Reconstructing 3D Charge Depositions in the MicroBooNE Liquid Argon Time Projection Chamber using Convolutional Neural Networks

MICROBOONE-NOTE-1082-PUB

MicroBooNE Collaboration*

We have developed a convolutional neural network (CNN) that reconstructs the three dimensional location of ionization produced by charge particles traversing a wire-readout liquid argon time projection chamber (LArTPC) at MicroBooNE. This method is the first CNN application to reconstruct 3D space points directly from two dimensional images of the different wire plane views.

I. INTRODUCTION

Liquid argon time projection chambers (LArTPCs) are the technology of choice for the MicroBooNE experiment [1], as well as other current and future experiments, because of their ability to scale to kiloton masses while maintaining precision tracking and calorimetry. One of the ways this scability is achieved is through the use of wire planes instrumented with charge sensitive electronics to detect the ionization left behind by charged particle tracks. Wire plane readout allows the number of electronics channels to scale with the linear dimension of the detector, significantly reducing the costs when compared to potential solutions directly capturing 2D position information (e.g. with "pixel"-based readouts). While there is promising R&D to perform pixel readout in a cost-effective manner, currently, wire planes are the most common readout approach used in current LArTPCs, and will be the technology used for the first DUNE far detector module.

The disadvantage to wire planes, however, is that they record tomographic projections of the charge locations. To recover the 3D information of the actual path of a particle through the detector, one must find a way to associate charge recorded on a wire at a given time to charge on a wire from a different plane. This is often a problem with large degeneracies.

* MicroBooNE_Info@fnal.gov

The problem is exacerbated for the case of LArTPCs operating near the surface, without shielding, where large numbers of cosmic ray trajectories are present (and overlapping in time) in an event.

In this work, we present the first solution to this problem using a deep convolutional neural network (CNN). The CNN is a class of machine learning algorithm that is capable of learning complicated patterns in an image in order to perform a wide variety of tasks. In the context of our application, the network can learn the 2D patterns needed to match regions across wire planes, thereby reducing the degeneracy of possible solutions when compared to using individual locations of deposited charge alone. We performed this work in the context of event reconstruction for the MicroBooNE detector, but the technique applies generally to all LArTPCs performing tomographic imaging.

II. INPUT IMAGE PREPARATION

The data provided to the network comes in the form of images which represent the amount and time of charge observed by wire planes in the LArTPC. In what follows, we briefly describe the MicroBooNE LArTPC. (For more details see [2].) A LArTPC consists of a homogeneous volume of liquid argon bounded by two vertical planes. The first is the cathode plane, typically consisting of a rectangular metal sheet, biased to a large negative potential. The second is the anode plane consisting of a series of wire planes whose wire orientation is offset by some angle with respect to the other wire planes. The anode planes are typically held near ground. The potential difference between the cathode and anode planes create a nearly uniform electric field. When charged particle traverse the volume between the cathode and anode, they create ionization electrons that drift towards the anode (and ions which drift toward the cathode). Each successive wire plane is more positively biased than the next in order to drift ionization past each plane before being collected on the last wire plane. When the ionization electrons approach the wires, they induce a current on the wires which is captured by current-sensitive readout circuits.

For the MicroBooNE LArTPC, the distance between the cathode and first anode plane is 256 cm. The cathode is biased at -70 kV creating a drift field of 273 V/cm. Ionization electrons travel at a drift velocity of $\sim$ 1.1 cm/microsecond, resulting in a maximum drift time of 2.33 ms. There are three anode planes. The plane furthest from the cathode where

charge is collected is referred to as the collection plane, Y. The wires on the Y plane are oriented vertically. The other two anode planes, referred to as induction planes U and V, have their wires oriented at +60 and −60 degrees with respect to the Y plane wires. Each plane has a parallel set of wires spaced by 3 mm. Because the charge drifts past the induction plane wires, the signals recorded on these planes are bipolar. The signal seen on the Y plane, which collects the charge, is unipolar.

The raw data from the MicroBooNE detector comes as digitized waveforms from a total of 8256 wires across all three planes. One "event" consists of 6048 charge samples for each wire taken at a rate of 2 MHz. The full readout window of MicrooBooNE is 4.8 ms, which corresponds to 9600 samples. However because of data size considerations the first 2400 and the last 1152 samples are truncated. Those fall outside of the beam spill window, and contain only cosmic-induced ionization tracks. The waveforms are recorded concurrently for all of the wires. Before forming the images to pass to the network, the raw waveforms pass through noise removal and signal processing algorithms, described in [3]. The important results of the signal processing are that (1) the electronics response is deconvolved from the waveform; (2) this converts the bipolar pulses on the induction planes into unipolar pulses; and (3) a threshold is applied and large portions of the waveform are set to zero in order to reduce low-frequency noise, which becomes amplified by the deconvolution of the bipolar induction field response. The post-processed waveforms are downsampled in time by a factor of 6. This is done for two reasons: (1) to lower the number of pixels (and thus image size) for the purpose of image processing, and (2) to achieve pixels roughly square in size, of 3 mm by 3 mm. However the filters in the signal processing leave the result oversampled by at least ×4. Thus the downsampling represents something much smaller than a ×6 loss of information. The result are three images, one for each plane, with wire × time dimensions of 2400 × 1008 for the two induction planes, U and V, and 3456 × 1008 for the collection plane, Y.

III. METHODS

In this section, we describe the three different components of the algorithm, namely a pre-processing algorithm followed by two neural networks. Figure 1 provides a diagram sketching the network architectures. The pre-processing step takes in the three wire vs.

time images, one for each plane, for an event and produces (1) a sparse representation of the data in a matrix format and (2) a list of candidate pixels combinations with one pixel from each plane. The sparse matrices are sent into the first CNN which is responsible for generating a 16-dimensional feature vector for each pixel. The weights of this CNN are shared between the three planes. After the feature vectors have been generated, the algorithm loops through each of the possible pixel combinations. For each combination, the feature vectors are retrieved and concatenated. The concatenated vector is passed into a classifier network which provides a score, indicating if the combination of pixels is good (score near 1.0) or bad (score near 0.0). After every combination is evaluated and provided a score, a post-processor saves a 3D space-point object.

