Cosmic Muon Clustering for the MicroBooNE Liquid Argon Time Projection Chambers Using sMask-RCNN

MICROBOONE-NOTE-1081-PUB

The MicroBooNE Collaboration

MicroBooNE_Info@fnal.gov

We develop an incarnation of Mask-RCNN, or Mask Regions with Convolutional Neural Net features, based on Facebook's 'Detectron' framework, to use the network for cosmic muon tagging. Our implementation of this network, sMask-RCNN (Sparse Mask-RCNN) is compared to the original dense version along several metrics. Both networks are trained to look at wire signal readout images from the MicroBooNE Liquid Argon Time Projection Chamber (LArTPC), and output a varying number of particle interactions within the image. Each object is classified between either coming from a neutrino interaction, or a cosmic muon, given a bounding box and an individual segmentation mask. sMask-RCNN has been modified from the original 'Detectron' structure to include a Sparse Convolution ResNet leading to a 95% reduction in ResNet processing time. However in order to benefit from the feature-finding ability of pretrained versions of dense ResNet, we translate dense ResNet weights trained on ImageNet into our sparse version. We find that sMask-RCNN has an average pixel clustering efficiency of 85.9% compared to the dense network's average pixel efficiency of 89.1%. While sMask-RCNN is primarily intended to be used for cosmic finding within the MicroBooNE detector, the network could be adjusted in the future to identify different interaction types in different detectors. The network may further be useful to other networks employed in MicroBooNE and LArTPC particle physics by providing a sparse ResNet pretrained to find particle interaction features. Code has been made available at: https://github.com/NuTufts/Detectron.pytorch

I. INTRODUCTION

One of the challenges faced by the MicroBooNE Collaboration [1] is dealing with the high ratio of cosmic muons crossing through the Liquid Argon Time Projection Chamber (LArTPC) relative to the number of neutrino events that interact inside the detector. The MicroBooNE LArTPC is a surface level detector [2] located on Fermilab's Booster Neutrino Beamline, and therefore does not use an underground location as cosmic shielding. In order to measure different neutrino interaction channels with high purity, techniques need to be developed to identify charge deposited in the detector due to cosmic muons and separate it from charge deposited due to neutrino interactions.

We present an approach to 2D interaction clustering using two versions of a machine learning network called Mask-RCNN (Mask Regions with Convolutional Neural Networks). We compare two versions of the network, the first is the original Mask-RCNN which is an instantiation of Facebook's 'Detectron' network [3] except instead of finding animals, people and objects, our network is trained to find and label particle interactions. The second network is our changed version of the Mask-RCNN we term sMask-RCNN for "sparse Mask-RCNN" which features a sparse submanifold convolutional ResNet in place of the original dense ResNet. The change from dense convolutions to sparse convolutions in the ResNet was made to decrease image processing time so that sMask-RCNN can be run efficiently on CPU computing clusters, rather than requiring GPU banks. In this way we ensure that sMask-RCNN can practically process the MicroBooNE dataset and act as a cosmic tagger for future Deep-Learning-based reconstruction in MicroBooNE [4].

We compare the two implementations of the network across several metrics throughout this note and have measured their effectiveness at properly finding, clustering, and labeling neutrino and cosmic muon interactions. Some neutrino interaction channels feature topologies different from that of an incoming cosmic muon. For example, a charged current quasi-elastic (CCQE) interaction with 1 lepton and 1 proton in the final state will typically feature a v-shaped pattern with the vertex placed on the neutrino interaction, and two branches extending outward: one proton track, and either a muon track or electron shower, based on neutrino flavor. In comparison a cosmic muon will appear as a long, nearly straight line, with a Bragg peak if the muon stops inside the detector, and potentially a short Michel decay coming off the end of the track.

We expect the pattern spotting machine learning techniques leveraged by our two implementations, Mask-RCNN and sMask-RCNN to be able to capitalize on this topological difference and learn to locate, identify, and cluster together charge associated with the difference interactions.

II. METHODS

A. Dense Mask-RCNN

The Mask-RCNN Detectron network structure is as follows: A Residual Network (ResNet), a Region Proposal Network (RPN), a classification layer, and a fully convolutional network (FCN) that we term the Maskifier to provide the masks. Each mask is a cluster of pixels within a found interaction. However pixels are allowed to be clustered into multiple different masks, if two overlapping proposed interactions both claim the same pixel. Figure 1 shows the network architecture. The input image feeds into the ResNet, which creates a feature map. That feature map then gets fed into the RPN, which proposes a number of potential bounding boxes for interactions in the image. A bounding box is meant to be the smallest rectangular box that contains all of the pixels in a given interaction cluster. Bounding boxes are defined by an x0, y0, a height, and a width. The RPN is free to propose as many boxes as it wants up to some maximum parameter. This parameter was not optimized by us, instead we take the configuration from Facebook's Mask-RCNN. These bounding boxes have their coordinates in the original image space, but an algorithm Region of Interest Align (ROIAlign) is responsible for snapping these boxes into the feature map space as well.

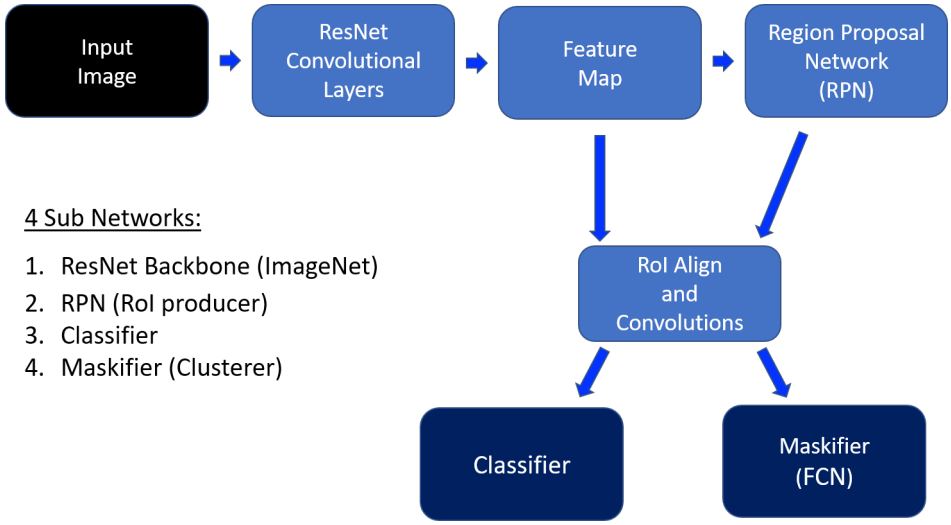

FIG. 1. Network Architecture

This suite of bounding boxes, along with the feature map goes through an alignment algorithm Region of Interest Align (ROIAlign) to align the features in feature map space and image space. Then the features and bounding box proposals are fed to a standard classification layer, which acts as our Classifier, and the Maskifier FCN in parallel. The Classifier is in charge of determining which interaction class is contained in the bounding box, and with what

classification score score. The Maskifier is in charge of creating a highlighted masked cluster of binary on/off pixels within the box. The Maskifier creates a different mask for each potential class choice, then gives the final one based on the result of the Classification layer. We use this as our pixel level cluster of the interaction.

The binary on/off mask is specific to a given bounding box. As such, if two bounding boxes overlap, as is often the case, their individual segmentation masks are both free to cluster the same pixel. This means that the network could cluster the same pixel in multiple masks. The network is trained in the same way as Facebook's Detectron, the entire network performs a forward pass on an image, and then the backward pass updates the weights of all the subnetworks based on a combined loss built from the outputs of the Maskifier, Classifier, and RPN, as described in the original Mask-RCNN paper [3].

