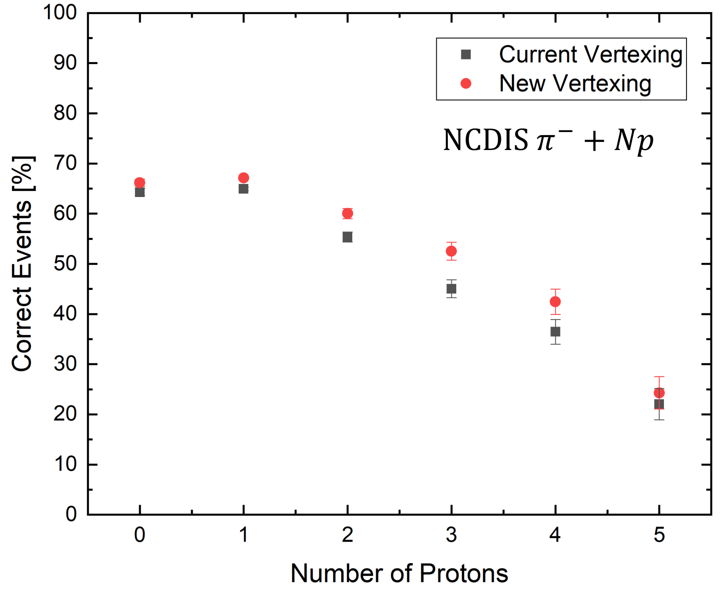


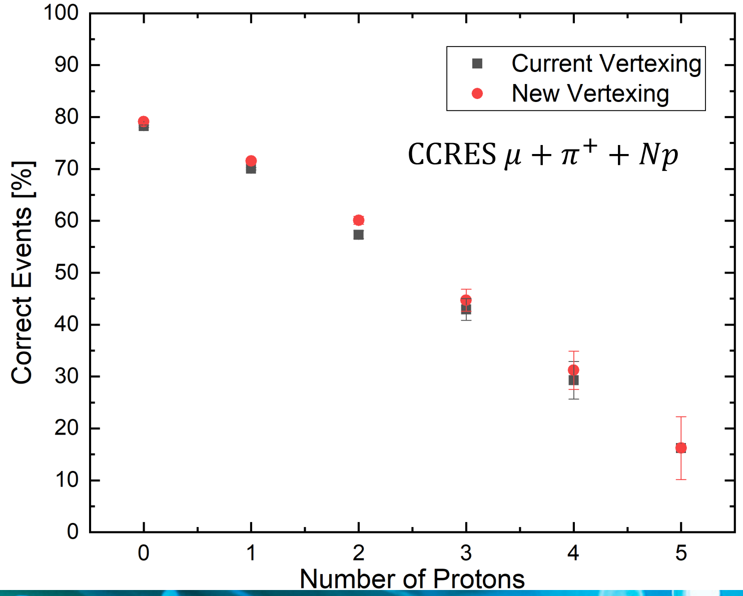


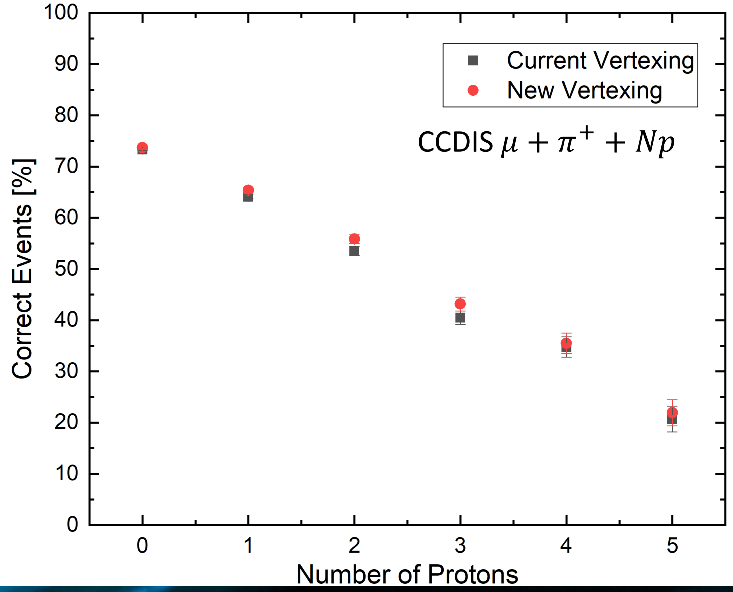


Correct Event Fraction Plots









Performance Metrics. Low Energy (MeV) Interactions.

Current BDT Vertexing	nu0_400	nu400_800	nue0_400	nue400_800
Number of Correct Events	662 /903 (73.31%)	8,255 /10,794 (76.48%)	536 /1,112 (48.20%)	6,771 /11,329 (59.77%)
dR68% (cm)	14.98	1.42	24.66	4.57
%>50cm	7.738	9.721	13.514	9.354

BDT with DL Score Vertexing	nu0_400	nu400_800	nue0_400	nue400_800
Number of Correct Events	667 /903 (73.86%)	8,421 /10,794 (78.02%)	578 /1,112 (51.98%)	7,254 /11,329 (64.03%)
dR68% (cm)	10.64	1.05	19.33	1.30
%>50cm	7.438	7.149	8.863	4.339

perfectVtx	nu0_400	nu400_800	nue0_400	nue400_800
Number of Correct Events	827 /903 (91.58%)	9,424 /10,794 (87.31%)	814 /1,112 (73.20%)	8,250 /11,329 (72.82%)
dR68% (cm)	0	0	0	0
%>50cm	0	0	0	0

Conclusions/Summary

- In conclusion, the vertexing has been improved in Pandora by utilizing a CNN coded with TensorFlow to create a feature score that is added to the current BDT vertex selection algorithm.

- **This new development provides a significant benefit to the quality of the vertex reconstruction for DUNE FD events compared to the current method (especially for nue).**

All Interactions "numu"	BDT Vertexing	BDT+DL Vertexing
dR68%	1.26cm	0.92cm
%>50cm	3.094	1.434

All Interactions "nue"	BDT Vertexing	BDT+DL Vertexing
dR68%	1.35cm	0.97cm
%>50cm	2.582	1.257

- Plans to continue developing algorithms to improve the vertex reconstruction incorporating aspects of Deep Learning.

- Working on efficiency improvements for vertex reconstruction, also on to do list.

The End

Thank you for your attention!