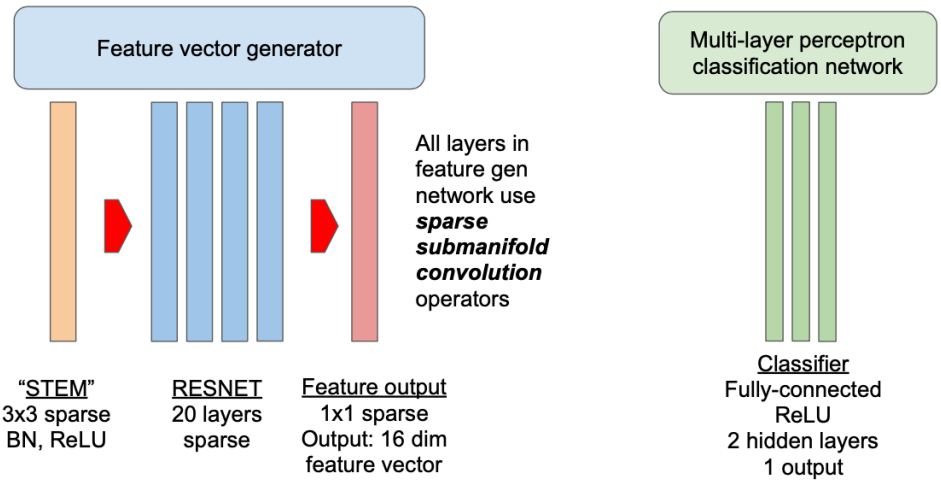


FIG. 1: Network diagram. There are two sub-networks. The first is a sparse CNN consisting of an initial processing block (or "stem"), main block of residual convolutional layers ("resnet"), and finally an output block. It takes in image data in the form of a sparse matrix and outputs a 16-dimensional feature vector for each pixel. This is done for each plane. The weights of the network are shared across all planes. The second network takes in a triplet of feature vectors made by concatenating one vector from each of the three planes. The triplet is then passed into a multi-layer perceptron (MLP) with 2 hidden layers and an output layer. For each triplet, the MLP produces a score between 0 and 1 representing the likelihood that the locations in the planes the triplet represents is correct.

A. Pre-processing: candidate pixel matches

For each event, the first step in pre-processing is to convert each of the three wire plane images into sparse matrices. Each matrix has a column for an index, time, wire, and pixel intensity in the image. An entry is filled in the sparse matrix from the image if the pixel intensity is above a threshold, which is greater than or equal to a value of 10.0. This threshold was chosen to remove remaining signal noise, especially in data images, and was found to have negligible effect on signal efficiency.

After the sparse matrices are formed, we then generate a list of index triplets, one from each plane, using knowledge about the wire locations. The triplets indicate what combination of pixels constitute a possible 3D space-point in the detector. The triplets are formed by first choosing a starting plane and a target plane, (e.g. Y to U). For each starting plane pixel, the tick (time sample) is used to get all pixels from the target plane corresponding to the same time slice. Then for each target plane pixel, we test if the wires corresponding to the columns of both planes intersect in a region inside the TPC. To save time, a look-up table is built ahead of time that lists for a given wire what wires in the other planes intersect with it. For each possible intersection, a 2D location is calculated. This location is then projected into the other plane not yet considered. (e.g. for Y to U intersections, we would project into the V plane). If there is a pixel at the same time within one wire pitch of the projected wire, then a triplet is formed.

This procedure is repeated for each (target plane,source plane) combination for a total of six times, corresponding to 3 planes and two directions of prediction for each plane. Only unique triplets are saved. No explicit multi-threading or vectorization (beyond what the compiler implements) is used.

A label must be associated to each triplet as to whether it is a true (positive) or false (negative) example when preparing the training data. We use truth information stored from the simulation to do this (see Section IV A). The truth information provides the true pixel matches between two planes. For each possible (source,target) plane pixel pair, we look up in the truth information if the proposed target pixel is within 3 pixels of the true target pixel. If at least one (source,target) pair satisfies this condition it is labeled as a correct (positive) example. The relaxed requirement, compared to an exact match requirement, is used to help training. This is based on the hypothesis that pixel associations a few columns

off from a true association will have similar topological features to the true locations. Asking the network to label exact triplets as correct while labeling close triplets as incorrect proved to yield lower training accuracy.

B. Network Architectures

Taking the sparse matrices generated as described in the previous section, the information is passed into two neural networks. The first is a CNN responsible for generating feature vectors for each pixel for all three planes. Figure 1 contains diagrams describing the architecture of this network. The network uses sparse submanifold convolution (SCN) layers [4] as a replacement for the familiar, dense convolution layers [5, 6].

The use of SCN layers is important for our application. First, LArTPC ionization activity is very naturally sparse. The typical image only has a pixel occupancy fraction of about 1% (estimate from data). This is an occupancy where sparse convolution operations are more efficient than dense operations (on a GPU). Furthermore, at inference time networks with SCNs are much more efficient than dense networks on CPUs.

The first part of the feature vector network is a 3×3 kernel SCN with Batch Normalization followed by a leaky ReLU [7] (rectifier linear unit) with 32 output channels. This is then followed by a succession of 10 residual modules [8]. The modules follow the design in [8], but replace the convolutional operations with SCN and the ReLU with leaky-ReLUs. Finally, the output of the residual modules is given to a final 1×1 kernel SCN layer with no ReLU on the output.

After feature vectors are generated for each pixel on all planes, the different pixel triplets are evaluated. For each triplet, the corresponding feature vectors are retrieved and concatenated into a 48-dimensional vector. This vector is sent into the second, classifier network. This network consists of two linear hidden layers with 32 units each. A leaky-ReLU is applied to the outputs of all the hidden layer units. This connects to a final output layer with one unit. The architecture of the network was loosely optimized to provide fast deployment, with the goal of ≤ 5 seconds for a forward+backward pass. The output is passed through a sigmoid function to restrict the range to a value between 0 and 1.0. The result of the classifier step is a score for each of the proposed pixel triplets.

C. Post-processing: 3D spacepoint formation

Each prepared triplet is associated to a 3D spacepoint. Post-processing simply amounts to storing this spacepoint along with information on the pixels and their integrated charge value. The classifier score is also saved.

IV. TRAINING

A. Data set preparation

The training data set was prepared using MicroBooNE's simulation and analysis framework. The model of the detector and propagation of particles through it is handled by GEANT4 [9]. The simulation of the wire response is performed by the Wire-Cell simulation code, common to several current LArTPC experiments. The waveforms go through the same noise-filtering and signal processing as the real data. Unresponsive wires are also included, based on a time-varying list of unresponsive wires in real data.

The simulation has the capability to associate 3D charge depositions with each sample of the digitized wire response waveforms. This allows us to store, in an image-like format, associations between pixels in two planes. For multiple charged particle tracks which produce ionization in the same waveform sample, the largest energy deposition in that bin gets to assign the pixel associations. These associations is what we use for the ground truth ("ground truth" is the true information used to compare reconstruction performance against).

The training data models cosmic rays passing through the detector and a neutrino interaction. Cosmic ray spatial, type, and momentum distributions are from the CORSIKA [10] cosmic particle generator. The GENIE [11] neutrino interaction generator is used to determine the final state particles for the neutrino interaction included in each event. The interactions simulated were for electron neutrinos from the Booster Neutrino Beam. A total of 40,000 events were used for the training sample. Another 9220 events were reserved for the validation sample.