Figure 2 shows the output of our network on a collection plane signal image shown in Fig. 3. We show the output masks, boxes and clusters from our network overlaid on the input signal image in Fig. 4. A cluster or 'mask' is defined as the group of pixels masked or grouped together by the Maskifier portion of the network. The input image to our network is a wire signal image from the collection plane that has undergone noise filtering and 2D deconvolution [5] [6], further, we threshold the image by setting any pixel with ADC value below 10 equal to zero. We find that this decreases the amount of noise in the image while maintaining interaction level features. The event shown is taken from a sample of neutrino beam data and was selected by the deep learning low energy excess analysis [7] as a candidate electron neutrino interaction with one electron and one proton in the final state. Figure 5 shows a data event recorded outside a neutrino beam spill, with network boxes, classes, scores and masks overlaid on the signal image. Since this is an off beam event, we expect everything inside the image to be cosmic background. However the network sometimes incorrectly classifies these cosmic muons as neutrino interactions. This would lead to a nonzero background being passed through a Mask-RCNN being used as a cosmic tagger.

Our dense ResNet Mask-RCNN version of the network was trained on an earlier version of MicroBooNE simulated data compared to the version used for the sparse ResNet 'sMask-RCNN' version of the network. The input images are pixels arranged in 3456 columns by 1008 rows. Each pixel's column corresponds to a specific wire in the LArTPC, and the row corresponds to a specific range of time that wire's signal was read out. The simulation uses CORSIKA [8] for the cosmic simulation. In the 2d event displays shown here, each the distance between column wires is 0.3 cm, which is plotted on the x axis; the time window of each row is 3.0 μs which is plotted on the y axis. The readout time of 3.0 μs is chosen as the time needed for electrons to drift 0.3 cm in the detector, matching the wire pitch. For the purposes of this note we use the collection plane in the LArTPC for all studies [2].

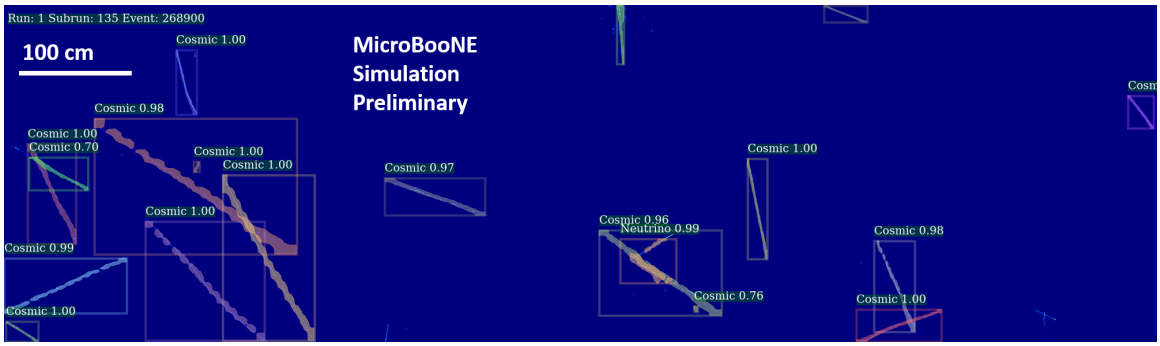

FIG. 2. An example labeled sparse ResNet sMask-RCNN output event display for MicroBooNE's LArTPC collection plane. This is an electron neutrino simulated event. The neutrino, located about halfway across the image in the bottom half is correctly masked by the network. Each interaction's bounding box and mask are shown with the same color, allowing the viewer to match box to mask and distinguish separate interactions. For this image, a classification score threshold of 0.7 was applied to make the network output less cluttered.

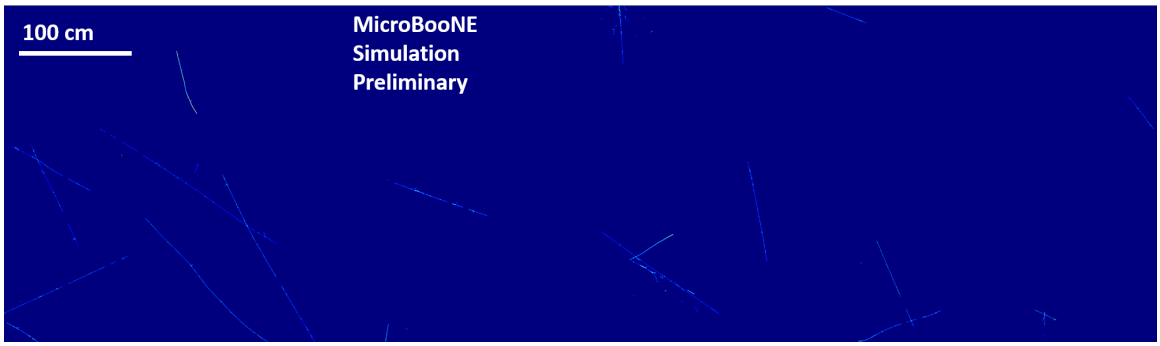

FIG. 3. An example wire signal event display for MicroBooNE's LArTPC collection plane. This is an electron neutrino simulated event. The neutrino interaction occurs about halfway across on the bottom of the image and is overlapped by a cosmic muon.

Due to differences in wire angle between the wireplanes we expect the interaction topologies to change from one plane to another. Therefore, we train the same model of network three times, one for each of the LArTPC wireplanes. Figure 3 shows an example signal input collection plane image to the network.

In order to accommodate GPU memory limits, we crop the image to dimensions of 832 columns by 512 rows. We performed the cropping by taking the original 3456x1008 signal image, and selecting on the order of 10 randomly positioned crops. We require the crops contain at least part of one interaction to avoid cropping an empty image, but maintain that the position be random rather than centered on an interaction so that the network can learn to see edges of interactions and a more cluttered and varied. The simulated training data was broken into six different

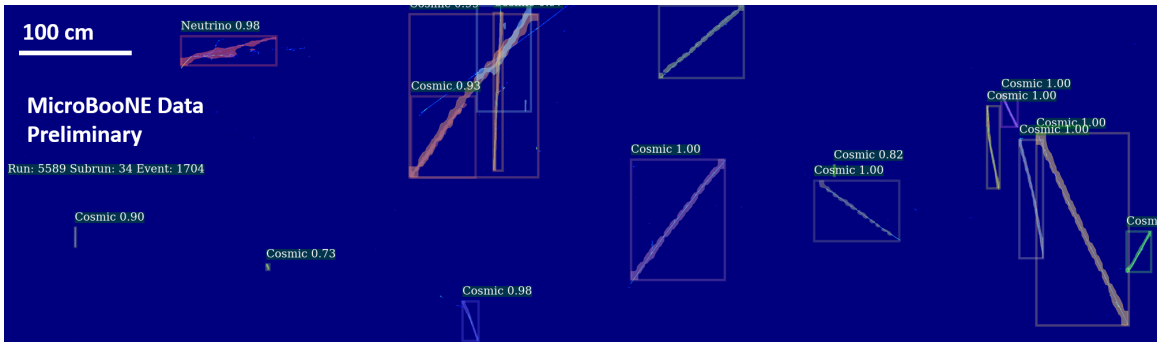

FIG. 4. An event in the open real data set. This event was chosen because it was selected by the Deep Learning 1e1p selection. Sparse ResNet sMask-RCNN correctly classifies and masks the neutrino, while also handling the cosmics in the image correctly.