Backup Slides

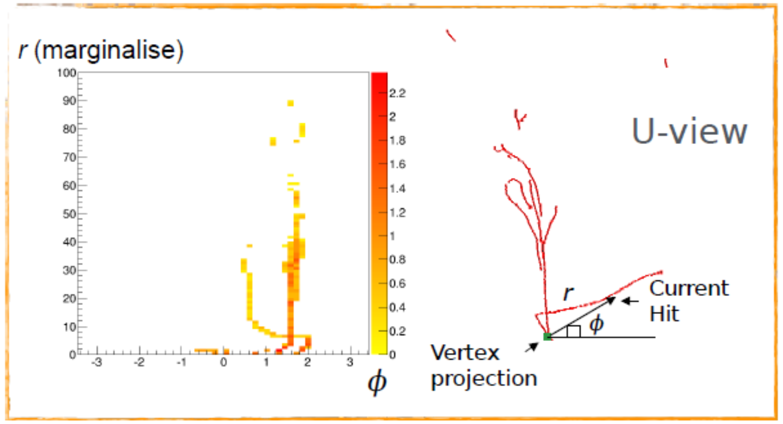
Current Pandora Vertex Selection Algorithm Features

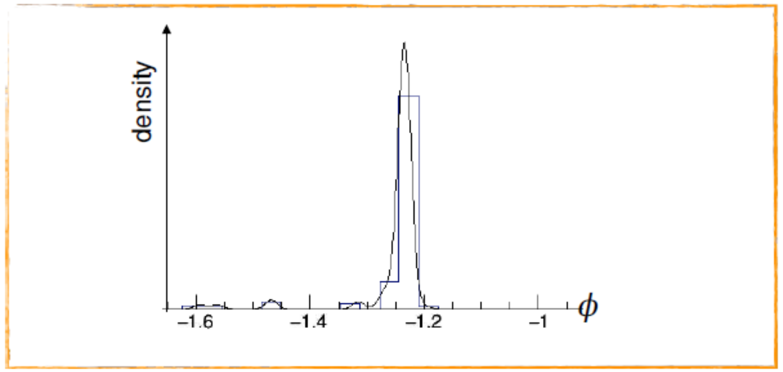
Region-Finding BDT:

- Variables:
  - ➔ Event-Based Features:
    - ➔ Shower Fraction (proportion of shower-cluster-associated hits)
    - ➔ Total Energy
    - ➔ Volume Spanned
    - ➔ Longitudinality (ratio of zspan to xspan)
    - ➔ Number of Hits
    - ➔ Number of Clusters
    - ➔ Number of Vertex Candidates
  - ➔ Vertex-Based Features:
    - ➔ Energy Kick
    - ➔ Local Asymmetry
    - ➔ Beam De-weighting
    - ➔ Global Asymmetry
    - ➔ Shower Asymmetry

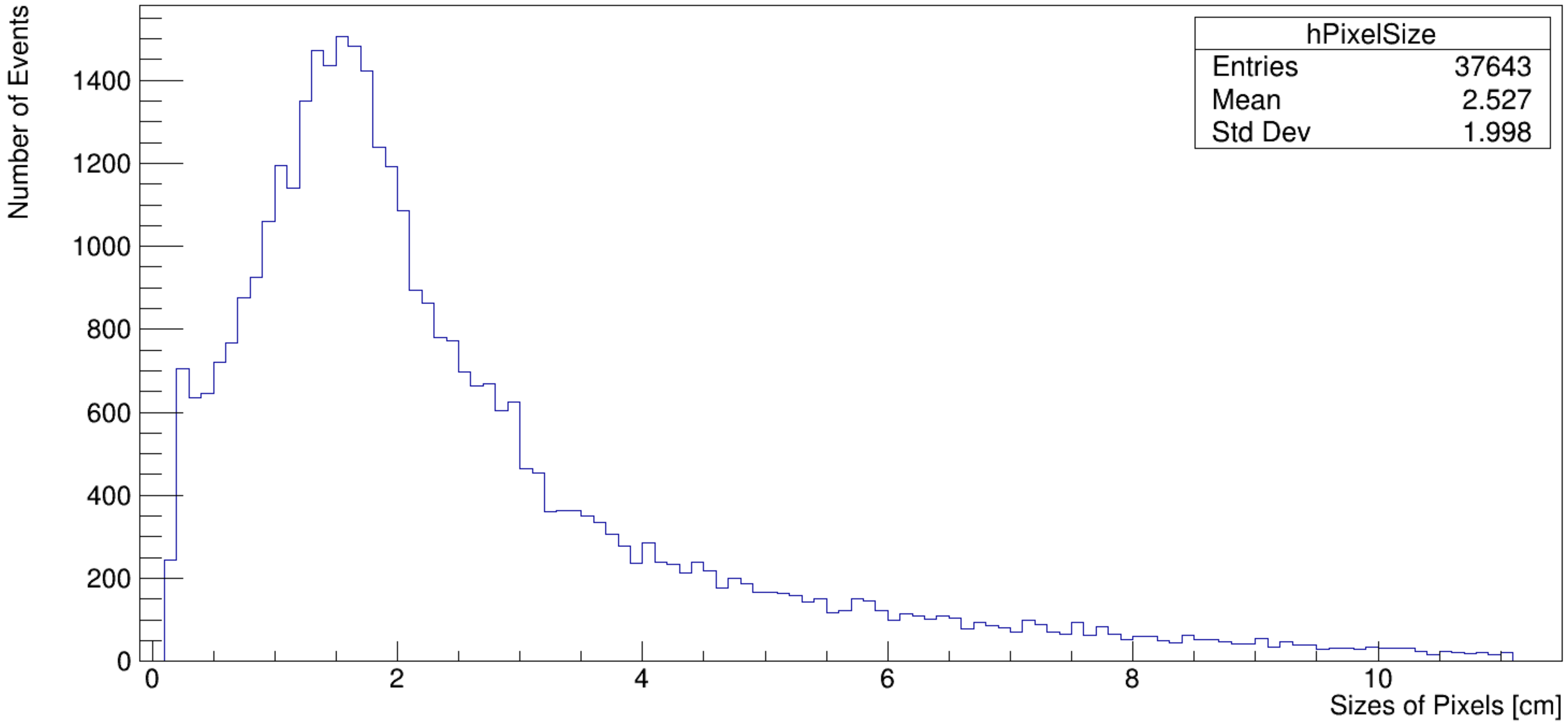
Vertex-Finding BDT:

- Same as above plus the vertex-based r/phi feature.