B. Procedure

The network was trained using the Adam [12] optimizer implemented in PyTorch [13] (with momentum set to 0.9 and weight decay to 1.0×10^{-4}). The optimizer was directed to minimize a binary cross entropy loss function. An initial learning rate of 1.0×10^{-2} is used. The network was trained for 150,000 iterations after which the learning rate was reduced to 1.0×10^{-4} and trained for another 150,000.

For each iteration, the network was given 50,000 examples (triplets) from one event image.

When calculating the loss for each iteration, the number of positive and negative examples need to be balanced. This is done by splitting up the loss function for positive and negative examples and then combining the two into a total loss which is weighted by the relative number of positive and negative examples.

The loss as a function of training iteration is shown for both the training and validation data sets in Figure 2 (left). The validation loss was calculated every 100 training iterations. Besides comparing the loss, the accuracy for both positive and negative examples were compared for the training and validation set. The accuracy versus training iteration is shown in Figure 2 (right).

The training was stopped after 3.75 epochs, since accuracies in both training and validation samples plateaued. It should be noted that we saw no signs of overtraining since the network was provided with a vast enough number of unique samples (each image contains O(100k) triplets). Because both the loss and accuracy showed no significant deviation between the training and validation set, the final model uses weights taken from the last training iteration.

V. RESULTS

A. Comparison to 2-plane matches

The network discussed in this note uses feature vectors for all three planes to make a judgment. A previous iterations of the network used feature vectors for a pair of wire planes. We do not discuss the 2-plane version here in detail, but only wish to compare qualitatively the results for tracks that are vertical in the detector. Such tracks are nearly parallel to the wire planes and, therefore, most of the charge is recorded by the wire planes at nearly the

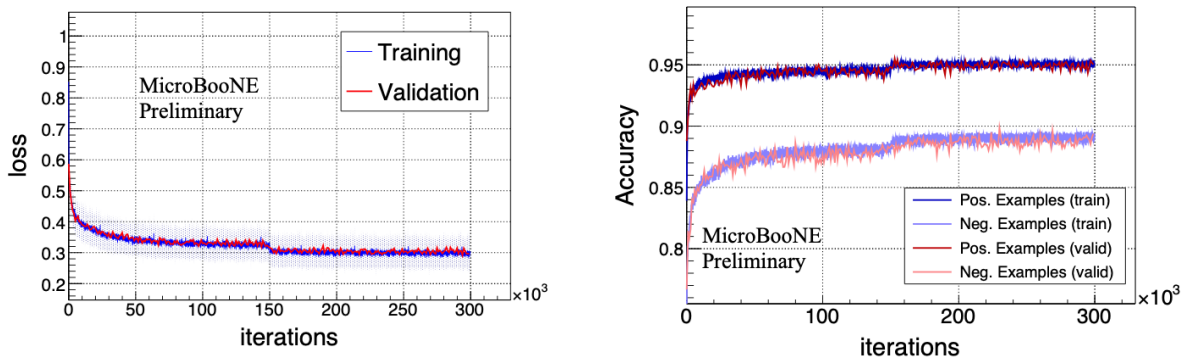


FIG. 2: Left: Loss as a function of training iteration both for the training and validation samples. The error band is the standard deviation in the loss for the training sample over 100 iterations. Right: Accuracy as a function of training iteration. Shown are curves for positive (dark lines) and negative (light lines) examples for both the training (blue) and validation (red) set. The dip at 150k iterations corresponds to change of learning rate.

same time. These "isochronous" tracks are highly degenerate and one of the tougher cases to handle. By using a straight line hypothesis for the points, the large degeneracy is improved, but still two lines are possible. By using a line hypothesis and three planes, this degeneracy is broken. This is an example where recognizing a topological pattern – something a CNN does well – resolves degeneracy.

Figure 3 provides diagrams and example output from a two- and three-plane network. In Figure 3(a), a cartoon shows an example of an isochronous track in the wire plane image. Using charge alone, the possible points consistent with these images form a 2D trapezoidal region. However, with a topological constraint, such as a single straight track, this degeneracy can be reduced to two possible lines. By assuming a particular particle with an dq/dx, one could choose between the two tracks. However, if we do not use such a hypothesis, we need the third plane to break the degeneracy as shown in Figure 3(b). The length of the segment in the third image allows us to choose which of the two tracks is the correct one. This is an example where topology (and calorimetry) allows one to greatly improve the choice of true 3D space points. These are the kinds of features which CNNs capture well.