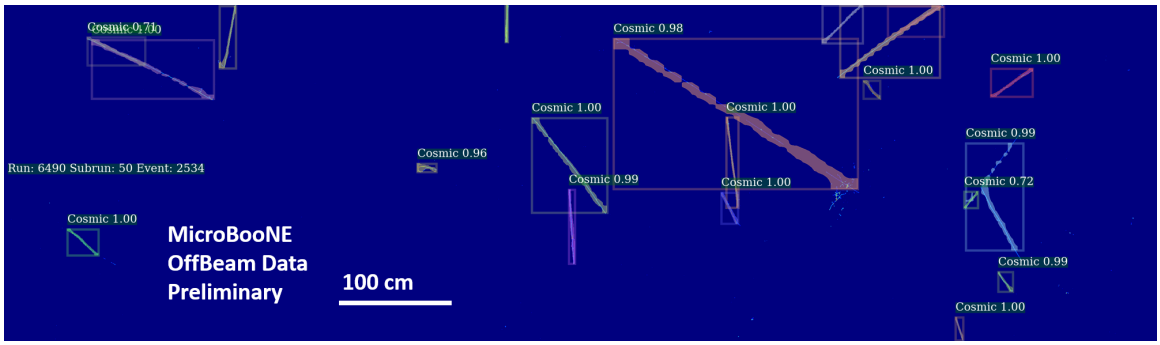

FIG. 5. An event in the External BnB off-beam dataset. As this is off-beam data we expect everything within to be classified as cosmic.

classes. The rate of occurrence of each of these classes are detailed in table I.

We cluster the charge within each image based on truth information referring to the ancestor particle that led to the charge depositing particle. We define given charge deposition's ancestor particle as the earliest particle in the mother/daughter simulation chain. In our 2D wire signal simulated images each pixel with charge deposited in it has an ancestor track or shower associated with it. We examine the particle ID (PID) of this track or shower, and categorize a given cluster as a "cosmic muon", a "neutrino interaction", or a "proton", "neutron" or "electron" that managed to find its way into our detector. If the PID associated with the track does not fall into any of these categories then we categorize the simulated truth cluster as 'Other'. For example, given a stopping cosmic muon that decays to a Michel electron, both the muon and electron's deposited charge would be clustered into one ancestor interaction truth, with a class of cosmic. Similarly, a muon neutrino that interacts with argon inside the detector forming a muon track and proton track would have charge deposited from both the muon and proton daughters count as part of the neutrino ancestor interaction.

Ancestor Class	Counts	Percentage
Cosmic Muon	2708730	92.99
Neutrino	97034	3.33
Neutron	26072	0.90
Proton	5738	0.19
Electron	155	0.005
Other	75026	2.58

TABLE I. The different class types and number of occurrences in the training set for the Dense implementation of Mask-RCNN. The ratio of occurrences of the different classes is the same for the test set for this version of the network.

While our truth information has labels for these six different classes, we focus on labeling and masking cosmic muons as a background to neutrino interactions. We do not at this time adjust class weights or in other ways worry about the difference in training examples for each class. However when we turn to training the sparse version of the network using the most up to date version of MicroBooNE simulation training set, the Neutrino class loss was upweighted relative to the cosmic class by a factor of roughly 20, in accordance with the cosmic class's abundance relative to the neutrino class in the training dataset.

As is the standard within the machine learning industry, we split our data into two distinct parts: a training set with 80% of the events and a validation set with 20%. When training the network, we always use events in the training set, and whenever we wish to measure the performance of the network, such as making event displays, or calculating the Efficiency and Purity metrics in the next section we use the validation set. The network is never allowed to perform a backward pass, the process by which a neural network 'learns' via gradient descent or loss minimization, on an event in the validation set. This is a critical part of machine learning because it prevents the network from memorizing the training dataset, and works to ensure that the network can perform its learned task on events outside of the training data. Otherwise, if we were to train and validate on the same data, then we would risk the network only learning the exact features present in the dataset, essentially overfitting to the simulated training data.

When beginning our training, we took advantage of pretrained weights available for the ResNet. These weights were trained on ImageNet, a dataset originally labeled by hand containing images of animals, people and everyday objects.

While a ResNet working with ImageNet is performing a classification task, we ignore the classification end layer,

and instead just take the weights associated with layers of the ResNet pre-classification. In this way we can leverage the feature-finding ability of the pretrained network, without needing to classify objects such as animals, objects and people. We briefly explored training ResNet from scratch, but found the pretrained weights gave better features for the other components of Mask-RCNN to utilize.

B. Sparse sMask-RCNN

Though the initial version of Mask-RCNN had good performance, the time that it takes to deploy on a given image was too slow for it to be realistically ran across the entire MicroBooNE dataset. In order to increase the speed at which the network can run over an image, we looked at leveraging the power of sparse images and submanifold convolutions to reduce the unnecessary multiplication of zeroes that occurs for empty pixels as their features propagate through the ResNet portion of Mask-RCNN. The occupancy of MicroBooNE images is on the order of 0.7-0.8%, where occupancy is defined as the number of nonzero pixels in an image divided by the total number of pixels in the image.

This means that generally there is a large number of zeroed out features that propogate through the ResNet, and later stages of Mask-RCNN. This wastes both computation resources and time. One solution to this is to employ sparse submanifold convolutions. The kernels of sparse convolutions only get applied when centered on nonzero values. Sparse convolutions are covered in more detail in [9]. This allows us to compress our images and matrices to the form of sparse matrices. Where a standard image is representated by a matrix array of dimension height by width, a sparse matrix only contains nonzero values and has dimensionality of 3 by number of nonzero values. Each nonzero value is then represented by a row, a column, and the value in the sparse matrix.

Submanifold convolutions only occur when the center input pixel in a convolutional kernel is nonzero. Otherwise the convolutional kernel passes over the location and no calculation is performed, saving time. However, in a traditional convolution, of say a 3x3 kernel. If the top left corner of the 3x3 input pixels is nonzero, but the rest of the pixels, including the center, are zero, then the convolution still outputs a value. As this occurs through multiple layers of a deep CNN information can smear, or spread out through this process. This smearing effect will not happen in a submanifold convolution as they are not allowed to output a value if the center of the kernel is zero. This means that mathematically the two processes are not identical, and the same result is not guaranteed.

We maintain the same ResNet structure as our original version of Mask-RCNN and change out the convolutions with sparse and submanifold convolutions. We also sparsify the input image, and desparsify the output features of ResNet so that the remaining RPN, RoIAlign, Classifier, and Maskifier can run on the features produced. It should be

noted that additional work could be done to make these later subnetworks use sparse convolutions. But examinations of the compute times for the individual parts of sMask-RCNN made this unnecessary for now.

In order to run on a computing cluster of CPUs the change to sparse convolutions is a significant gain in terms of network speed. We find that running the network in the original dense configuration would be too slow for production purposes, and therefore not a viable solution for cosmic tagging. The timing information for running the two different configurations of the network is shown in table II. These timings were performed on an Intel(R) Core(TM) i9-9820X CPU @ 3.30GHz, and are a measurement of wall time. Our investigation found that the time spent performing the RPN and Maskifier subnetworks is small compared to the remaining runtime of the network, and so they were not sparsified. Instead we find that the lion's share goes to performing non-maximum suppression (NMS) cuts on the RPN's proposals. A good description of what the NMS process looks like is given by Ref. [10] , but essentially it is an algorithm used to reduce duplicate proposals for a single simulated interaction. However the process scales as the (N-1)! where N is the number of proposed boxes. We can tune the parameter N to reduce time, but doing so will also affect performance. After the NMS cut the number of boxes passed onto the later half of the network is cut down to some maximum number, which is a tuneable parameter.

When it came to training the Sparse ResNet we chose to employ the concept of transfer learning on our original network rather than train the SparseResNet from scratch. We took the same dense ResNet weights pretrained on ImageNet and loaded them into the Sparse version of ResNet. We found that despite the loss of the spread of features the weights still gave us a good starting point. We also upweight the loss of the neutrino interaction class by a factor of the ratio of cosmics to neutrinos in the training set, on the order of 20, this is done to try to make the network learn the neutrino features, despite them being a small fraction of the training data.

With the implementation of Sparse ResNet we now have the GPU capacity to train sMask-RCNN on full MicroBooNE LArTPC 1008x3456 images because the space required on the GPU to move a full image through the network was drastically reduced by sparsifying at the ResNet stage.

