Scalable deep convolutional neural networks for sparse, locally dense liquid argon time projection chamber data

Laura Dominé,[1,2] and Kazuhiro Terao[2]

(DeepLearnPhysics Collaboration)[*]

[1]*Stanford University, Stanford, California 94305, USA*
[2]*SLAC National Accelerator Laboratory, Menlo Park, California 94025, USA*

(Received 10 January 2020; accepted 9 June 2020; published 10 July 2020)

Deep convolutional neural networks (CNNs) show strong promise for analyzing scientific data in many domains including particle imaging detectors such as a liquid argon time projection chamber (LArTPC). Yet the high sparsity of LArTPC data challenges traditional CNNs which were designed for dense data such as photographs. A naive application of CNNs on LArTPC data results in inefficient computations and a poor scalability to large LArTPC detectors such as the Short Baseline Neutrino Program and Deep Underground Neutrino Experiment. Recently, submanifold sparse convolutional networks (SSCNs) have been proposed to address this class of challenges. We report their performance on a three-dimensional (3D) semantic segmentation task on simulated LArTPC samples. In comparison with standard CNNs, we observe that the computation memory and wall-time cost for inference are reduced by a factor of 364 and 33, respectively, without loss of accuracy. The same factors for 2D samples are found to be 93 and 3.1, respectively. Using SSCN and public 3D LArTPC samples, we present the first machine learning-based approach to the reconstruction of Michel electrons, a standard candle for energy calibration in LArTPC due to their very well-understood energy spectrum. We find a Michel electrons identification efficiency of 93.9% and a 96.7% purity. Reconstructed Michel electron clusters yield 95.4% in average pixel clustering efficiency and 95.5% in purity. The results are compelling in showing the strong promise of scalable data reconstruction technique using deep neural networks for large scale LArTPC detectors.

DOI: 10.1103/PhysRevD.102.012005

I. INTRODUCTION

Deep convolutional neural networks (CNNs) have become the standard machine learning (ML) technique in the fields of computer vision, natural language processing, and other scientific research domains [1]. Applications of CNNs are actively developed for neutrino oscillation experiments [2–4], including those that employ liquid argon time projection chambers (LArTPC). LArTPCs are a type of particle imaging detector which can make two-dimensional (2D) or 3D images of charged particles' trajectories with a breathtaking resolution (∼mm/pixel) over many meters of detection volume. Current and future neutrino oscillation experiments using LArTPCs include MicroBooNE [5], short baseline near detector (SBND) [6], ICARUS [7], and the Deep Underground Neutrino Experiment (DUNE) [8]. The active volumes of these experiments are respectively about 90, 112, 600, and 40,000 tons of liquid argon.

Particle trajectories in LArTPC data, many of which have the shape of 1D lines, are recorded in 2D or 3D matrix format with an approximate pixel resolution of 3 to 5 mm. Each image has millions to billions of pixels for large LArTPC detectors (e.g., MicroBooNE produces 80 megapixels images). Those trajectories are produced by ionization electrons and are thin (a few pixels in width) and continuous. In each recorded data, depending on the experimental environment, there may be a few to dozens of particle trajectories. Therefore, LArTPC images are generally sparse, yet locally dense (i.e., no gap in between pixels that form a trajectory). This characteristic of LArTPC data poses two serious challenges for the application of CNNs. First, the matrix algebra associated with CNNs is computationally inefficient for LArTPC data which are mostly filled with zeros. Second, in photographs for which CNNs are originally developed, all pixels carry information. The strength of CNNs to automatically extract

[*]contact@deeplearnphysics.org

signal features may be affected when applied on mostly zero-filled LArTPC data.

Recently, a submanifold sparse convolutional network (SSCN) [9,10] has been proposed to address these concerns with data represented by sparse matrix or point clouds. In this paper, we demonstrate that SSCN holds strong promise for analyzing LArTPC image data with respect to both accuracy and computational efficiency, thus for being scalable to future large detectors including DUNE. Our contributions include the following:

(i) Demonstration of the better performance and scalability of sparse techniques in data reconstruction tasks that may be part of a general reconstruction chain in LArTPC data.
(ii) Study of typical mistakes made by the algorithms and mitigation methods.
(iii) First ML-based approach for the reconstruction of Michel electrons using a publicly available simulation sample, and quantification of the purity and efficiency of this approach.

All studies in this paper are reproducible using a SINGULARITY software container[1] [11], our implementation of semantic segmentation algorithms,[2] and public data samples[3] [12] provided and maintained by the DeepLearnPhysics Collaboration.

Section II gives an overview of the public data set used in this paper. Section III details the design of a neural network, U-ResNet, chosen for studying the impact of SSCN. Section IV describes our experiment including the performance metrics and training setup. Section V presents the results including the performance comparison between an SSCN (sparse) and standard (dense) implementations of U-ResNet. We discuss causes of poor performance of U-ResNet and propose mitigation methods in Sec. VI. Last, in Sec. VII, we present our approach and the results of reconstructing Michel electrons in the public simulation sample using the sparse U-ResNet.