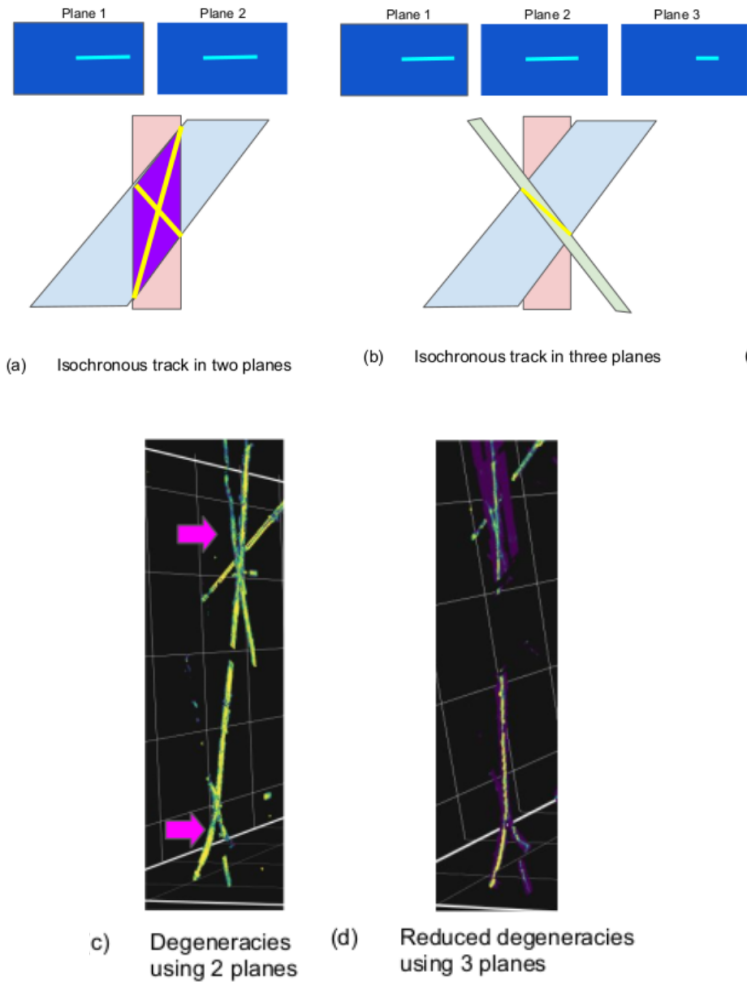


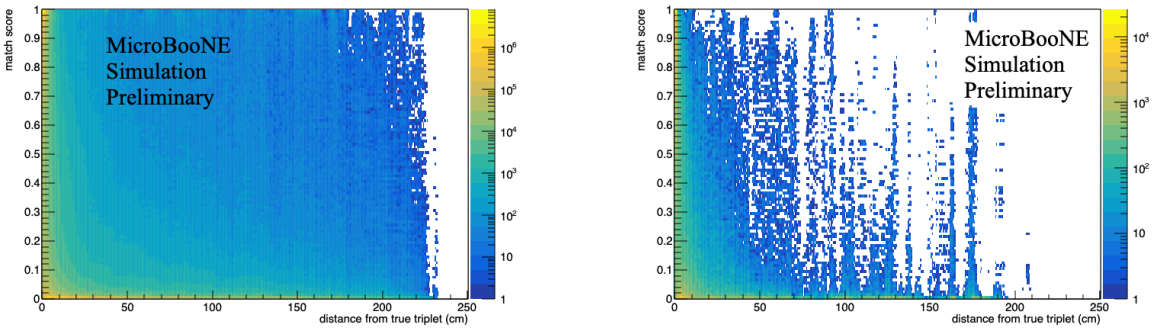
FIG. 3: Illustration showing how information from three planes improves isochronous tracks. (a) Two possible straight line trajectories (yellow) are possible when an isochronous tracks is observed in two planes. (b) Degeneracy is broken when using three planes. (c) Example 3D space point output for two planes. High confidence points are yellow. Middle confidence points are green. Low confidence points are purple. Magenta arrows point to two regions where isochronous region causes degeneracy. (d) Example 3D space point output for three planes. Color scale same as in (c). Majority of points follow single trajectory.

B. Performance validation with MC

We use CORSIKA + BNB unoscillated electron neutrino MC to validate the network. The sample covers the full detector volume in terms of position and angular acceptance.

Each event contains a single BNB electron neutrino interaction and order of 20 cosmic rays. Thus the sample is dominated by downwards going cosmic muons, as well as neutrino-induced electron showers and proton tracks along the beam direction (horizontal). Below we discuss some basic metrics of network performance.

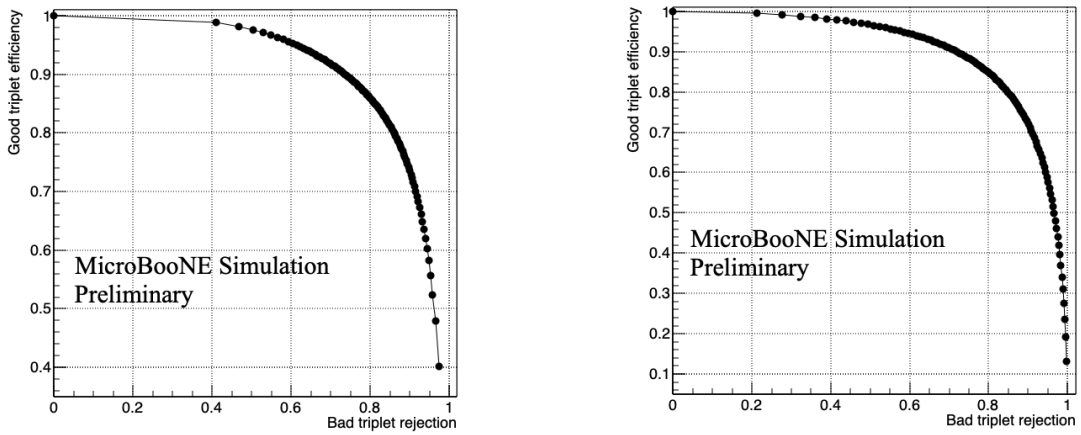
Network output (match score) is plotted vs distance from true triplet for all evaluated



(a) Tracks (b) Showers

FIG. 4: Network score vs. distance from true triplet for all geometrically possible triplet configurations.

matches in Figure 4, for tracks and showers respectively. As expected, it can be seen that the worst 3D position reconstruction resolution (largest distance from truth) comes from triplets with a low network score. Figure 5 shows the efficiency for keeping good triplets versus bad triplet rejection as a function of match score for all evaluated triplets, both in tracks and showers. Each point on the graphs corresponds to a match score output increment of 0.01. Here a triplet is considered good if it is within 1 cm from the true triplet, and bad if it is further away. These curves prove that match score (or in simple terms how certain the network is that a reconstructed triplet is good) correlates with true reconstruction goodness (as measured by distance to true point).

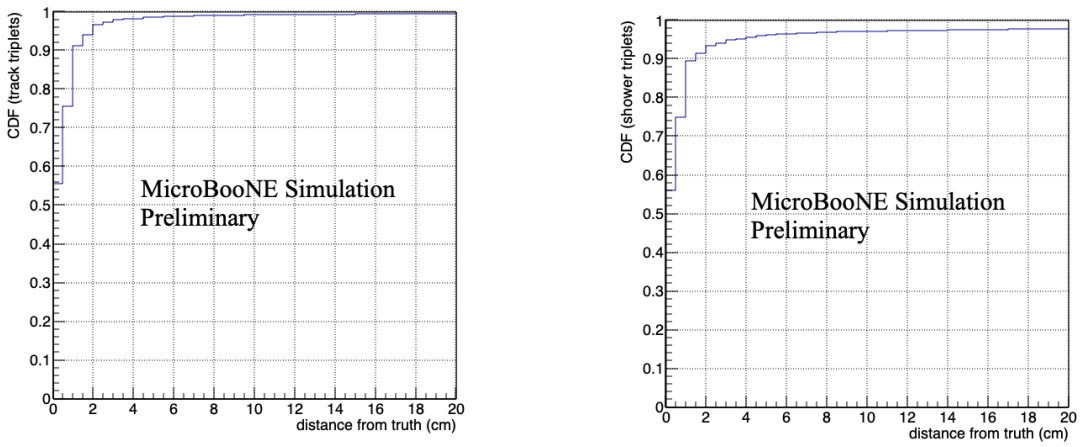


(a) Tracks (b) Showers

FIG. 5: Fraction of good triplets kept vs. bad triplets rejected as a function of network score for tracks and showers. Good triplet defined as within 1 cm from truth.