When we were ready to start work on upgrading the network MicroBooNE had updated its version of monte carlo simulation. As such in order to use the best tools available we had to create new labeled training data. For the upgraded version of the network we used a sample of simulated electron neutrino events featuring simulated cosmic background. This means that every full sized image has a single electron neutrino interaction in it among many cosmic muons. The ratio of cosmic occurrences to neutrino interactions was 53 to 3 instead of the ratio described by

	Dense ResNet	Sparse ResNet
ResNet Run Time	3.172 s	0.1758 s
Full Detection Time	8.438 s	5.79 s

TABLE II. The runtimes for a 3456 x 1008 pixel image. The first row is the runtime for just the ResNet portion of Mask-RCNN on our images. The second row is the time to run the entire network on the images. In the case of the Sparse ResNet the time spent making the input image into a sparse tensor and the output features into a dense tensor is included in the Sparse ResNet module time.

Table I.

While the original implementation of Mask-RCNN had the six different classes in Table I, we realized that the our primary objective was to distinguish cosmic muons from neutrino interactions. In order to accomplish this we eliminated training on the labels for 'neutron', 'proton', 'electron', and 'other' by setting their loss weights to zero. While these interaction types would still be present in our simulated dataset, the network would now not be told to learn from their occurrence and instead ignore them. All of the event displays shown with labeled network output in this note use Sparse ResNet sMask-RCNN to perform their labeling.

III. PERFORMANCE EVALUATION

A. Efficiency and Purity

We use two primary metrics for measuring the effectiveness of our network in terms of the simulated ground truth interactions. A ground truth interaction is defined as the simulated interactions we know appear in an image, that we wish the network be able to find, identify, and mask. The first metric, efficiency, is a measure of how well each ground truth is covered, and the second, purity, is a measure of how well each prediction maps to a single ground truth interaction. For all of our performance metrics we only consider network proposed interactions where the classification score is above 0.4. As this score is increased higher you get worse metrics, but the remaining proposed boxes are typically better matched to the simulated truth. But decreasing the threshold allows more proposals in at the risk of getting multiple proposals for the same true interaction. We choose 0.4 to provide a balance between these two aspects, and believe a lower score threshold, with occasionally multiple proposals for a given truth better tackles the problem of cosmic tagging. If needed, we can explore a grouping algorithm that groups multiple proposals that overlap in both box and clustered region into a single proposal. Figure 6 describes our calculation of efficiency. Figure 7 describes

our calculation of purity. In both cases we only include pixels that are above an ADC threshold of 10, thus setting the ADC content into a binary 1 or 0. This cut is performed as we do not mind whether or not the network clusters empty pixels, and do not want those to count against or for the network. As such an individual image, whether it is a crop or a full image, will have an efficiency measurement per ground truth simulated interaction in the image, whereas that same image will have a number of purity measurements equal to the number of interactions predicted by the network. Both the efficiency and purity are pixel level measurements and are described in more detail below.

$$Efficiency = \frac{Sum[(Best\ Overlapping\ Prediction)\ x\ (Ground\ Truth)\ x\ (ADC\ Binary)]}{Sum[(Ground\ Truth)\ x\ (ADC\ Binary\ )]}$$

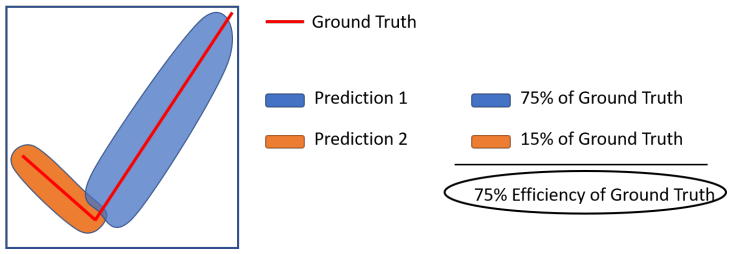

FIG. 6. The definition for efficiency. For each simulated interaction, we take the prediction with the best overlapping mask, and find the fraction of the simulated interaction's pixels that are masked.

$$Purity = \frac{Sum[(Prediction)\ x(Ground\ Truth)\ x\ (ADC\ Binary)]}{Sum[(Prediction)\ x\ (ADC\ Binary\ )]}$$

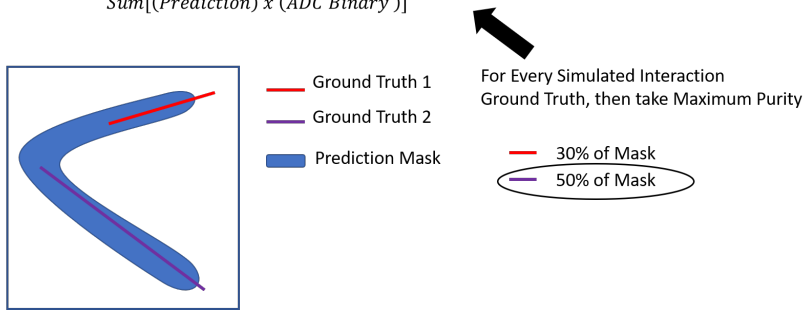

FIG. 7. The definition for purity. For each prediction we look at how many simulated interactions it maps to in order to find how purely it maps to a single interaction. Zero value pixels in the signal image included in the prediction mask area are ignored.

The efficiency is measured for each ground truth interaction by taking the pixels in the image that are associated with that ground truth simulated interaction's mask, and finding the percentage of them that are masked by the predictions. The prediction with the most overlapping mask is then used for that interaction's efficiency. The efficiency calculation is bounded between zero and one, with one being the ideal. Our efficiency metric serves to tell

us how well the network is covering the interactions taking place in the image.

The purity is measured for each prediction by taking the pixels in the predicted mask and calculating what fractions of them map to different ground truth masks. Then the purity is taken to be the maximum of the different fractions. Masked pixels that fall on no truth interaction are not counted in the denominator or the numerator. For example, if a predicted mask covered 40% Truth A, 10% Truth B and 50% blank pixel, then that mask is said to have purity of 80%, because the 50% blank pixels are not counted. We do not count prediction pixels on blanks against the network or for the network because we do not mind if we cluster additional zero charge pixels in our image, as they add no energy to the interaction, and are essentially blanks that can be thrown away later if desired via a cut on pixel value. We find that due to the scaling of the masking portion of the network, masks typically have buffers on either side of the tracks that it masks, resulting in extra 'empty' pixels clustered in with the interaction. See Fig. 2 . The purity is bounded by zero and one with the network ideal being one. We use this as a measure of how well the network is predicting that a mask maps to only one interaction, and is not using one prediction to cover multiple cosmics interactions or a neutrino and a cosmic muon together with one prediction mask.

For each event, we evaluate the average 'efficiency' for each ground truth, and the average 'purity' for each prediction. We create a 2D histogram where each entry is a single event's average efficiency and purity. In a full scale 3456x1008 pixel wire plane image the average number of simulated truth interactions is 20.82, so these averages are taken from O(20) individual efficiencies and purities. This 2D average Efficiency Purity plot is shown for the dense network in Fig. 8 and for the sparse network in Fig. 9.

The titles of these plots reference the number of epochs the networks were trained on. In the context of machine learning, an epoch is one training pass through the training data. So for 1.5 epochs would mean the network has performed a training pass on the entire dataset, and has looked at half of the dataset a second time. In this context specialized epochs refers to the number of epochs the dense network was trained on a smaller subset of the training data featuring examples of high intersection over union (IOU) simulated interactions. The IOU is just the area of the intersection of the two bounding boxes, divided by the area of the union of the two boxes. This specialized training was employed as the network was having trouble proposing multiple boxes on overlapping interactions.