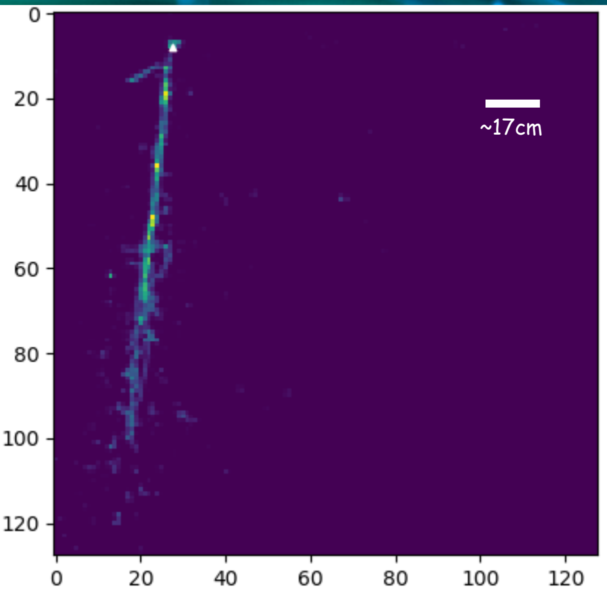


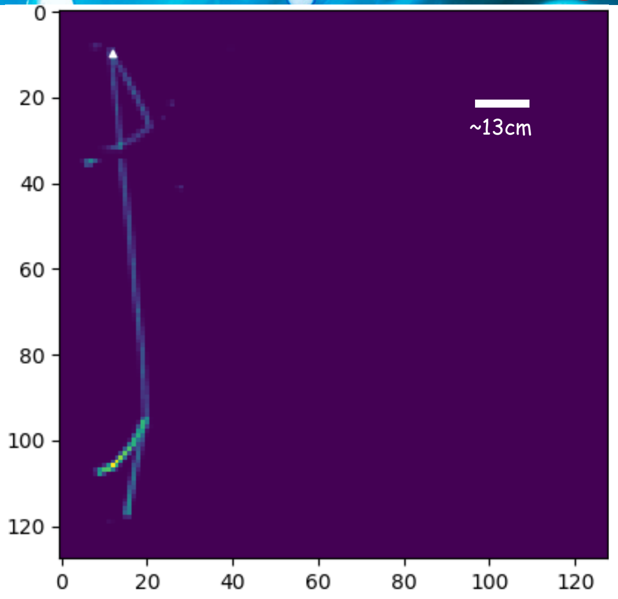


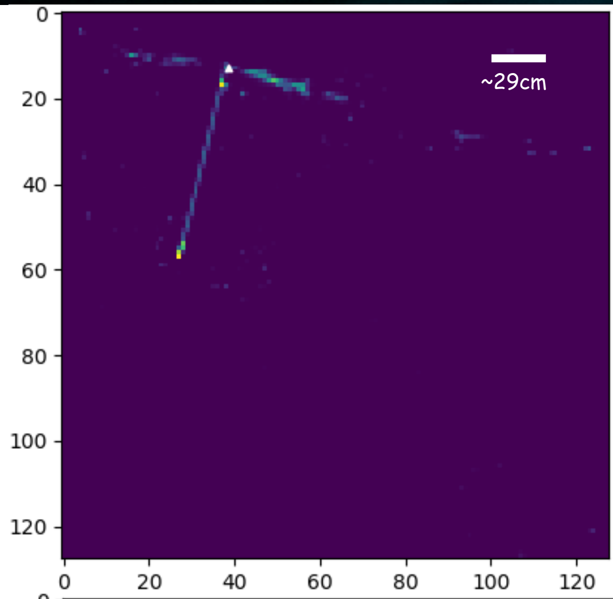
Histogram Of Pixel Sizes For W Images From Part 1 Of The Algorithm.

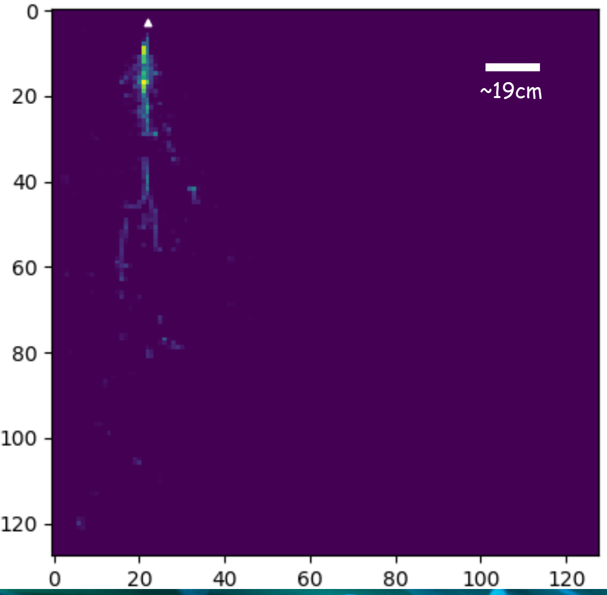


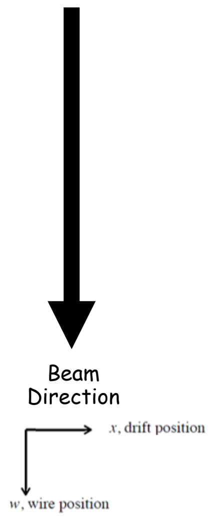
Example W View Deep Learning Training Images from Part 1 of the Algorithm. True Vertex=White Triangle