II. DATA SET

A. Particle images

In this paper, we use 2D and 3D LArTPC simulation samples made publicly available by the DeepLearnPhysics Collaboration [12]. These are images of particles traversing a cubic volume of liquid argon, whose size can be 192 pixels (px), 512 px, or 768 px. The spatial resolution of each pixel is 3 mm. The data set contains 100,000 images for each size in both 3D and each 2D projections. We split each sample into 80% and 20% fractions as train and test sets, respectively. There are two sources of particles in this data set which are as follows:

[1]https://www.singularity-hub.org/containers/6596.
[2]https://github.com/Temigo/uresnet_pytorch.
[3]https://dx.doi.org/10.17605/OSF.IO/VRUZP.

TABLE I. On average, only 0.01% of pixels in an event are nonzero. This table shows to which class these nonzero pixels belong.

	HIP	MIP	Shower	Delta rays	Michel e^-
Fraction	17%	34%	47%	1%	1%

(i) Single, isolated particle: an electron, muon, antimuon, or proton. Ten such particles are generated in a larger volume and a cropped 3D volume is recorded.
(ii) Multiparticle vertex: About 1 to 6 particles produced at the same 3D point, including electrons, gamma rays, muons, antimuon, charged pions, and protons.

Particle interactions with the liquid argon medium are simulated using GEANT4 [13] and LArSoft [14]. The particle energy depositions are recorded in each pixel. The drift simulation, which would include, for example, the electron lifetime, recombination, diffusion, and space charge effects, is not included in this data set. However, energy depositions are smeared by a Gaussian distribution with a 3 mm width to mimic a diffusion effect while total deposited energy is conserved.

B. Labels

Among the tasks available for a benchmark in this public data set, we choose the semantic segmentation. The task is to predict a class of particle at pixel level. The labels in the data set for supervised learning include five possible classes for each pixel which are as follows:

(i) Protons, referred to as heavily ionizing particle (HIP), which typically display short, highly ionized tracks.
(ii) Minimum ionizing particles (MIP) such as muons or pions, with usually longer tracks.
(iii) Electromagnetic showers induced by electrons, positrons, and photons, with kinetic energy above critical value (about 33 MeV in argon).
(iv) Delta ray electrons from hard scattering of other charged particles.
(v) Michel electrons from the decay of muons.

The statistics of each class in the data set are shown in Table I. Figure 1 shows an example of a simulated image from this data set and the corresponding pixel-wise labels. More details about the data set can be found in Ref. [12].

III. NETWORK ARCHITECTURES

A. Dense U-ResNet: Baseline

We use a network architecture which we call U-ResNet. It is a hybrid between two popular architectures: U-Net [15] and ResNet [16].

U-Net is an autoencoder network architecture (Fig. 2) which has been successful for medical image segmentation.

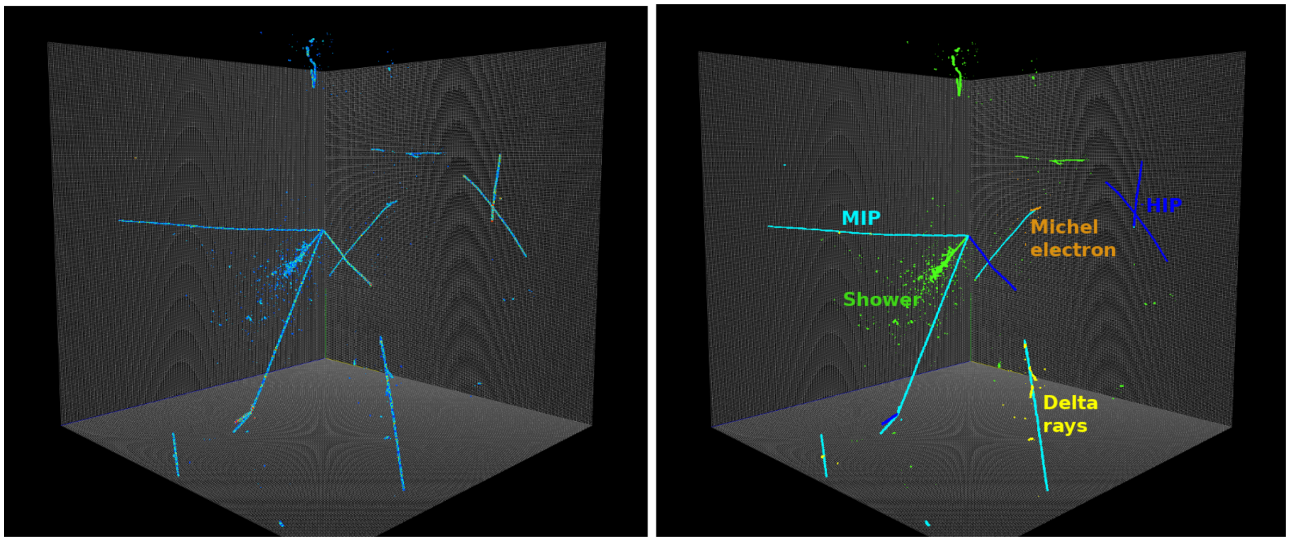

FIG. 1. Simulated LArTPC event data (left) and labels (right). The data show energy deposits from charged particle trajectories. The color corresponds to an energy scale. In the label image, each pixel is assigned one of five colors: heavily ionizing particles (HIP) in blue, minimum ionizing particles (MIP) in cyan, electromagnetic showers in green, delta rays in yellow, and Michel electrons in orange.