With the creation of our new training set and our new sparse version of the network we recalculate our efficiency and purity metrics. However in the interest of comparing the older dense version of Mask-RCNN to the new sparse ResNet version of sMask-RCNN the efficiency and purity of both versions are reexamined using 3456x1008 full image simulation of electron neutrino interactions among cosmic background. The number of neutrino and cosmic interactions are shown

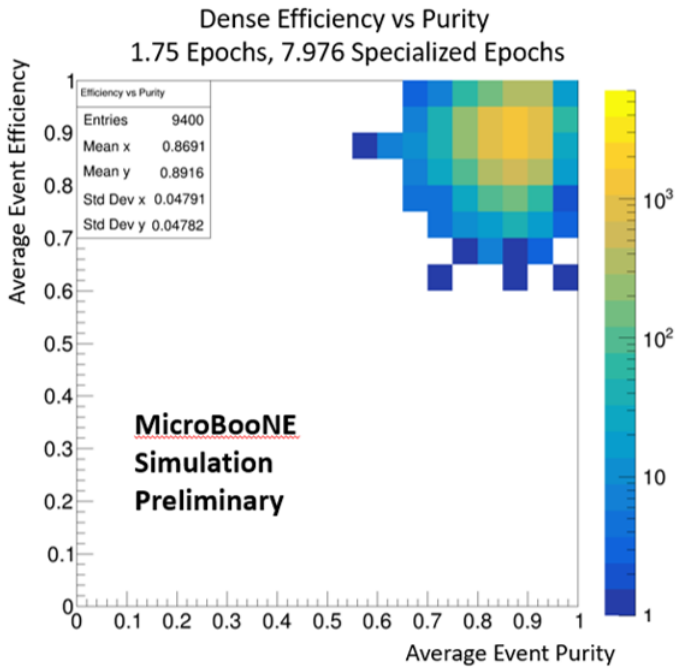

FIG. 8. The efficiency and purity the dense version of Mask-RCNN.

in table III. It should be noted that the old dense version of the network was not retrained on the new simulation. Instead the checkpoint for our original dense network was deployed on the new dataset it was not trained on. Here the term checkpoint refers to the learned weights of the network at a specific point along its training.

Ancestor Class	Counts	Percentage
Cosmic Muon	786050	95.24
Neutrino	39296	4.76

TABLE III. The different class types and number of occurrences in the training set for the Sparse implementation of sMask-RCNN. The ratio of occurrences of the different classes is the same for the validation set for this version of the network. This is the dataset for which the full image processed Efficiency and Purity comparisons are made.

It is also useful to look at the efficiency for each individual ground truth, rather than the average efficiency for each image. This is shown for the dense and sparse versions of our network in Fig. 10. We can also look at the Charge Efficiency captured by the two networks. Rather than a pixel level efficiency, we define charge efficiency as the using the adc values captured by the predicted and simulated masks. The charge efficiency is displayed in Fig. 11. In

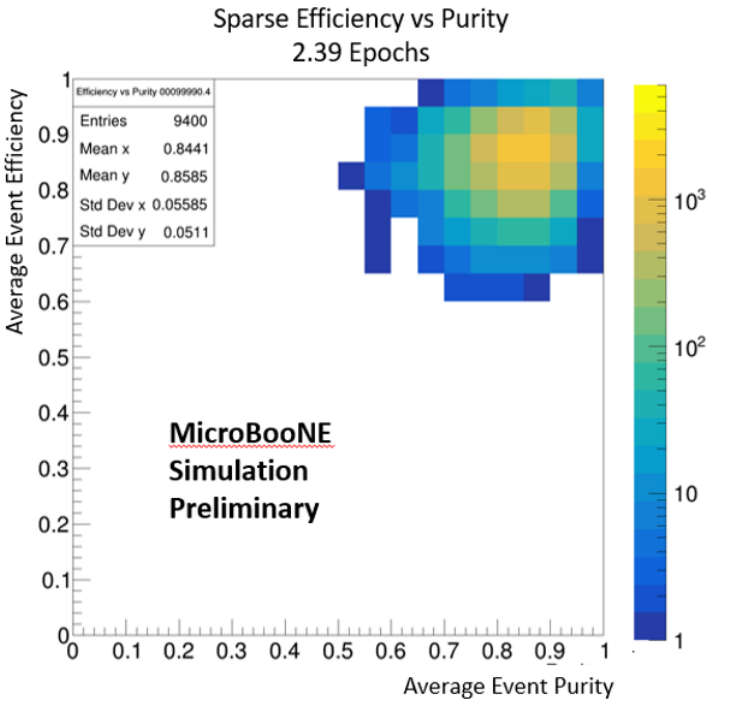

FIG. 9. The efficiency and purity for our modified Sparse ResNet sMask-RCNN version of the network.

comparing these two figures we note the charge efficiency is somewhat higher for both networks than the pixel level efficiency. This indicates the network has a preference for clustering the higher adc pixels within a given interaction than the lower value ones. While this is not surprising, it is good to confirm. We also note that regardless of the kind of efficiency we see that in each case there is a spike of efficiencies at 0 for both networks in both cases. This indicates a significant number of ground truths that have no prediction mask coverage at all. This would seem to be cases where the RPN portion of the network either does not place a box around the region, or places a box with an objectness score that is too low. Then in either case, the maskifying portion of the network has nothing to run on in that region of the image, and no overlap with the simulated ground truth is made. It is useful to note that the distribution of efficiencies besides the peak at 0 seems to peak at 1. This indicates that when the masking network has a box around a given ground truth it does a good job of masking the interaction.

At first glance in order to improve this peak at zero efficiency improvements to either the RPN or the features put out by ResNet would have to be made. However a cursory investigation of the simulated truth interactions that end up in this zero bin leads us to the conclusion that some of our true simulated interactions are unreasonable to ask the network to find. In Fig. 12 we show a bounding box for the simulated truth interaction that the network has zero efficiency for. This simulated interaction occurs nearly entirely within a dead region, and is unreasonable to ask

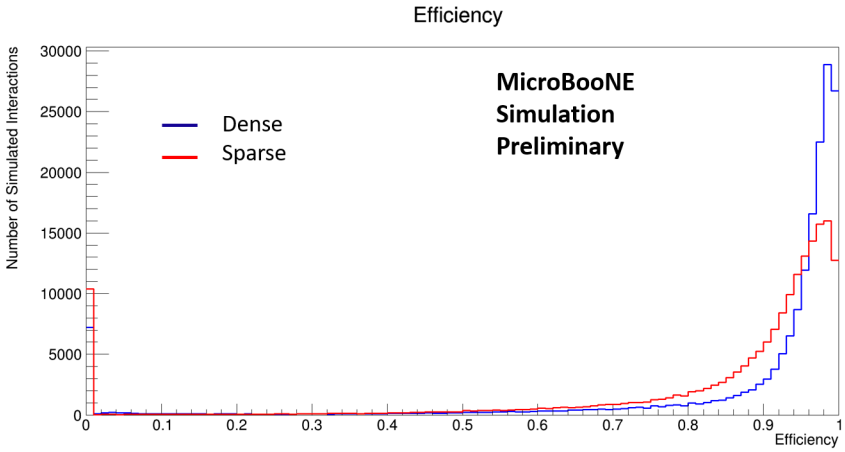

FIG. 10. The pixel level efficiency of the Dense and Sparse versions of the Mask-RCNN network. The efficiencies here are calculated using only their single most overlapping prediction mask. Run on full sized 3456x1008 images with a classification score of 0.4 Histograms integrate to the number of simulated interactions in the validation set.