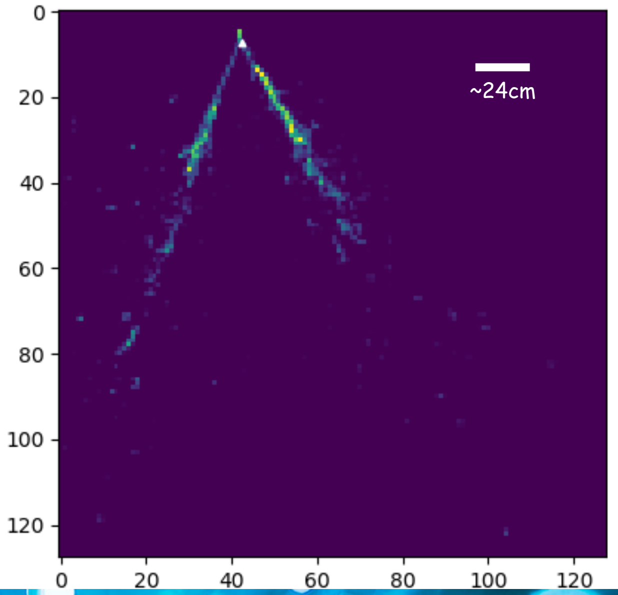


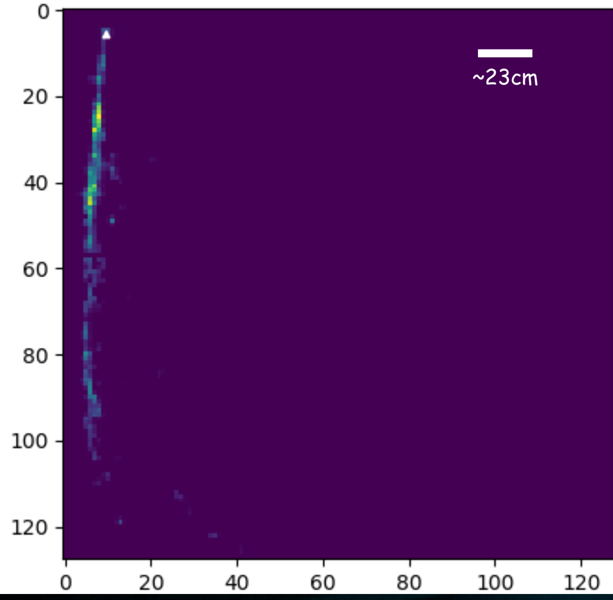




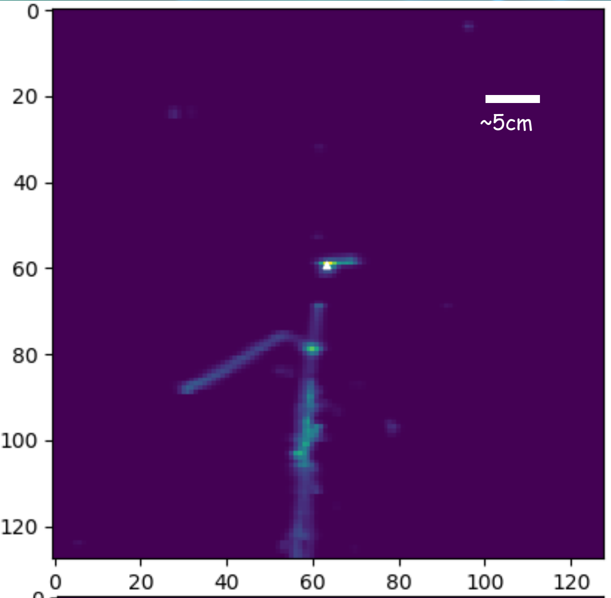


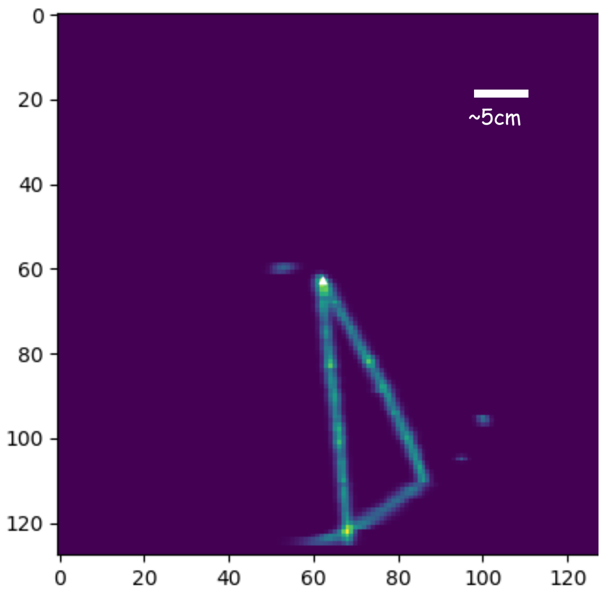


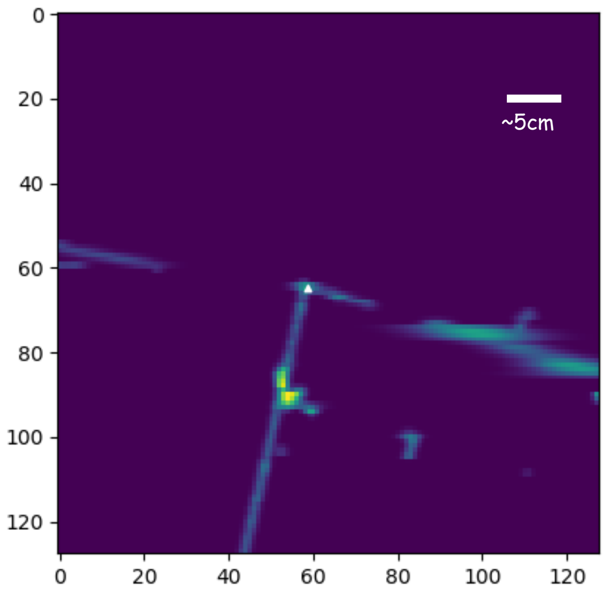


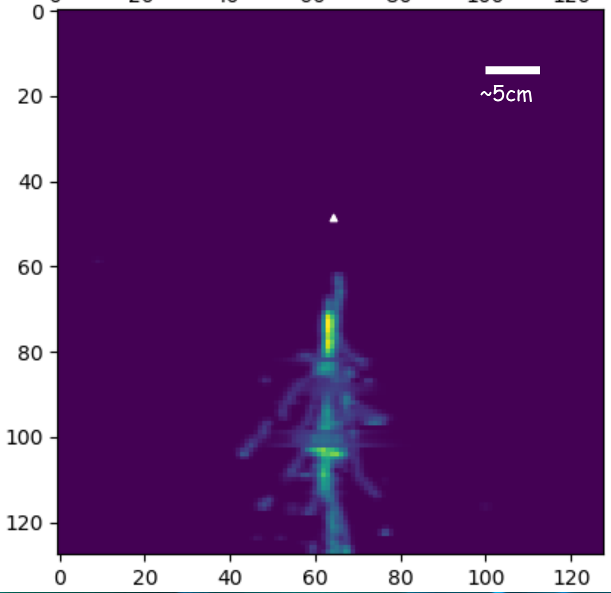


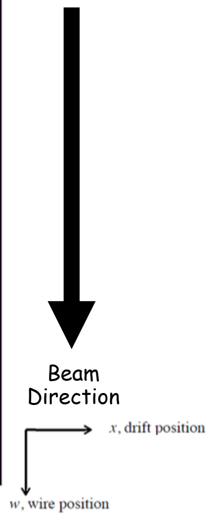
Example W View Deep Learning Training Images from Part 2 of the Algorithm. True Vertex=White Triangle