It is made of two parts: the first half downsamples the spatial size (using strided convolutions in our case) the input image with several convolution blocks. This part learns the image features on different scales in a hierarchical manner, yielding a tensor with a low spatial resolution but a large number of channels. These channels contain a lot of compressed feature information; hence, it is called the encoder part of the U-Net. The number of downsizing operations is referred to as *depth* in this paper and affects the receptive field area of the network. The second half of the network applies to this tensor several upsampling (we use transpose convolutions) and convolution operations. It is called a decoding path. Concatenation takes two input tensors of the same spatial dimensions and stacks them along the feature dimension (i.e., the channel axis of an image tensor). We concatenate between the feature map of the previous layer in the decoding path and the feature map of the same spatial size in the encoding path, which helps the decoder to restore the original image resolution. The input image is single channel, but the output has as many channels as there are classes.

U-Net is a generic CNN architecture. In our case, each block of convolutions/up or downsampling is made of two convolution layers, followed by a batch normalization and a rectified linear unit activation function (ramp function). According to the ResNet architecture, we also add residual skip connections (extra connections between different layers that allow to skip some layers in the network

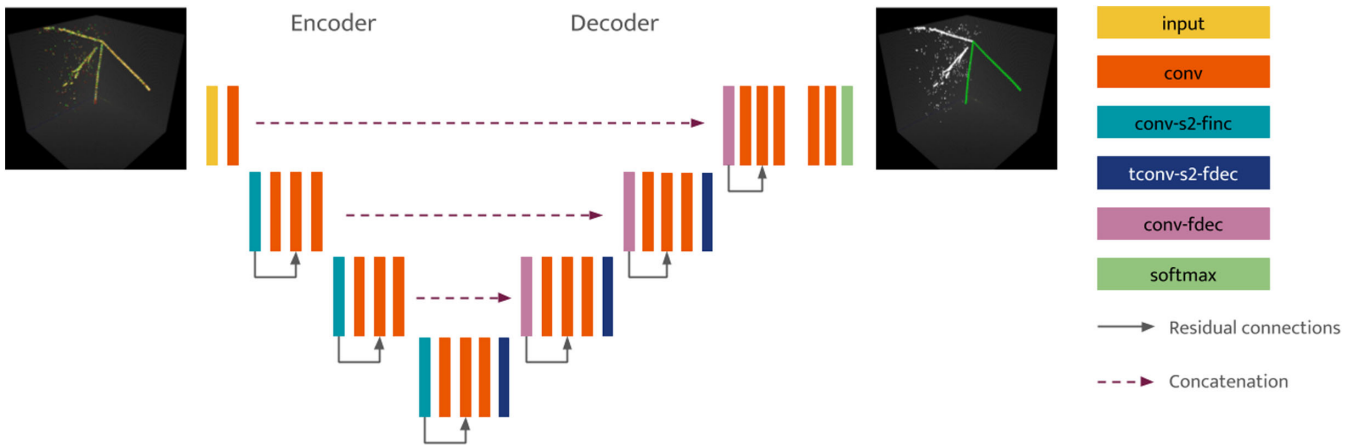

FIG. 2. U-ResNet architecture for semantic segmentation. In this example, we say that the U-ResNet has a depth of 3 since we perform three downsamplings. Turquoise boxes represent convolutions with stride 2 and increasing the number of filters. Dark blue boxes are transpose convolutions with stride 2 and decreasing the number of filters. Purple boxes are convolutions with stride 1 that decrease the number of filters. The spatial size of feature maps is constant across the horizontal dimension.

architecture) which allow the network to learn faster and be deeper. In our implementation, the number of filters at each layer increases with depth in a power law for our dense U-ResNet and linearly for our sparse U-ResNet.

The strong performance of this network for two-class semantic segmentation (between particle track and electromagnetic shower) at pixel level on real detector data was already demonstrated [3] by MicroBooNE experiment, which makes it a network of choice to benchmark a sparse technique on LArTPC simulation data.

B. Submanifold sparse convolutional networks

The key element of submanifold sparse convolutional networks [9] is a so-called submanifold sparse convolution operation. It was designed for cases where the effective dimension of the data is lower than the ambient space, for example, a 2D surface or a 1D curve in a 3D space. For such cases, the standard dense convolutions are not suitable for several reasons which are as follows:

(i) Traditional convolutions involve dense matrix multiplication operations, which are computationally inefficient for sparse data.
(ii) The submanifold dilation problem, as described in Ref. [9]: a single nonzero site in the image yields 3^d nonzero sites in the next feature map after a dense convolution, where d is the spatial dimension (in our case $d = 2$ or $d = 3$). After two convolutions, there will be 5^d nonzero sites and so on. This inescapable growth "dilates" the originally sparse image which becomes denser.