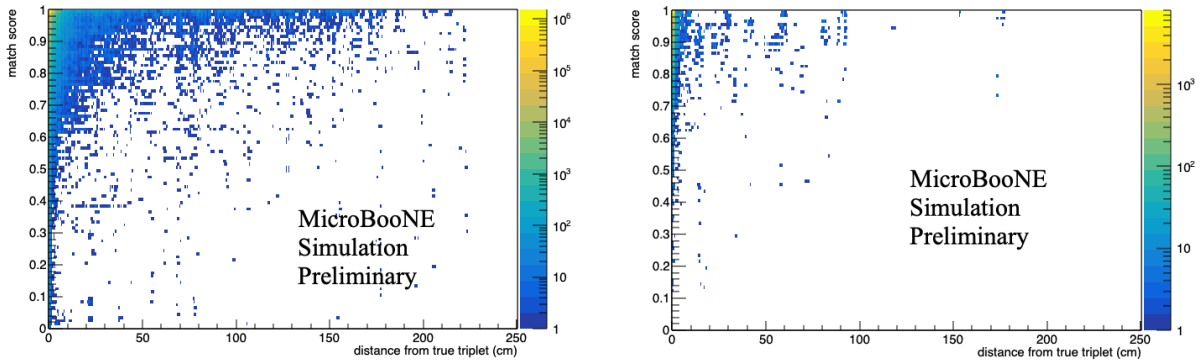
It should be noted that while the networks provides multiple predictions for each 3D space-point, in practice one can use only the best prediction for each point (meaning the reconstructed space-point with highest match score output). Figure 6 shows the cumulative fraction of all evaluated best-match triplets (cumulative distribution function, or CDF for short) as a function of 3D distance from true triplet. One can see that $\sim 90\%$ of all triplets are within 1 cm from truth for both tracks and showers, with slightly better performance for tracks. 3D distance from truth versus match score for best-match triplets only is shown in Figure 7. Since we have selected the best-match triplets for every point, even points very far from the truth have generally high scores, which explains the difference in shape between the distributions for all triplets and best-match triplets. This points to difficult network training examples that we would like to explore for further improvements.

Another important aspect of evaluating the network is checking whether its performance is robust to angular dependence. Figure 8 shows distance from truth versus azimuth (in detector xy plane) angle of the track/shower, for all generated triplets, and for best score ones. It should be noted that the big excess of down-going tracks come from the fact the majority of tracks in the validation sample are cosmic rays. We plot the distance to true triplet for best-match triplets belonging to tracks with azimuth angle in the range $[-1.64, -1.35]$ radian in Figure 9. It is clear that best-match candidates are robust (close



(a) Tracks (b) Showers

FIG. 6: Performance of network for best-match triplets. Cumulative fraction of triplets (CDF) vs. 3D distance from truth.

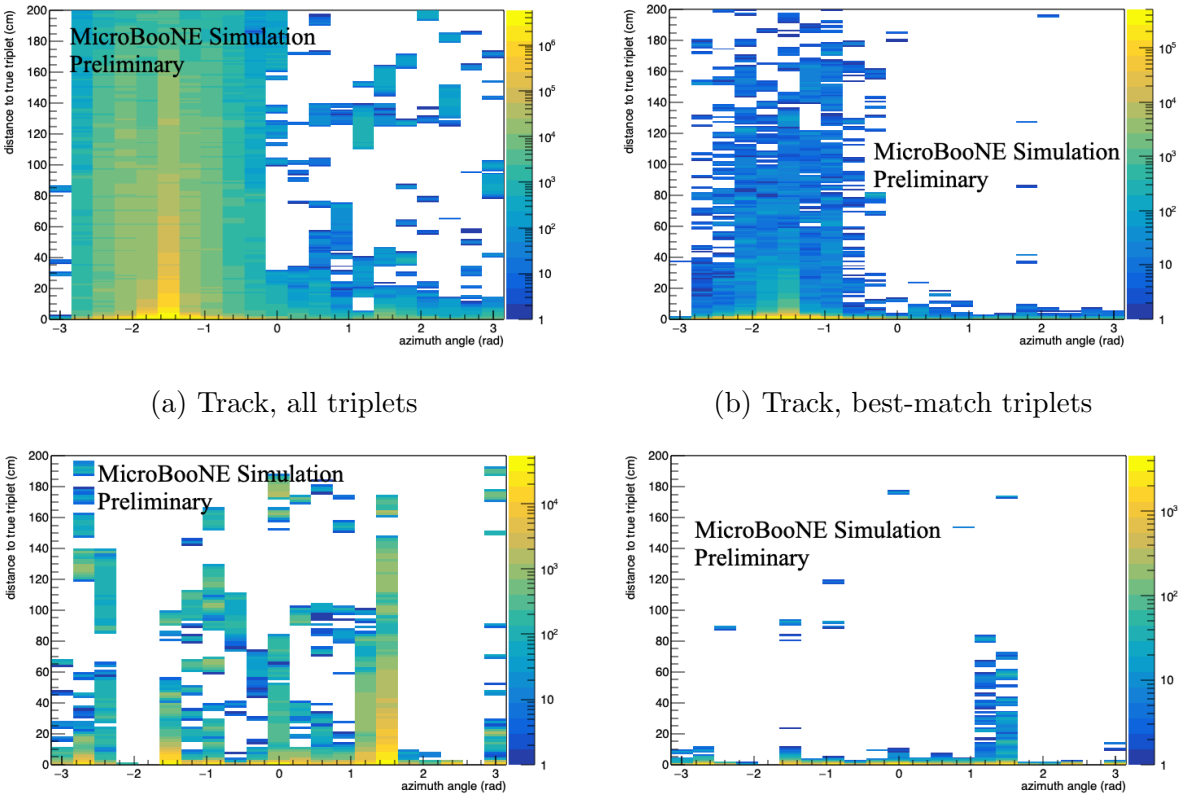


(a) Tracks (b) Showers

FIG. 7: 3D distance from truth as a function of network score for best-match triplets.

to truth) even for vertical tracks. This speaks in favor of the 3-plane network being able to resolve degeneracies and reconstruct vertical tracks. Figure 10 shows the performance versus polar angle (with respect to beam direction). Again, best-match triplet distributions are rather flat.

In an attempt to estimate the performance in terms of how it will affect higher-level track and shower reconstruction, we change our metric from distance to true triplet to distance of reconstructed triplet to track or shower axis. We define track/shower axis in a simplistic way, as the vector between the end and the start point of the MC track or shower object.



(a) Track, all triplets

(b) Track, best-match triplets

(c) Shower, all triplets

(d) Shower, best-match triplets

FIG. 8: 3D distance from truth vs track/shower azimuth angle.