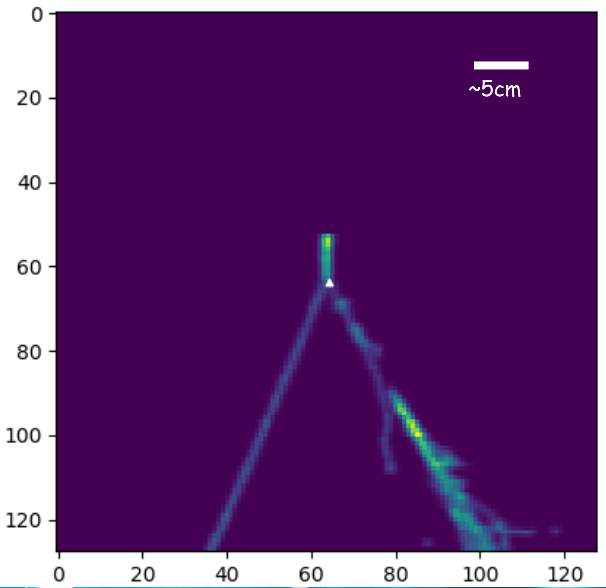


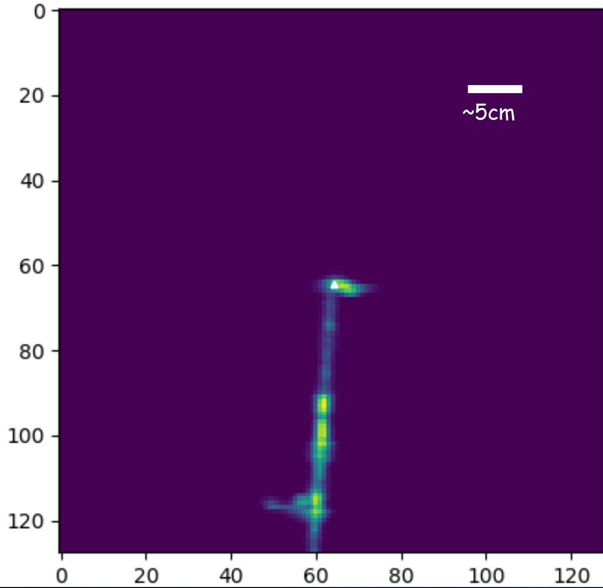




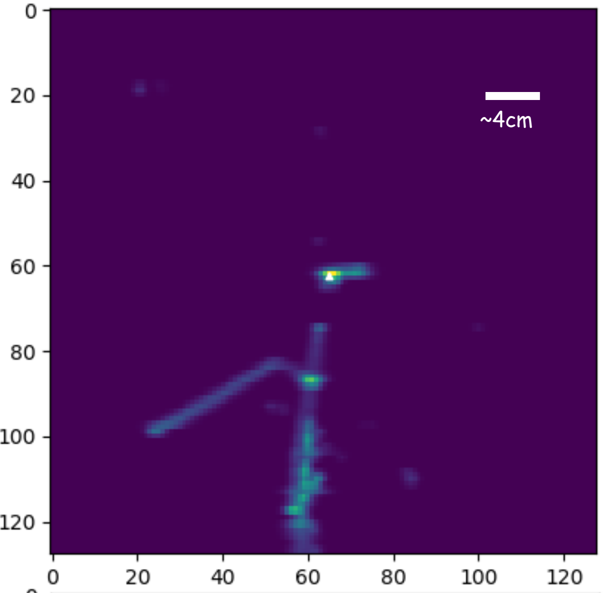


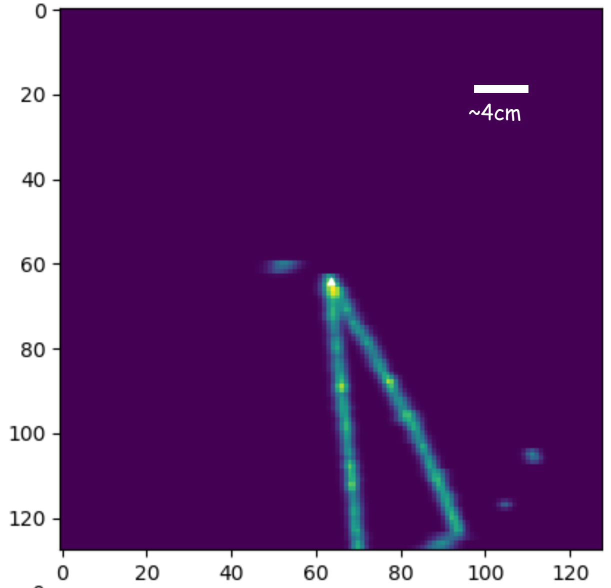


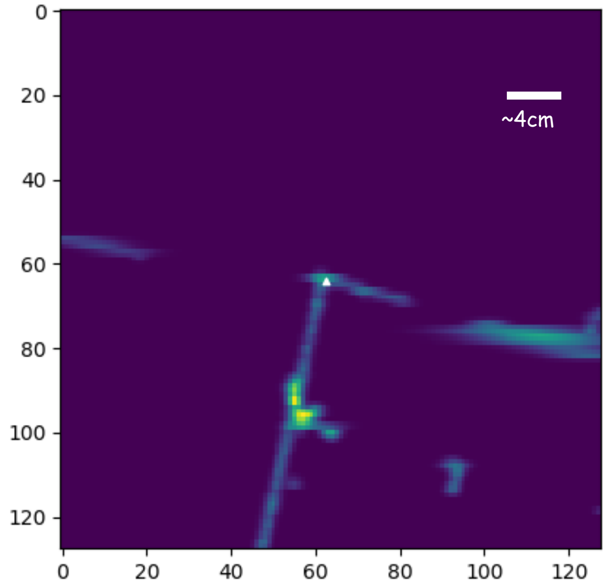


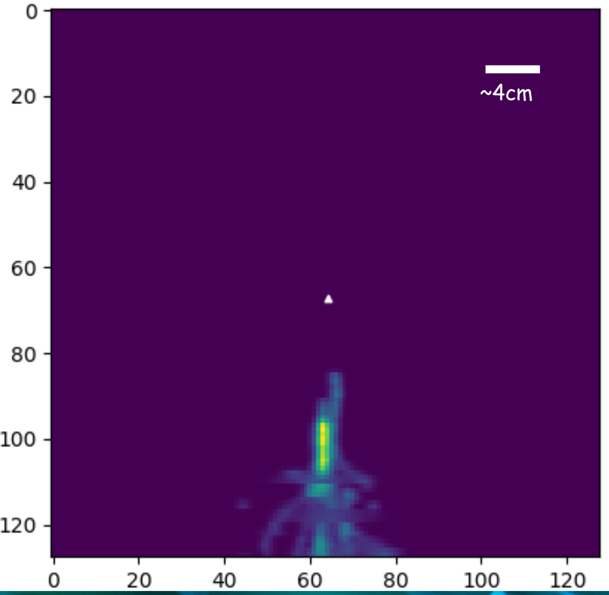


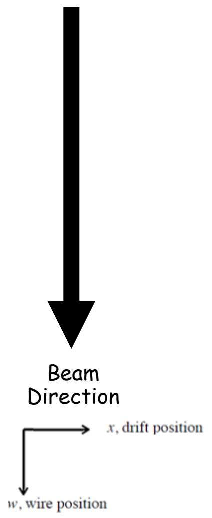
Example W View Deep Learning Training Images from Part 3 of the Algorithm. True Vertex=White Triangle