The idea of SSCNs is to keep the same level of sparsity throughout the network computations, especially convolutions. It has been shown to require significantly less computations while outperforming the dense CNNs on two 3D semantic segmentation challenges in the field of computer vision [10].

Reference [9] defines two new operations. First, sparse convolutions $SC(n, m, f, s)$ with n input features, m output features, f filters, and a stride s are defined. They address the first issue mentioned above and work in the same way as standard convolutions except they assume that the input from nonactive pixels, which are zero or close to zero, is zero. The output feature map will have a size $(l - f + s)/s$ where l is the size of the input. Second, they define a submanifold sparse convolution $SSC(n, m, f)$ with similar notations as a modified $SC(n, m, f, s = 1)$: the input is padded with $(f - 1)/2$ zeros on each side to ensure that the output image will have the exact same size. An output pixel will be nonzero if and only if the central pixel of the receptive field is nonzero in the input feature map. SSC operation tackles the second issue by constraining the output sparsity. In order to build complete CNNs based on these two operations, the authors also define a set of other custom operations such as activation functions and batch normalization layers by restricting the corresponding standard operations to the set of nonzero pixels.

IV. EXPERIMENTS

We perform two sets of experiments. First, we compare the performance between dense and sparse U-ResNet using several evaluation metrics for 2D and 3D samples. Second, we study the variation of the performance of sparse U-ResNet with key architecture hyperparameters and different image sizes.

A. Evaluation metrics

The network is trained by minimizing a loss which is a softmax cross-entropy loss averaged over all the pixels of an image. We define different metrics of interest which are as follows:

(i) Nonzero accuracy: Fraction of nonzero pixels whose label is correctly predicted.
(ii) Classwise nonzero accuracy: For each event and for each class, fraction of nonzero pixels in that class that are correctly predicted.
(iii) Resources usage during the training and testing time:
  – Graphic Processing Unit (GPU) memory occupied
  – Computation wall time

B. Implementation and training details

All networks were implemented using the PyTorch [17] (version 1.0) deep learning framework. SSCN relies on the library SPARSECONVNET[4] We use LARCV2[5] to interface with the LArTPC data files. To train the networks, we used ADAM optimizer [18] with the default learning rate of 0.001. We trained the networks for 30k iterations in 3D and 40k iterations in 2D. We used NVIDIA V100 GPUs with 32 GB memory. On 3D images of size 192 px approximately, 10 and 212 hours were required for convergence of the sparse and dense networks, respectively.

V. RESULTS

Notation: We write, for example, [2D, 512 px, 5–16] to represent "2D images of size 512 px, and U-ResNet of depth 5 with 16 filters."

A. Sparse vs dense U-ResNet

We start by comparing the performance of dense versus sparse U-ResNet using the nonzero accuracy metric as well as the computational resources usages at train and inference (or test) time.

As shown in Table II, for a fixed 3D image size of 192 px and identical training parameters (notably batch size), the

[4]https://github.com/facebookresearch/SparseConvNet.
[5]https://github.com/DeepLearnPhysics/larcv2.

TABLE II. Sparse and dense U-ResNet scalability with the 3D image spatial size. The dense U-ResNet could not fit 3D images of size 512 px nor 768 px on a single GPU. Both sparse and dense networks here have a depth 5 and number of filters 16. The batch sizes are not optimized for accuracy.

	Dense	Sparse			
Batch size	4	4		64	
Image size	192 px	192 px	192 px	512 px	768 px
Nonzero accuracy mean	92%	94%	98%	99%	99%
Nonzero accuracy std	0.096	0.088	0.049	0.014	0.0037
		Nvidia V100 GPU			
Memory (test) [GB]	16	0.044	0.19	0.67	1.3
Memory (train) [GB]	26x4	0.21	1.3	5.1	9.3
Wall-time (test) [s]	3.3	0.10	0.66	2.4	4.4
Wall-time (train) [s]	25	0.21	1.2	5.0	8.8
		Intel Xeon Silver 4110 CPU			
Memory (test) [GB]	···	0.57	0.81	1.9	3.0
Memory (train) [GB]	···	0.59	1.9	3.9	4.0
Duration (test) [s]	···	0.25	1.7	8.0	16
Duration (train) [s]	···	1.1	6.1	24	47

final nonzero accuracy mean value over the whole data set for the sparse U-ResNet is slightly higher than the dense counterpart by 2%. However, using the same batch size does not do justice to the real feat of SSCN: the GPU memory usage and computation duration are drastically cut down using sparse convolutions, which allows us to train the sparse U-ResNet with a dramatically larger batch size and a larger 3D image size. Harnessing both of these advantages allows one to beat the baseline dense 3D U-ResNet by a large margin in nonzero accuracy.