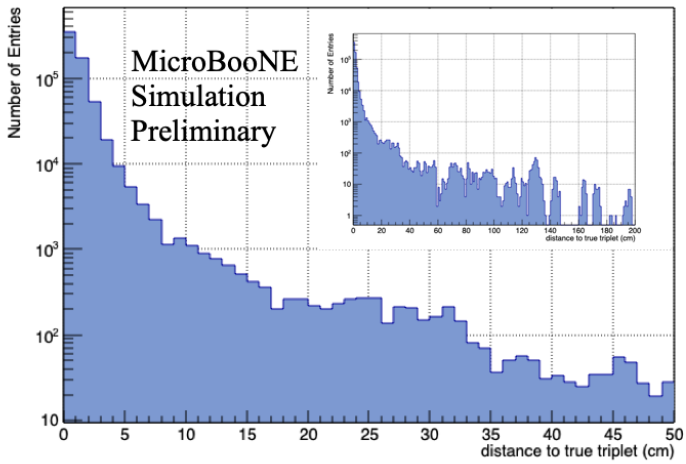
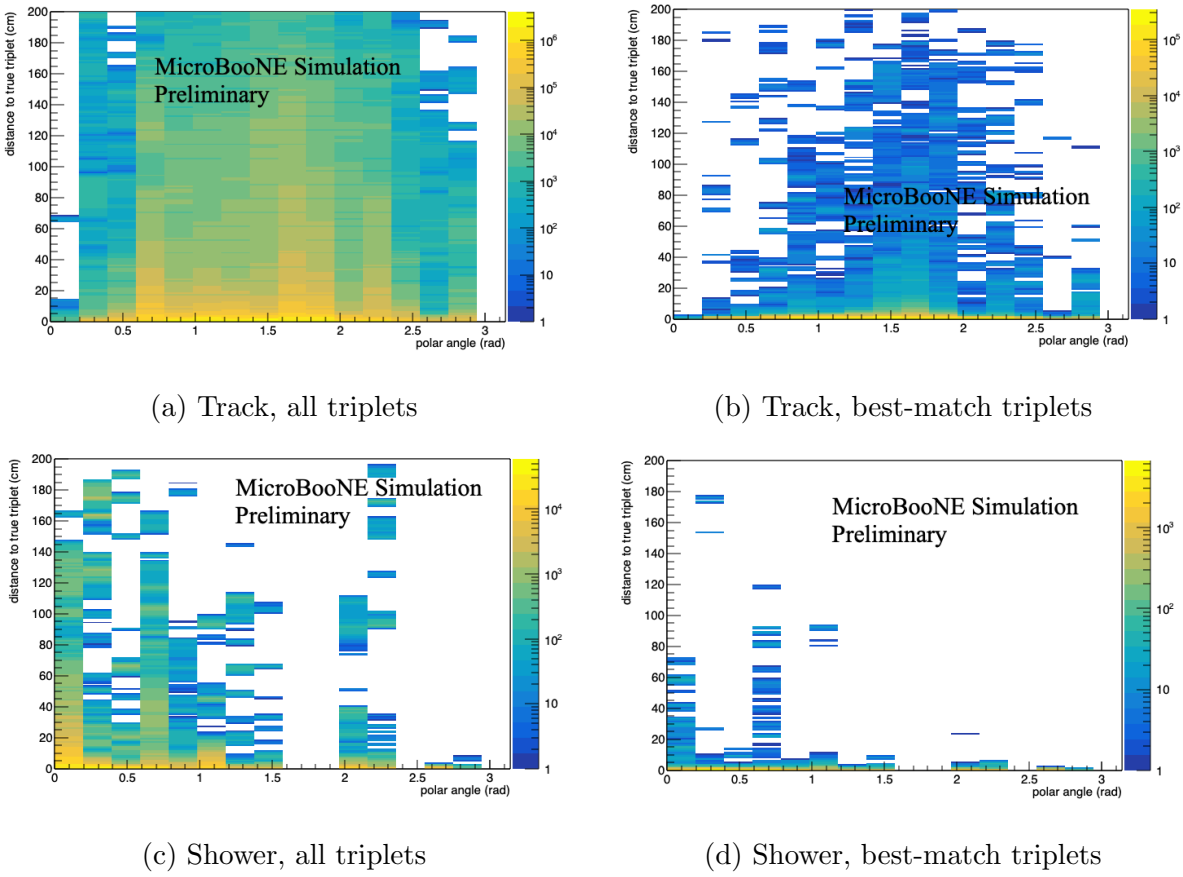


FIG. 9: Distance from true triplet for vertical tracks (azimuth angle $\in [-1.64, -1.35]$ rad).

Below we only consider neutrino-induced tracks and showers, since they are the main object of interest in MicroBooNE. In Figure 11 we show the cumulative fraction of reconstructed



(a) Track, all triplets

(b) Track, best-match triplets

(c) Shower, all triplets

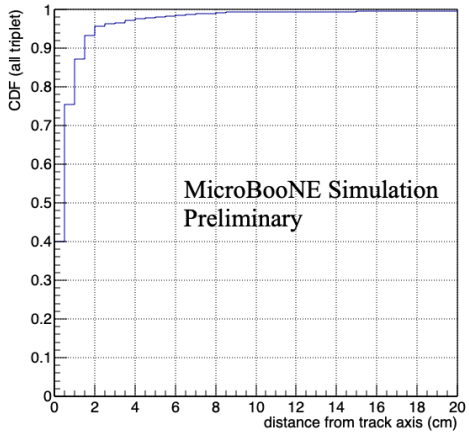
(d) Shower, best-match triplets

FIG. 10: 3D distance from truth vs. track/shower polar angle.

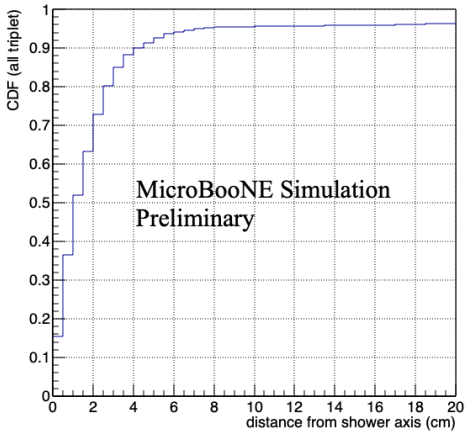
best-match triplets as a function of radial distance to track/shower axis. In Figure 12 we show the same, but for good triplets only (within 1 cm from truth). It can be seen that if one integrates all triplets in a 5 cm radius from a track or shower axis, one collects $\geq 99\%$ of good track space points, and $\geq 95\%$ of good shower space points.

C. Run time estimates

An important requirement for the algorithm in order to be applicable to large-scale LArTPC datasets, is fast inference. We measured the processing time per event on jobs running on worker nodes which are a part of the Tufts High Performance Computing cluster. The nodes used employ Intel(R) Xeon(R) CPU E5-2660 v2 @ 2.20GHz. Each worker job was assigned one core. We gathered data for 40 events which are summarized in Table I. The measurements do not include overhead for IO of the file to the worker nodes or steps

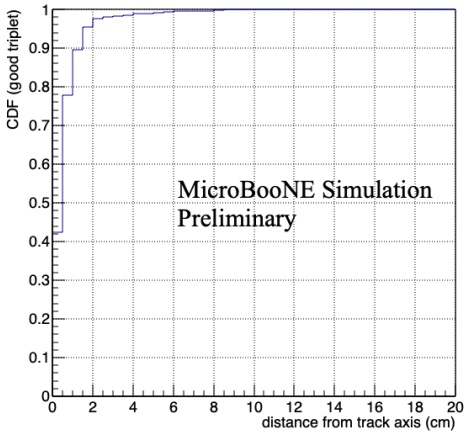


(a) Tracks

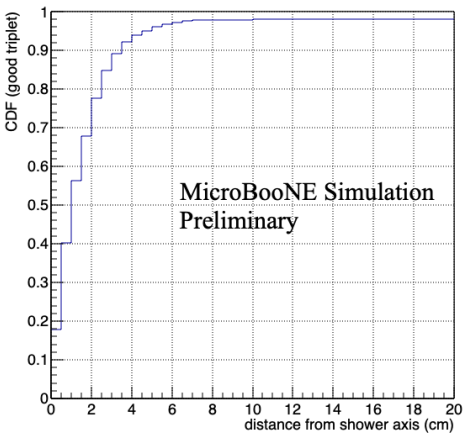


(b) Showers

FIG. 11: Cumulative fraction of reconstructed triplets as a function of distance from track/shower axis. Neutrino induced primary track/shower only, best-match triplets.