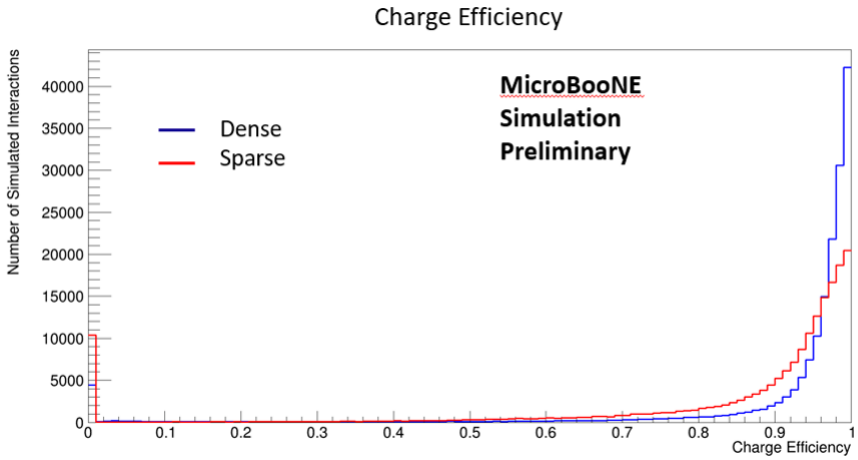

FIG. 11. The ADC or 'charge' level efficiency of the Dense and Sparse versions of the Mask-RCNN network. The efficiencies here are calculated using only their single most overlapping prediction mask. Run on full sized 3456x1008 images with a classification score threshold of 0.4. Histograms are integrate to the number of simulated interactions in the validation set.

our network to find. This is a common failure mode for the zero efficiency bin, and deflates our average efficiency. Figure 13 shows another failure mode where the network has zero efficiency on a pair of particularly small cosmic interactions. While these interactions may be a more reasonable ask, we are not as disheartened by the network missing them as they are small relative to most cosmic interactions.

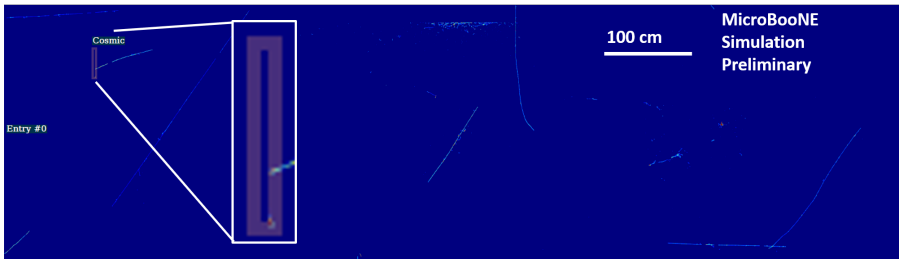

FIG. 12. The simulated true interaction bounding boxes for interactions for which the Sparse Network returns 0 efficiency. Note the missed interaction occurs nearly entirely in a dead region. This is unreasonable for the network to find, and deflates our efficiency average. This appears to be a typical failure mode for the zero efficiency bin. The network's output is suppressed in this image.

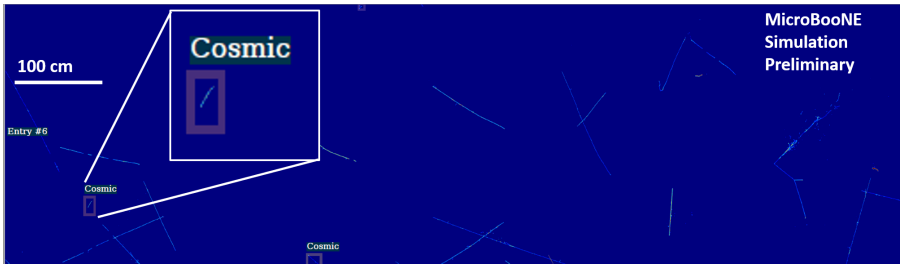

FIG. 13. The simulated true interaction bounding boxes for interactions for which the Sparse Network returns 0 efficiency. Note the missed interaction is quite small. This appears to be a typical failure mode for the zero efficiency bin. The network's output is suppressed in this image.

B. Interaction Coverage

Now that we have a measure of pixel level efficiency we can examine the ability of the network to find entire ancestor interactions. Where the average purity and average efficiency give measures of how the network is doing on a general pixel level scale. We want to know how well the network does at finding entire interactions. A simulated ground truth interaction is considered covered if the efficiency of that ground truth mask is above 90%. We choose 90% as a threshold because we want to be sure almost all of an interaction's pixels are covered, but beyond that the threshold was somewhat arbitrary. We note that this value is somewhat more stringent than we need in order to remove a significant amount of a cosmic interactions charge.

Another two dimensional histogram is made describing interaction coverage, shown in Fig. 14 and Fig. 15. On the x axis the number of simulated interactions is plotted, and on the y axis the number of interactions that reach this 90% efficiency threshold is plotted. The number of interactions covered, has to be by definition less than or equal to

the number of simulated interactions in the image. Each image creates one entry in the histogram. The ideal network would plot points along the line y=x.

A comparison of interaction coverage can also be made between the dense ResNet Mask-RCNN 14 and the sparse ResNet sMask-RCNN 15. We note the distribution of covered objects for the dense version of Mask-RCNN comes closer to the desired line of y=x. The dense version of the network covers an average of 16.59 interactions out of an average of 20.82 per event. This translates to a covering of 79.7%. Meanwhile the sparse version of the network only covers an average of 13.41 interactions, for 64.4%.

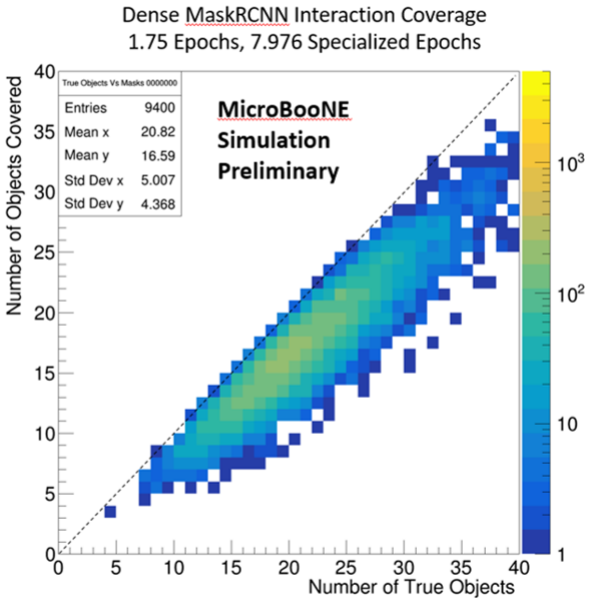

FIG. 14. The interaction coverage for our original dense ResNet Mask-RCNN network. The y axis is the number of interactions covered at 90% efficiency or more in one full sized 3456x1008 image. The efficiencies here are calculated using only their single most overlapping prediction mask. The x axis is the number of true simulated interactions in that same image. The ideal network would place all images on the line y=x.

We also examine the fractional interaction coverage. In Fig. 16 and Fig. 17 the number of interactions covered divided by the number of true simulated interactions is entered into a histogram for each image.