Figures 3 and 4 show the variation of memory and computation wall time for sparse U-ResNet in [2D, 512 px, 5–16]. The latter grows linearly but slowly as a function of batch size, which makes larger batch sizes practical not only for training but also for the inference. In particular, the sparse U-ResNet can easily process a whole MicroBooNE event data with a conventional GPU (memory of 4–11 GB). The resource usage scales well with the batch size to handle ICARUS detector, which is about 6 times larger than MicroBooNE. At the batch size 88, which is the maximum possible for a single NVIDIA V-100 GPU with the dense version, the reduction factors for memory and computation wall time with the sparse U-ResNet are 93 and 3.1, respectively. Further, because the computational cost scales with nonzero pixel count instead of the total pixel count in the bounded volume, sparse U-ResNet will be an ideal solution for DUNE far detector which will be sparser in the absence of cosmic rays. These benefits apply to a training phase of an algorithm. Figure 5 shows how using sparse U-ResNet speeds up the training by several orders of magnitude. This is crucial for reconstruction algorithm R&D work which often requires a short turnaround time for development.

Finally, looking at the evolution of the softmax scores for different classes across iterations indicates that the sparse U-ResNet may be learning more uniformly over pixels than its dense equivalent. Figure 6 shows how the standard deviation of the mean softmax value in the image evolves

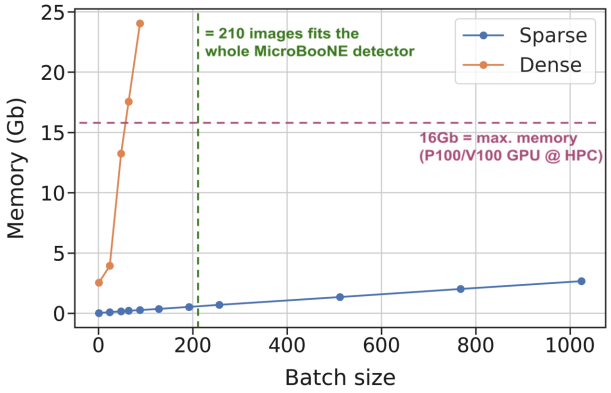

FIG. 3. GPU memory usage as a function of batch size at inference time [2D, 512 px, 5–16].

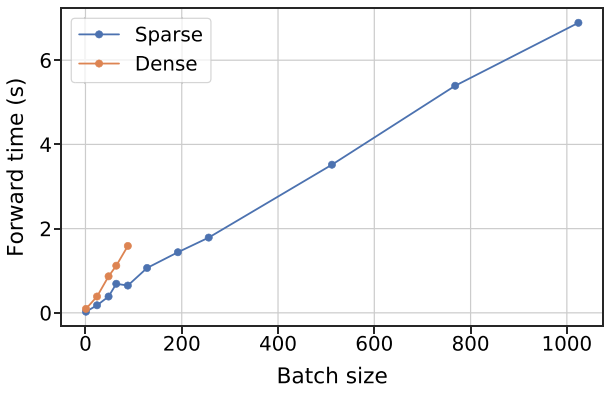

FIG. 4. Computation wall time as a function of batch size at inference time [2D, 512 px, 5–16].

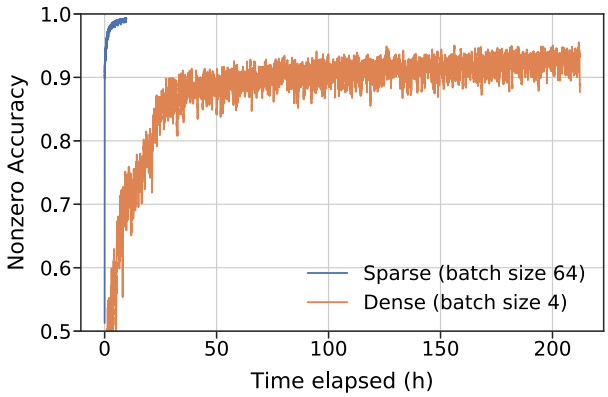

FIG. 5. Nonzero accuracy as a function of wall time during the training [3D, 192 px, 5–16]. The sparse U-ResNet uses a batch size of 64 and dense U-ResNet uses a batch size of 4.

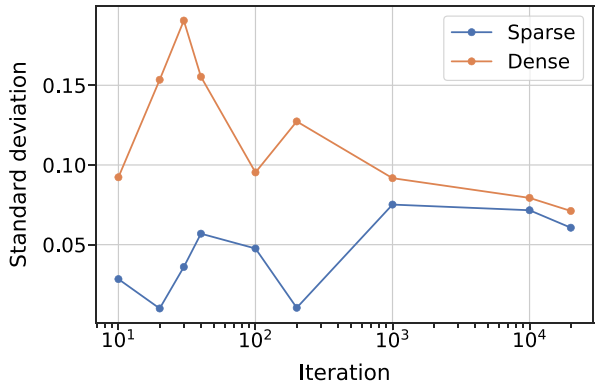

FIG. 6. Standard deviation of the mean softmax value of pixels predicted as shower pixels in an image, as a function of the training iteration step. The sparse U-ResNet appears to learn in a more uniform manner across the pixels [2D, 512 px, 5–16].

with the training iterations. The dense network results in a much higher variance. Their variances end up converging after about 1000 iterations. This observation is illustrated in Fig. 7, in which a MIP trajectory crosses an electromagnetic (EM) shower. Figures 8 and 9 compare how the softmax scores for track and shower particles change over training iterations between sparse and dense U-ResNet. The difference appears most strikingly at the iteration 40, where the dense network is extremely confident in some pixels (yellow ones) and still very unsure about others (in dark blue), while the sparse one is increasing its confidence level much more uniformly across all pixels.