(a) Tracks



(b) Showers

FIG. 12: Cumulative fraction of good (within 1 cm from truth) reconstructed triplets as a function of distance from track/shower axis. Neutrino induced primary track/shower only, best-match triplets.

like loading the network at the beginning of jobs. The average total time per event is 34.8 seconds which is in line with other reconstruction steps currently employed by MicroBooNE.

In addition to measuring the mean time, we also checked how the run time scales with the number of space point proposals for a given event. The number of proposals scales with

Step	Time per event (sec)
Read from disk+spacepoint proposals	13.5
Network inference	16.0
Post-processing+saving to disk	5.3
Total	34.8

TABLE I: Run time statistics for worker jobs deploying the LArMatch network on the Tufts cluster.

the number of particle tracks in the image. It is also strongly effected by the presence of noise features, in particular ones that produce above threshold pixel values all at the same time tick. We find that the run time scales with the number of proposed points as expected with 0.97 secs per event per ten thousand proposed space points. The average number of proposed space points in a typical CORSIKA+BNB simulated event is 380K.

VI. EXAMPLE EVENT DISPLAYS

A. All proposed points with score

In Figure 13 we show an example of the network output on a BNB simulated neutrino interaction overlayed with cosmic rays from data. The color of the points represents network score: dark blue is low score (low confidence), bright red is high score. MicroBooNE TPC outline plotted as a mesh. Red circles represent the 32 photomultiplier tubes (PMTs) installed behind the anode plane in the MicroBooNE detector. Color of the circles corresponds to intensity of observed light by each PMT. All coordinates are in detector coordinate system, in centimeter units.

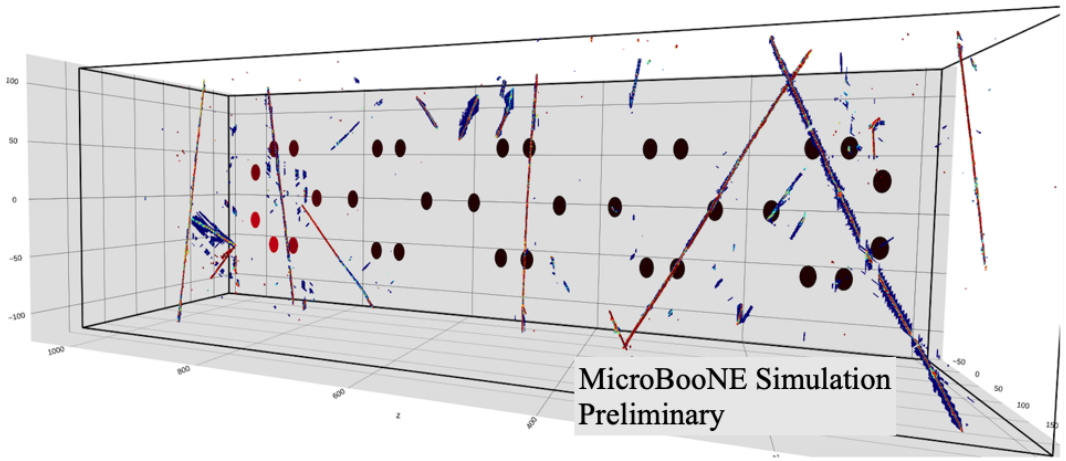


FIG. 13: Example output for one event. This is an overlay event where a simulated ν_e interaction is overlaid into off-beam data. The color score represents the network score. Bright red indicates a score near 1.0. Dark blue indicates a score near 0.0. Portions of trajectories with only low-confidence points generally correspond to unresponsive wires in one of the wire planes, where geometrically consistent charge was collected only on two planes. Since every geometrically possible triplet is evaluated, there often exist clouds of low-confidence points around the edges of trajectories; these correspond to ghost (fake) points.

B. Example output on BNB events with Neutrinos

Figures 14 through 16 demonstrate the output of the network on real BNB events. The events are selected by using tools from the current DL LEE 1e1p selection [14]. These events are a subset of events selected by requiring a 1e1p BDT score between 0.5 and 0.7. The intention is to provide examples with both cosmic muons and neutrino interactions that are likely to include showers. Events in this region of the 1e1p BDT score are predominantly CC and NC events with a π^0.

VII. FUTURE WORK

In this note, we have successfully demonstrated the first CNN to produce 3D space points directly from 2D image data in LArTPCs.

In the near future we hope to demonstrate that the network, trained with simulated

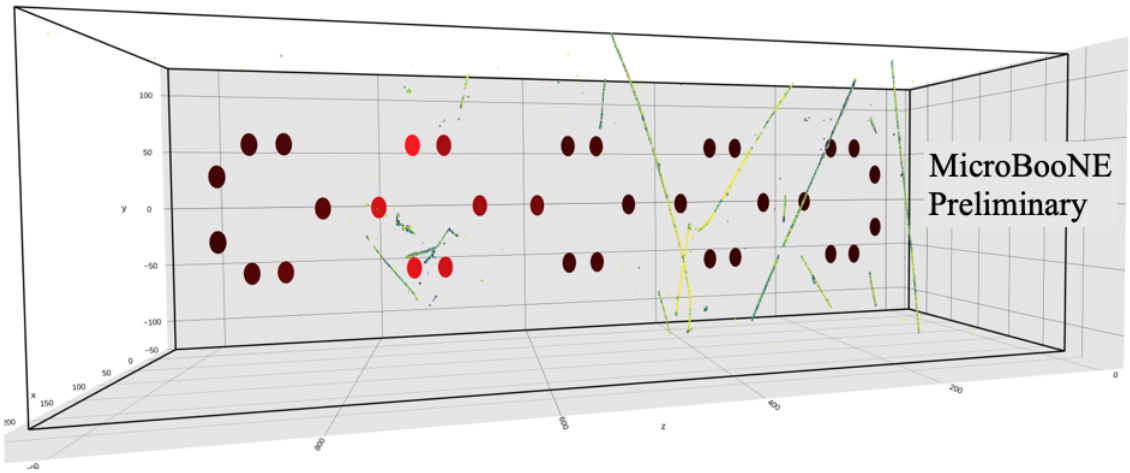


FIG. 14: BNB event with 1e1p selection BDT score outside the signal region. This event was chosen as it shows a multi-particle neutrino interaction with the presence of two low-energy showers. Note that one of the showers is hard to distinguish as such due to the viewing angle here. The color scale represents the match score. Points are shown which have a match score between 0.5 to 1.0. The neutrino beam travels from right to left.