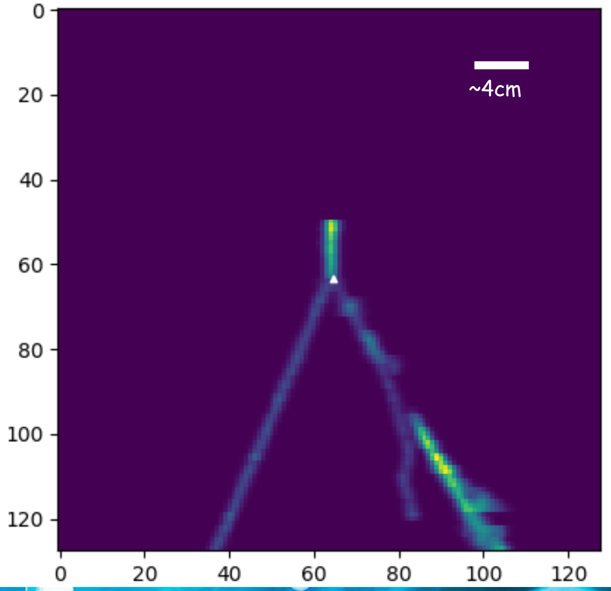


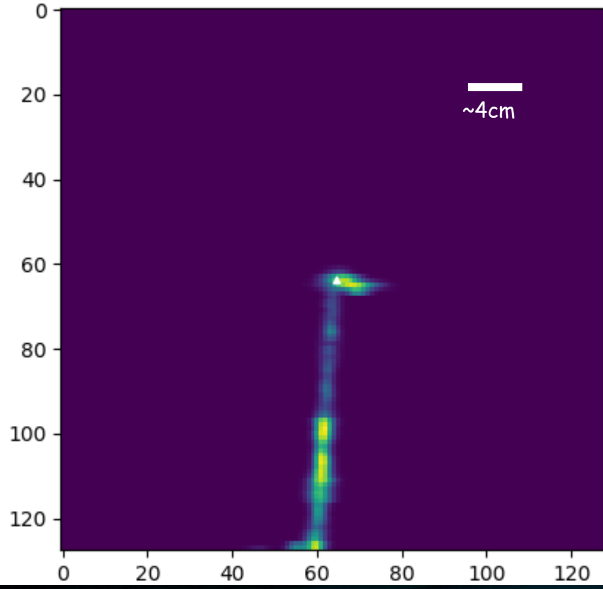






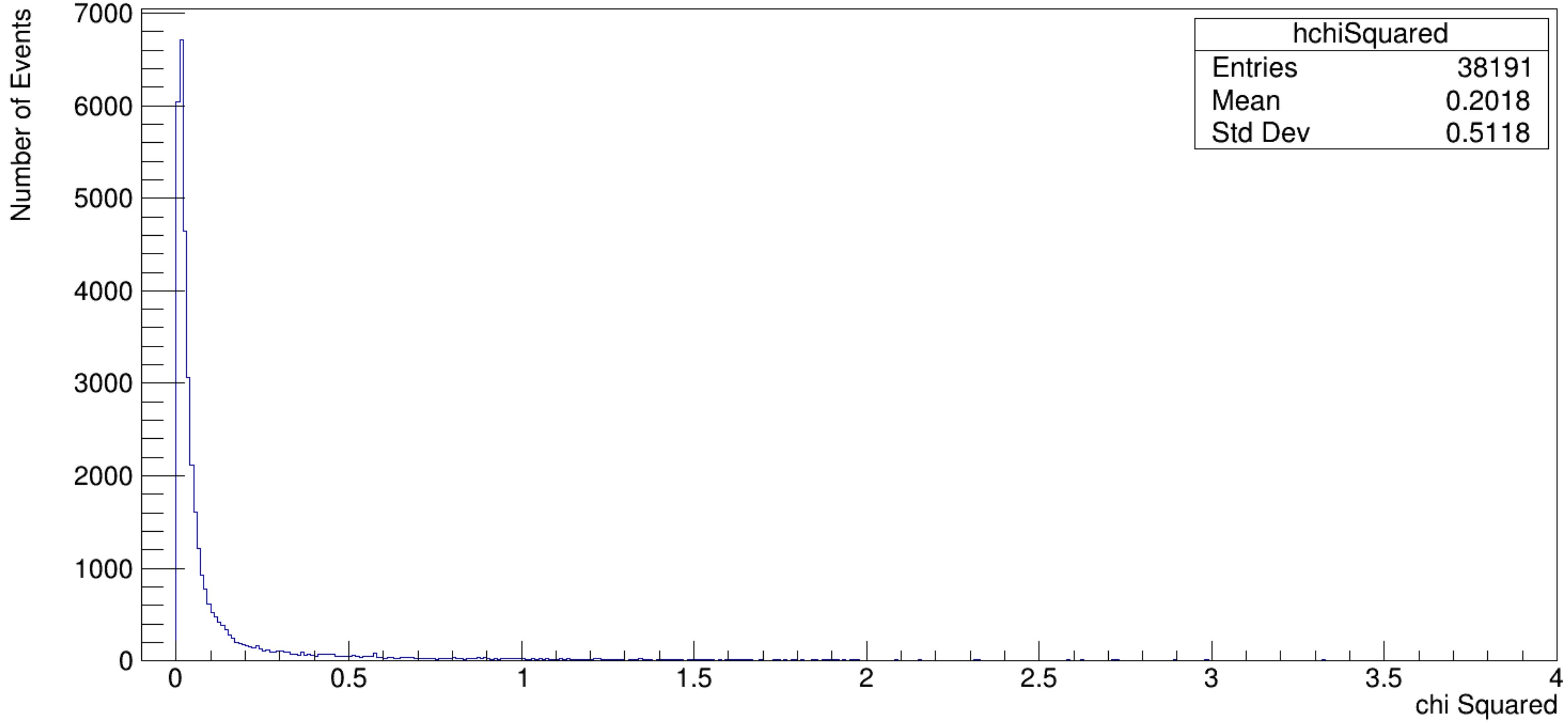






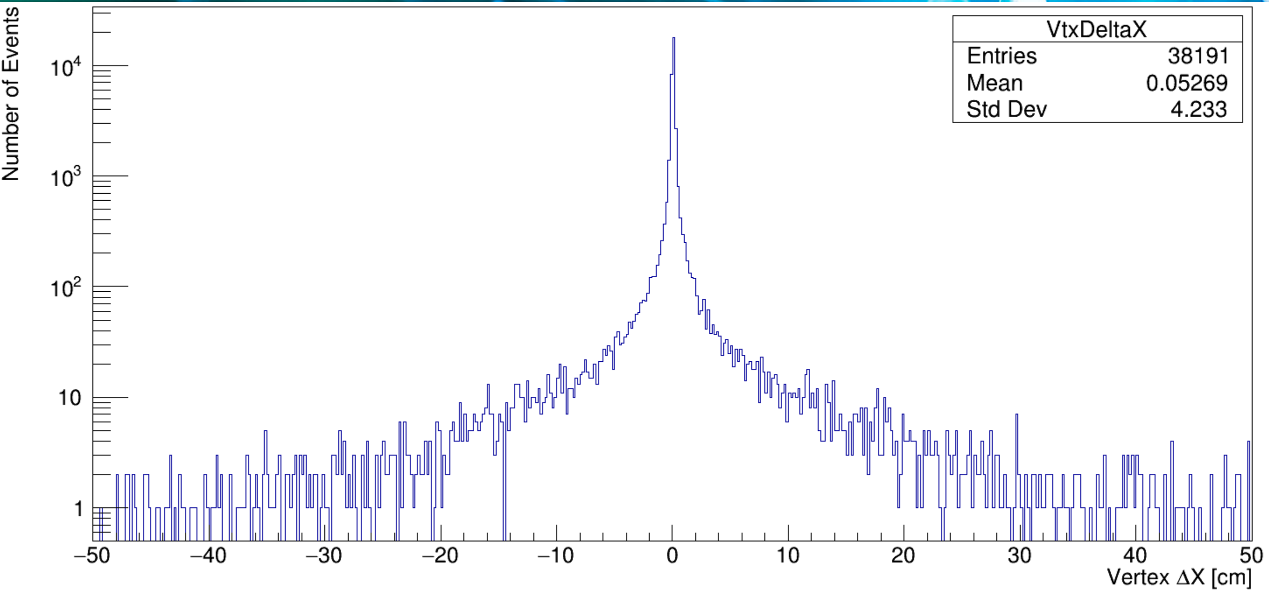
Reconstructing the Final CNN 3D Vertex

- The final three predicted 2D vertices from the DL algorithm are combined to form a final CNN 3D vertex using a Pandora function.