B. Sparse U-ResNet performance variation

We study the influence of the two main parameters of the network architecture on performance and resource usage: depth (number of layers) and the number of filters in the first layer. Table III, Figures 10 and 11 show the results of

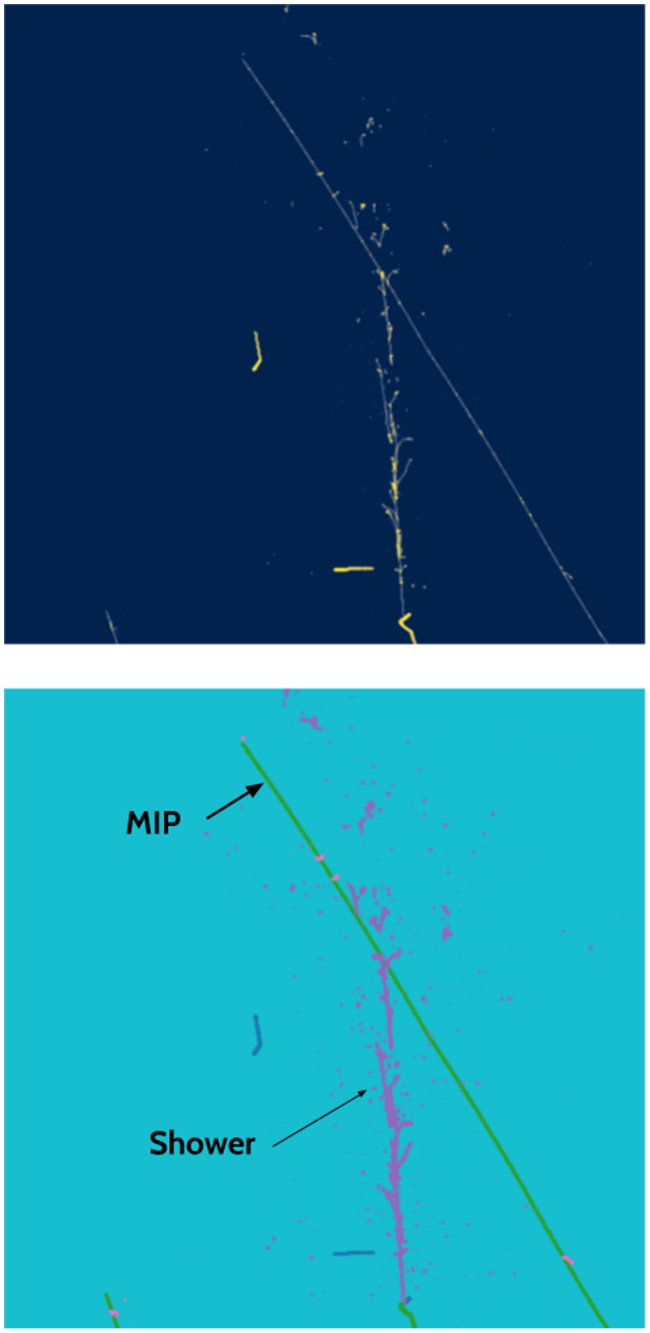

FIG. 7. Top: energy depositions in the image, the pixel color corresponds to an energy scale. Bottom: labels, each color corresponds to a different class. Electromagnetic shower pixels are colored in purple and MIP pixels are in green.

nonzero accuracy and computational resource usage, respectively. The filter counts have a larger effect on achieving a higher accuracy while it also causes a linear increase in memory usage. The increase in the computation wall time is only ≈10% between 8 and 32 filter counts.

Table IV shows the result of comparing network class-wise nonzero accuracies for varying 3D image sizes at the