C. Discussion on comparison between Dense and Sparse Networks

In comparing the early dense version of our network to the newest Sparse version along these various metrics we find that the dense version of the network performs better than the current version of the sMask-RCNN. When

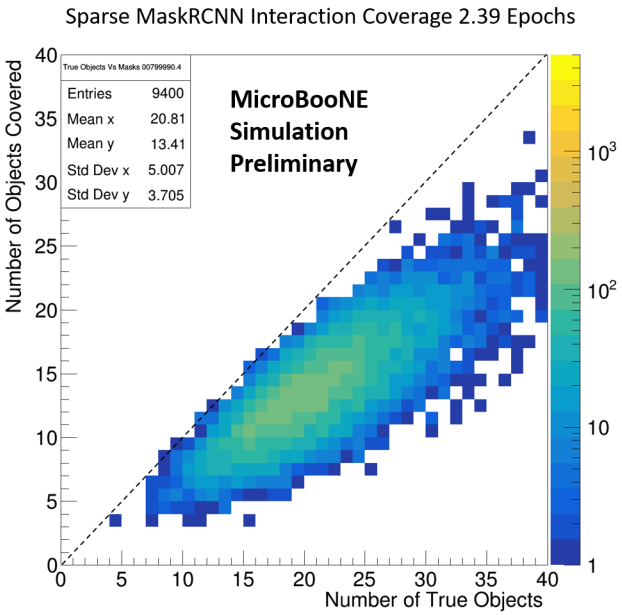

FIG. 15. The interaction coverage for our sparse ResNet sMask-RCNN network. The y axis is the number of interactions covered at 90% efficiency or more in one full sized 3456x1008 image. The efficiencies here are calculated using only their single most overlapping prediction mask. The x axis is the number of true simulated interactions in that same image. The ideal network would place all images on the line y=x

comparing the efficiencies of these two networks, the peak at 1.0 efficiency for the dense ResNet Mask-RCNN is larger and less spread than the sparse sMask-RCNN. Similarly the peak at 0 efficiency, representing interactions that are missed is smaller, indicating the dense network misses fewer entire interactions. The dense Mask-RCNN edges out sMask-RCNN in average efficiency as well with 89.1% compared to the slightly lower 85.9%. While this difference in average efficiency is only a few percentage points, because we set a threshold for interaction coverage at 90% the fact that the dense network has its average right on the threshold means that the difference in the interaction coverage is much wider. With the dense network covering 79.7% of interactions to sMask-RCNN's 64.4%. This gap we blame mainly at the somewhat arbitrary placement of the 90% cut off, instead we find it more fruitful to use the average pixel level efficiencies for a comparison. There is similarly a slight edge in our purity metric for the dense network, with 86.9% compared to sMask-RCNN's 84.4%. However, despite the dense network's slight advantage in efficiency, purity, and thus interaction coverage, we find the sMask-RCNN's particle coverage, and efficiency sufficient for the task of clustering cosmic muons, and the advantage of being able to deploy on CPU clusters invaluable in order to process the entire MicroBooNE dataset.

While the Dense version of the network has better efficiency and purity by a few percentage points we can think of

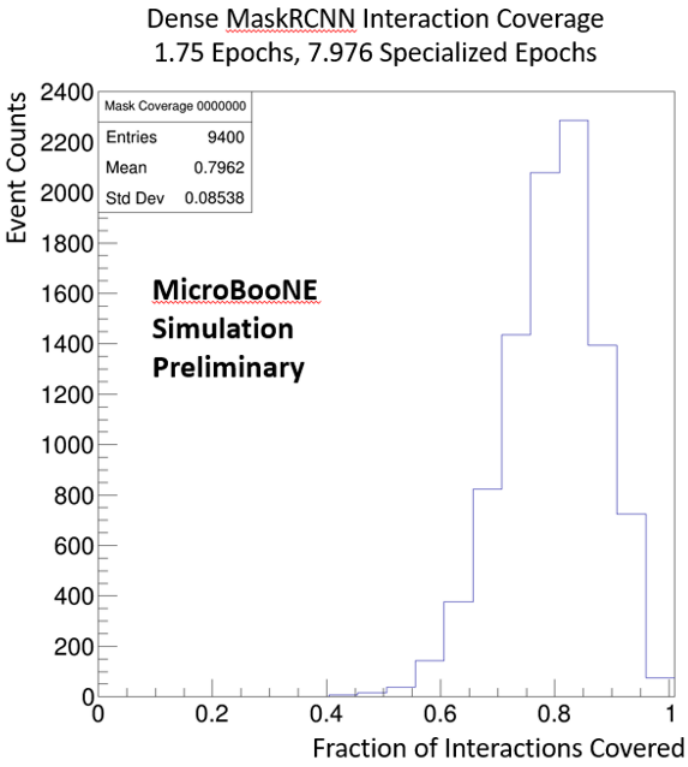

FIG. 16. The fractional interaction coverage for our original dense ResNet Mask-RCNN network. The fractional interaction coverage is defined as the number of covered interactions divided by the number of true simulated interactions in a full sized 3456x1008 image. An interaction is considered covered if its efficiency is above 90%. The efficiencies here are calculated using only their single most overlapping prediction mask.

several reasons why this may be the case. The differences between the sparse and dense networks are somewhat subtle, and challenging to pin down their effects. The dense network was trained on a simulation sample of cropped event displays of dimension 832x512. As such every image fed to the network was zoomed in and focused in around a specific few interactions. Each forward and backward learning pass for the network contained a very specific few examples. The Sparse version of our network was trained on a later improved version of MicroBooNE's monte carlo simulation. The sparse network's training set was entirely 3456x1008 full event displays, featuring on average 20.82 interactions on every training iteration. The sparse network also uses submanifold convolutions which maintain locality to features as the network deepens. This stops the spreading out of features and information.

We note that it is difficult to track the effects of these differences on the training and learning of the networks. As such we conclude that while we cannot distinguish whether the slight difference in performance of the two networks is due to the change from dense ResNet to sparse ResNet, one of these differences, or a combination of both, we determine that sMask-RCNN's ability to cluster interactions is sufficient for us to compare its cosmic tagging ability

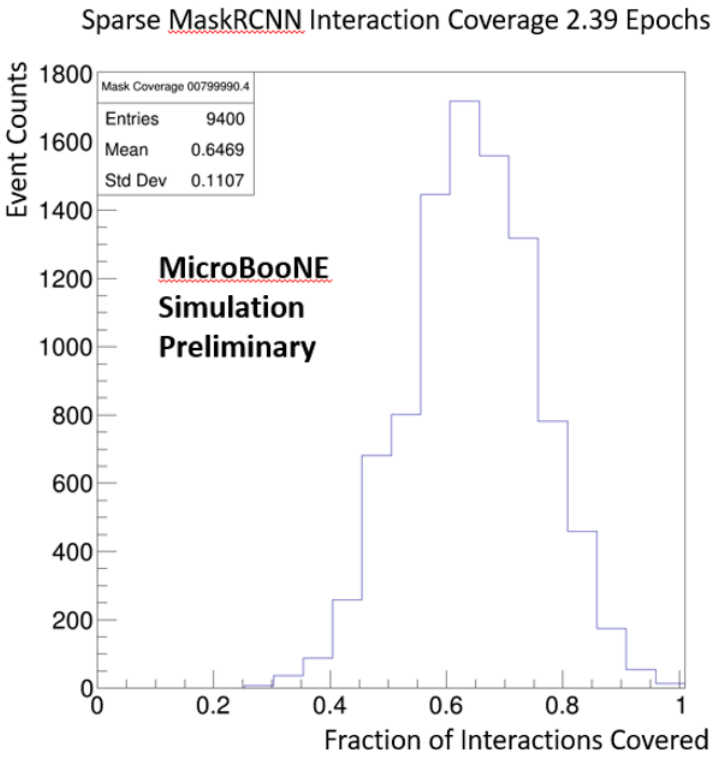

FIG. 17. The fractional interaction coverage for our sparse ResNet sMask-RCNN network. The fractional interaction coverage is defined as the number of covered interactions divided by the number of true simulated interactions in a full sized 3456x1008 image. An interaction is considered covered if its efficiency is above 90%. The efficiencies here are calculated using only their single most overlapping prediction mask.

to current methods, particularly given the speedup acquired by moving to submanifold convolutions.

IV. NEUTRINO IDENTIFICATION

One of the potential uses for sMask-RCNN in MicroBooNE is to identify neutrino interactions. There are two major approaches we have considered using. The 'identification by positive' approach would use the output of sMask-RCNN that gets labeled the neutrino class, and cut on some score threshold taking those examples to be neutrino interactions. The other approach, 'identification by negative' involves removing charge deposited in the detector when that charge gets clustered as part of a cosmic muon output by sMask-RCNN. Then everything that remains would be considered part of the neutrino interaction at this stage of identification and get passed on to the later stages of the Deep-Learning-based reconstruction in MicroBooNE.