- The Pandora function also outputs a "chiSquared" value which is a measure of how consistent the final three 2D vertices are with each other.



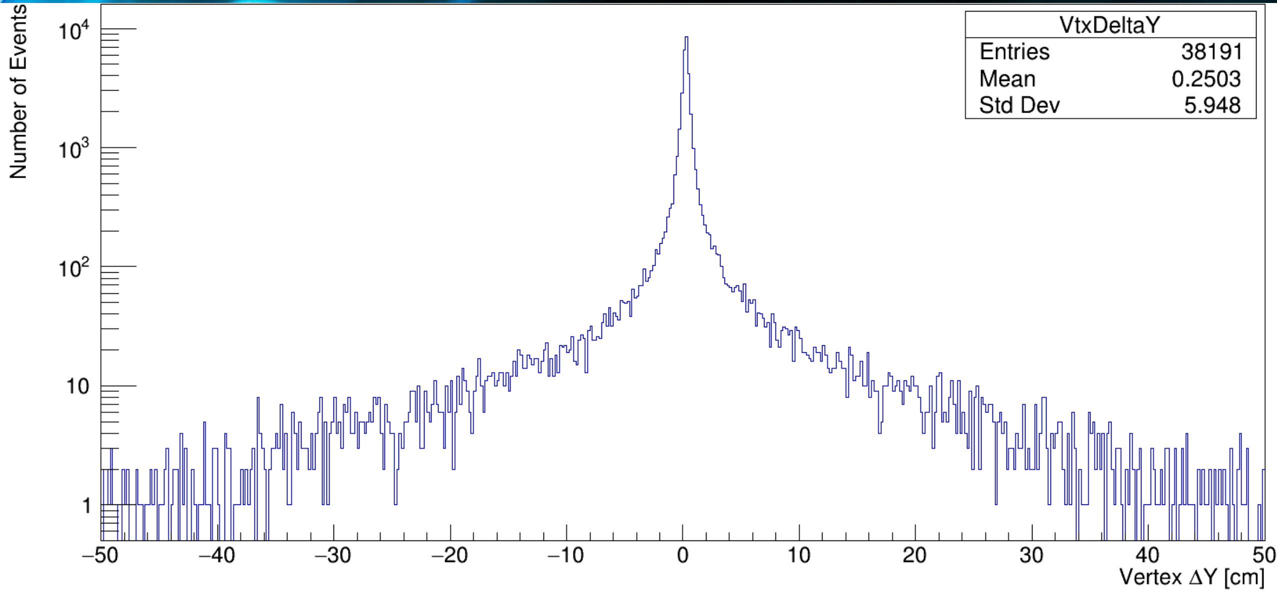
hchiSquared	
Entries	38191
Mean	0.2018
Std Dev	0.5118

(Reco-True) Vertex Positions Distributions. "nue" and "nu". All Interactions. Pure CNN Vertexing.