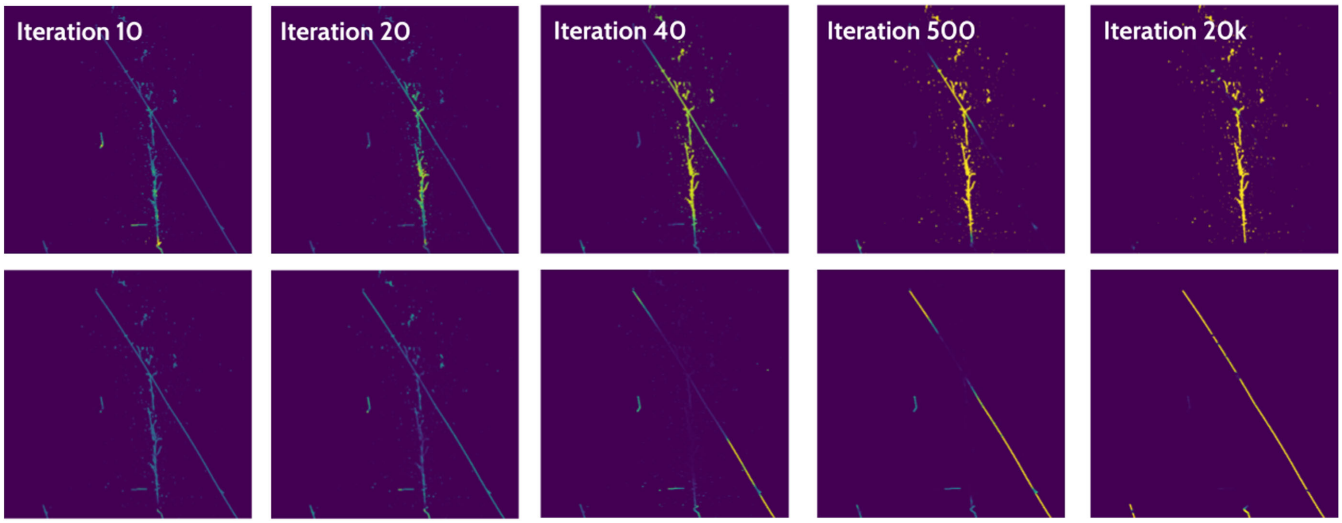

FIG. 8. Dense U-ResNet evolution of softmax value for EM shower (top) and MIP (bottom) across training iterations.

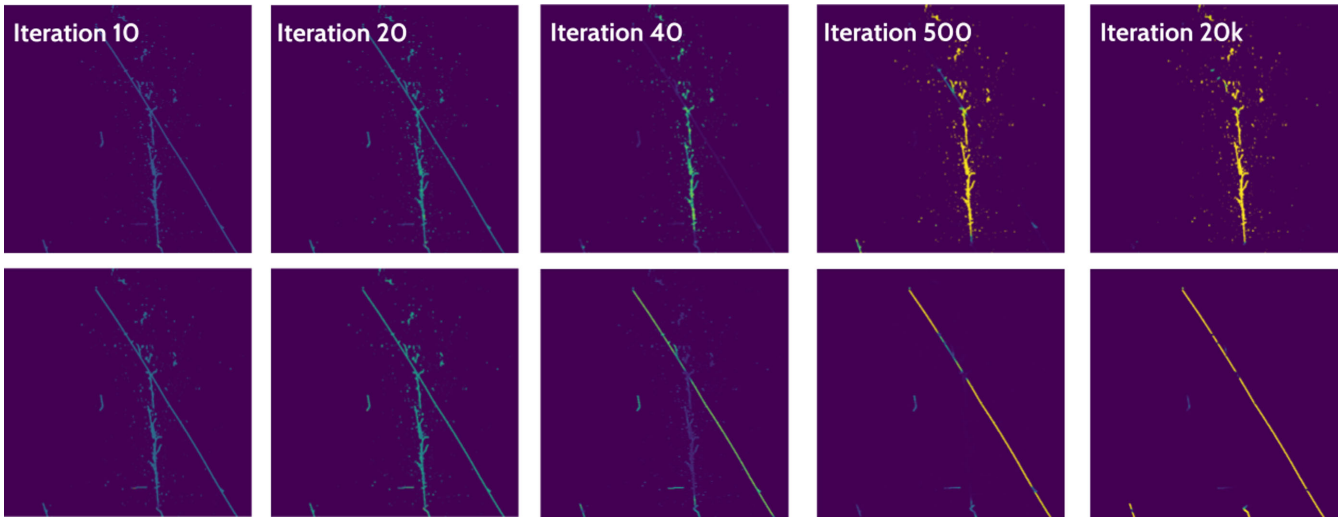

FIG. 9. Sparse U-ResNet evolution of softmax value for EM shower (top) and MIP (bottom) across training iterations.

train and test times. For a given image size at train time, using a larger image size at test time systematically improves the performance. This table also shows that the classwise accuracy of delta rays and Michel electrons is lower than other classes across all image sizes.

TABLE III. Comparison of the nonzero accuracy at inference time on the test set of 3D 512 px images for sparse U-ResNet, for different depths and initial number of filters.

Filters	8	16	32
Depth 6	98.94%	99.16%	99.23%
Depth 5	98.86%	99.07%	99.06%
Depth 4	98.74%	99.00%	99.07%

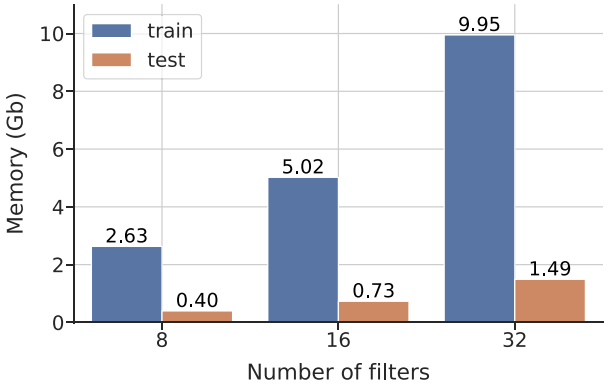

FIG. 10. Memory usage of sparse U-ResNet with depth 6, 3D 512 px images, and a varying number of initial filters.