We can examine the current capabilities of our upgraded version of the network by looking at the efficiency and

purity as defined above but now only for the neutrino class. In Fig. 18 we calculate the efficiency metric for simulated neutrino interactions, but throw away any predictions by the network that are not classified by sMask-RCNN to be Neutrino. If the network proposes no neutrino interactions, then the efficiency is 0. Though the statistics are low, there appears to be a second peak below 1.0 around 0.9 efficiency. While it has not been investigated yet, we present two potential cases for this peak. First, as this is for neutrino interactions it is possible that for a 2 or more prong event the network masks the longest prong successfully, or the track-like trunk of a shower successfully, but consistently misses the ends of showers, or one of the shorter prongs of the interaction. This would lead to a consistent near 1.0 efficiency value. The second case we imagine, is that for particularly long tracks, that get rescaled after the RPN stage before the Masking stage the end of the track may get rescaled and cut off, leading to a consistent missed amount of charge.

In Fig. 19 we calculate the purity of all proposals that the network classifies as neutrino, but only use ground truth simulated neutrino interactions. This means that a predicted neutrino that is improperly placed on a cosmic muon interaction has no truth simulation to match against, and has zero purity. Indeed we see a peak at 0 purity that represents this exact case: network proposed neutrinos that map to cosmics or noise. If sMask-RCNN were used to remove cosmics by positive identification this would be the background that gets through.

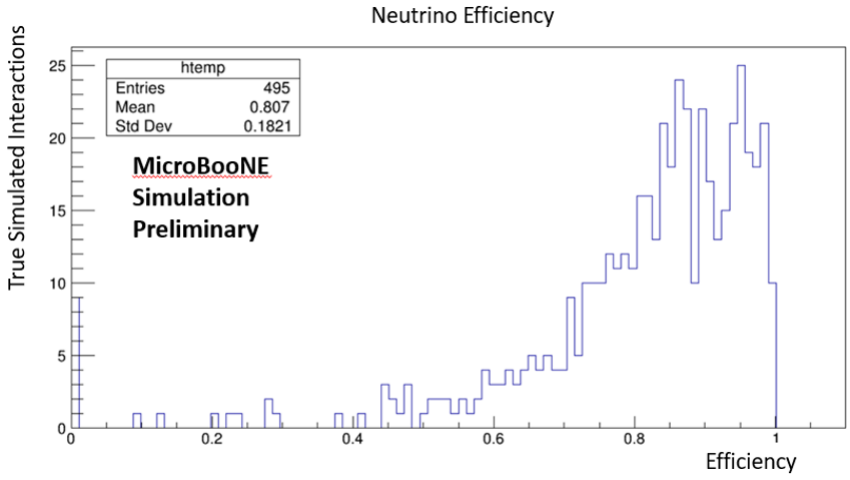

FIG. 18. The pixel level efficiency of the Sparse ResNet sMask-RCNN network for neutrino ground truth interactions. Predictions only used if the network correctly predicts a neutrino class. Run on 500 full sized 3456x1008 images with a classification score threshold of 0.4

The peak at zero purity indicates that the network currently often mislabels cosmics as neutrinos. However, the efficiency plot seems to indicate that generally the network is able to capture on average 80.7% of the neutrino pixels

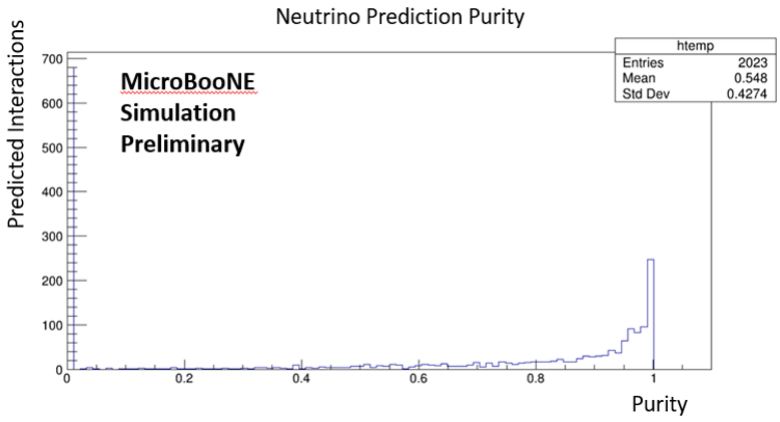

FIG. 19. The pixel level purity of the Sparse ResNet sMask-RCNN network for neutrino class predictions matched only against neutrino ground truth. The cases of 0 purity are likely cosmics mislabeled by the network as neutrinos. Run on 500 full sized 3456x1008 images with a classification score threshold of 0.4

and 81.5% of the deposited charge due to the neutrino. In addition the network rarely misses the neutrino completely.

V. FUTURE WORK

In order to determine the potential of using sMask-RCNN as a cosmic tagging tool within MicroBooNE we will need to study the ability of sMask-RCNN to mask and remove cosmics relative to that of the current cosmic tagging methods within the experiment. Currently the Deep Learning Low Energy Excess Analysis [7] uses Cosmic Tagging provided by the Wirecell group's cosmic tagging [11]. One of the upcoming important pieces of work is establishing a metric that both of these tools can be tested on. However any such test, and potential implementation would target a second generation analysis.

As we mention earlier in this note, we will explore the potential need for a grouping algorithm if we find that too often sMask-RCNN proposes multiple different clusters of the same true interaction.

Another potential use for the network is in using the 2D clusters of the network alongside a 3D matching network termed "LArMatch" in MicroBooNE [12]. LArMatch is designed to match charge in one wireplane to charge in another. When combined with the output clusters of sMask-RCNN this would give us 3D clustered, classified interactions.

We also have considered working toward making the ResNet weights portion of sMask-RCNN publicly available. As ResNet is in charge of learning features that the various subnetworks then use, it has likely learned to find features important to identifying different types of particle interactions. However this may prove more useful for a version of

sMask-RCNN that is trained on a more diverse style of particle physics interactions.

VI. ACKNOWLEDGEMENTS

The MicroBooNE Collaboration would like to acknowledge the assistance of Felix Yu, an undergraduate student at Tufts University, for their assistance on this work.

VII. REFERENCES

[1] MicroBooNE, "Proposal for a new experiment using the booster and numi neutrino beamlines: Microboone," *FERMILAB-PROPOSAL-0974*, 2007.

[2] MicroBooNE, "Design and construction of the microboone detector," *JINST*, vol. 12, 2017.

[3] K. He, G. Gkioxari, P. Dollár, and R. B. Girshick, "Mask R-CNN," *CoRR*, vol. abs/1703.06870, 2017.

[4] MicroBooNE, "Vertex-finding and reconstruction of contained two-track neutrino events in the microboone detector," *arXiv:2002.09375*, 2020.

[5] MicroBooNE, "Ionization electron signal processing in single phase lar tpcs i: Algorithm description and quantitative evaluation with microboone simulation," *JINST*, 2018.

[6] MicroBooNE, "Ionization electron signal processing in single phase lar tpcs ii: Data/simulation comparison and performance in microboone," *JINST*, 2018.

[7] MicroBooNE, "Event selection in the microboone deep learning based low energy excess analysis using two-body scattering criteria," 2020.

[8] D. Heck, J. Knapp, J. Capdevielle, G. Schatz, and T. Thouw, "Corsika: A monte carlo code to simulate extensive air showers," *FZKA 6019*, 1998.

[9] B. Graham, "Spatially-sparse convolutional neural networks," *CoRR*, vol. abs/1409.6070, 2014.

[10] K. Sambasivarao, "Non-maximum suppression (nms)," *towardsdatascience*, 2019.

[11] MicroBooNE, "Neutrino event selection in the microboone liquid argon time projection chamber using wire-cell 3-d imaging, clustering and charge-light matching," 2020.

[12] MicroBooNE, "Reconstructing 3d charge depositions in lartpcs using convolutional neural networks," 2020.