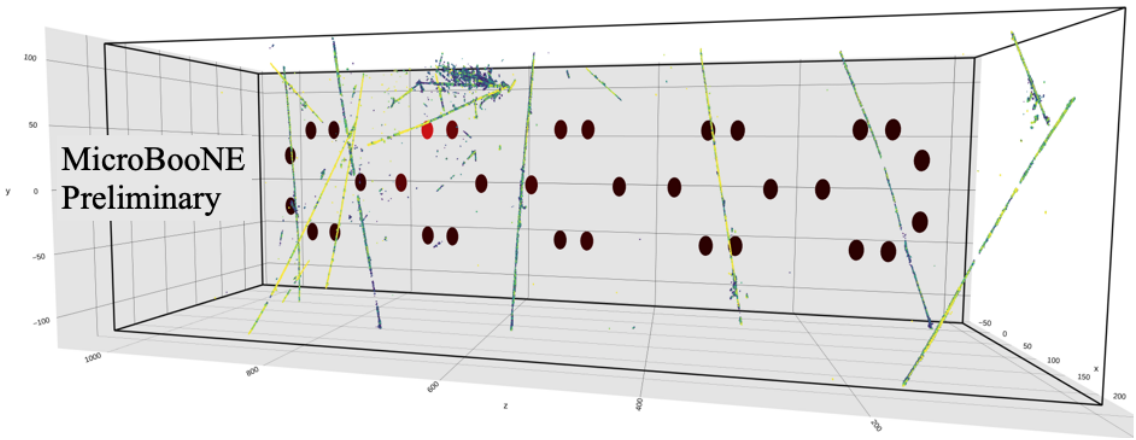


FIG. 15: BNB event with 1e1p selection BDT score outside the signal region. This event was chosen as it shows very complicated high-energy multi-particle neutrino interaction with the presence of a high energy shower. The color scale represents the match score. Points are shown which have a match score between 0.5 to 1.0. The neutrino beam travels from right to left.

images, performs similarly on real data to within systematic uncertainties. To do this we plan to use cosmic rays tagged by the MicroBooNE Cosmic Ray Tagger [15]. We plan to

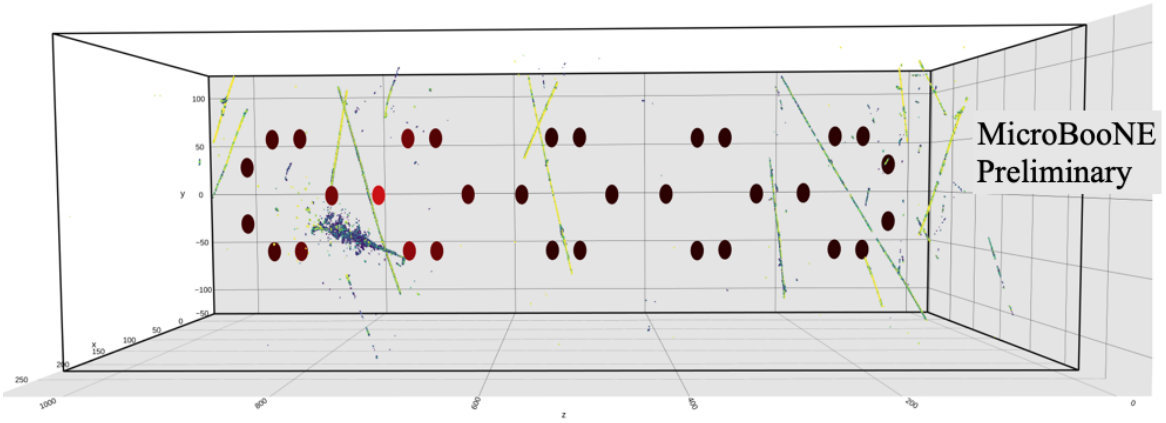


FIG. 16: BNB event with 1e1p selection BDT score outside the signal region. This event was chosen as it shows a clean CC electron neutrino-like candidate with a high-energy shower. The color scale represents the match score. Points are shown which have a match score between 0.5 to 1.0. The neutrino beam travels from right to left.

estimate the effect of signal noise and signal processing on the network performance by using dedicated simulation. Future work also includes estimating systematic uncertainties on the network performance with different MC samples with varied detector properties.

[1] H. Chen *et al.*, *A Proposal for a New Experiment using the Booster and NuMI nNeutrino Beamlines: MicroBooNE*, Tech. Rep. (2007).

[2] R. Acciarri, C. Adams, R. An, A. Aparicio, S. Aponte, J. Asaadi, M. Auger, N. Ayoub, L. Bagby, B. Baller, *et al.*, Journal of Instrumentation **12**, P02017 (2017).

[3] C. Adams, R. An, J. Anthony, J. Asaadi, M. Auger, L. Bagby, S. Balasubramanian, B. Baller, C. Barnes, G. Barr, *et al.*, Journal of Instrumentation **13**, P07006 (2018).

[4] B. Graham and L. van der Maaten, arXiv preprint arXiv:1706.01307 (2017).

[5] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, Neural computation **1**, 541 (1989).

[6] K. Fukushima, Biological cybernetics **36**, 193 (1980).

[7] K. He, X. Zhang, S. Ren, and J. Sun, in *Proceedings of the IEEE international conference on computer vision* (2015) pp. 1026–1034.

[8] K. He, X. Zhang, S. Ren, and J. Sun, in *Proceedings of the IEEE conference on computer*

vision and pattern recognition (2016) pp. 770–778.

[9] S. Acostinelli *et al.*, Nuclear Instruments and Methods in Physics Research Section A **506**, 250 (2003).

[10] D. Heck, G. Schatz, J. Knapp, T. Thouw, and J. Capdevielle, *CORSIKA: A Monte Carlo code to simulate extensive air showers*, Tech. Rep. (1998).

[11] C. Andreopoulos, C. Barry, S. Dytman, H. Gallagher, T. Golan, R. Hatcher, J. Perdue, and J. Yarba, arXiv preprint arXiv:1510.05494v1 (2015).

[12] D. P. Kingma and J. Ba, arXiv preprint arXiv:1412.6980 (2014).

[13] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, *et al.*, in *Advances in Neural Information Processing Systems* (2019) pp. 8024–8035.

[14] MicroBooNE, *Internal Note: Low Energy Excess Search using DL Tools*, Tech. Rep. (2020).

[15] C. Adams *et al.*, Journal of Instrumentation **14**, P04004 (2019).