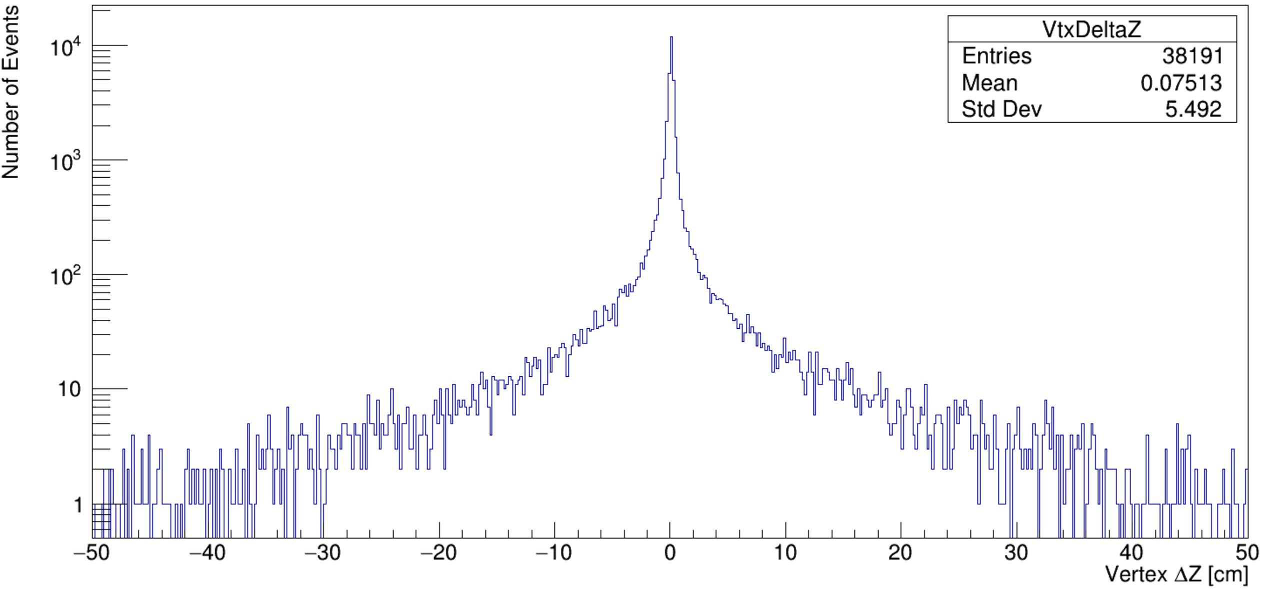
VtxDeltaX

Entries	38191
Mean	0.05269
Std Dev	4.233



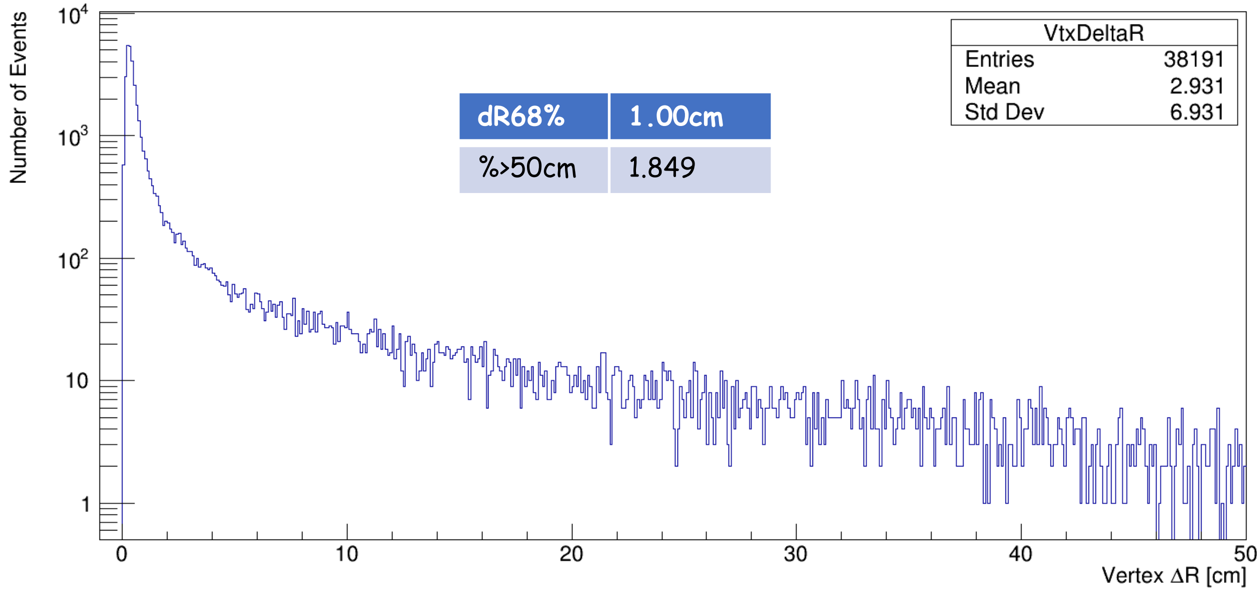
VtxDeltaY

Entries	38191
Mean	0.2503
Std Dev	5.948



VtxDeltaZ

Entries	38191
Mean	0.07513
Std Dev	5.492



VtxDeltaR

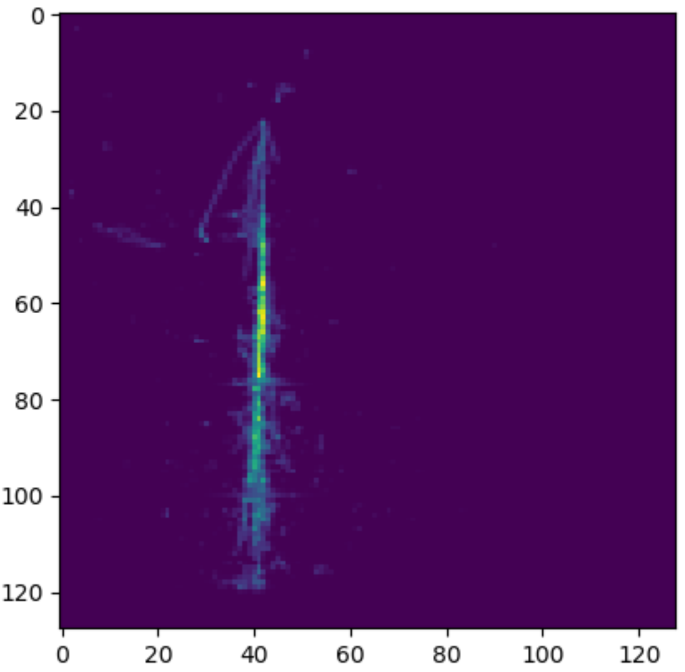
Entries	38191
Mean	2.931
Std Dev	6.931

dR68%	1.00cm
%>50cm	1.849

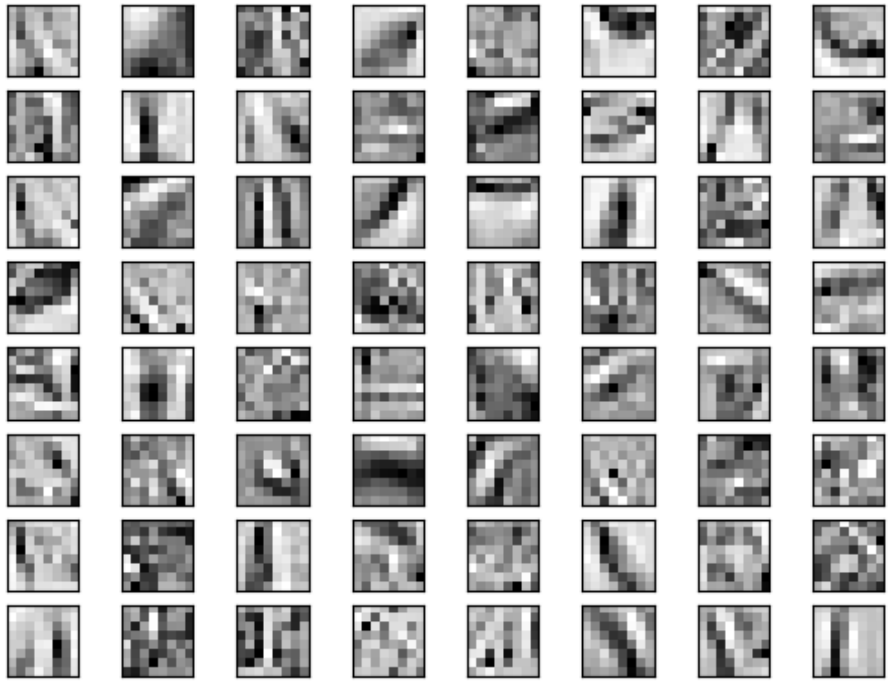
Note: These distributions created using only events which had at least 1 reconstructed cluster with at least 5 hits at the point that vertex reconstruction occurs.

CNN Visual Example (Part 1 of the Algorithm, W view)

Example Input Image



Example CNN Filter Plots (Conv Layer 1)



Example Output Of Conv Layer 1