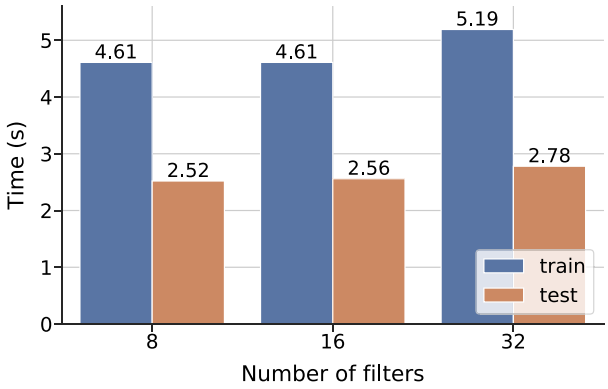

FIG. 11. Computation wall time of sparse U-ResNet with depth 6, 3D 512 px images and a varying number of initial filters.

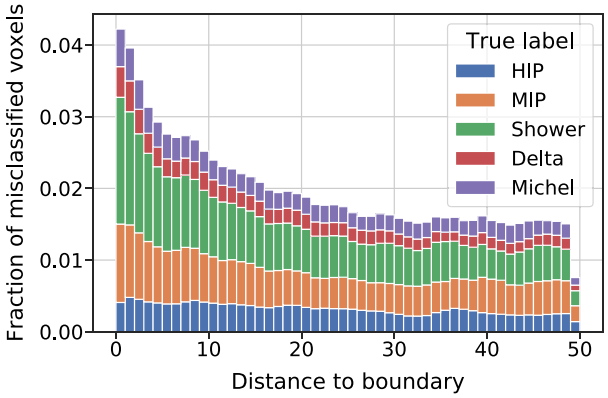

FIG. 12. Fraction of misclassified pixels as a function of the pixel distance to the image boundaries [3D, 192 px, 5–16].

C. Mistakes analysis

One may consider that a poor performance may be partially due to particle trajectories being cut out at the recorded volume boundaries. We looked at the distribution of the fraction of misclassified pixels as a function of their distance to the boundaries, which is defined as follows in d dimensions:

$$d(\{\mathbf{x_i}\}_{i=1,\ldots,d}) = \min_{i=1\ldots d} \min \mathbf{x_i}, \quad (1)$$

where d runs from 1 to 3 for 3D data. In other terms, the distance of the pixel to the image boundaries is the distance from the pixel to the closest face of the cubic image boundaries.

Figure 12 shows that, in general, pixels are more likely misclassified near image boundaries as expected. We can see that, however, this is not clearly visible for Michel electrons and delta rays. Therefore, this is not an explanation for the poor performance on these two classes. We investigated possible explanations beyond originally planned experiments and report our findings in the following sections.

Figure 13 shows some 3D images of size 512 px, which are examples of typical mistakes made by the sparse U-ResNet. This includes Michel electrons mistaken for an electromagnetic shower and vice versa, HIP mistaken for a MIP, and the tracklike beginning of a short EM shower mistaken for a MIP.

VI. MICHEL ELECTRONS AND DELTA RAYS

We propose two hypothetical explanations for low prediction accuracies on Michel and delta ray pixels by U-ResNet. The first is statistical imbalance in the fraction of pixels of each class: delta rays and Michel electrons represent each about 1% of the total pixels. The second is an ambiguous definition of these two classes: both Michel electrons and delta rays can emit gamma rays (e.g., Bremsstrahlung radiation) which appear to be indistinguishable from EM shower class. During training, we employed a softmax loss for classifying pixels under the assumption of exclusive class definitions, which may not hold for these classes.

We implemented two changes in order to test our hypothesis. The first is a modification to the pixel labels used in our supervised training. For Michel electrons and delta rays, pixels are relabeled as EM shower except for those that belong to a primary ionization trajectory, which carries distinctive features. Second, we experimented a pixel-wise loss weighting factor to accommodate statistical imbalance across five classes. This allows U-ResNet to focus more on pixels with low statistics, inspired by attention mechanisms.

TABLE IV. Classwise nonzero accuracy. Comparing the performance of sparse U-ResNet with different 3D image sizes at train and test time. The batch size, the depth, and the initial number of filters are 64, 5, and 16, respectively.

Test image		192 px			768 px	
Train image	192 px	512 px	768 px	192 px	512 px	768 px
HIP	96.0%	95.6%	93.7%	98.8%	99.0%	98.9%
MIP	96.2%	96.6%	95.4%	99.4%	99.7%	99.6%
EM shower	97.6%	96.9%	96.6%	99.5%	99.6%	99.7%
Delta rays	74.3%	76.7%	75.1%	85.9%	89.6%	90.1%
Michel e^-	36.5%	42.6%	43.9%	62.6%	70.0%	70.4%
Overall	**98.0%**	**98.1%**	**97.7%**	**98.9%**	**99.2%**	**99.3%**

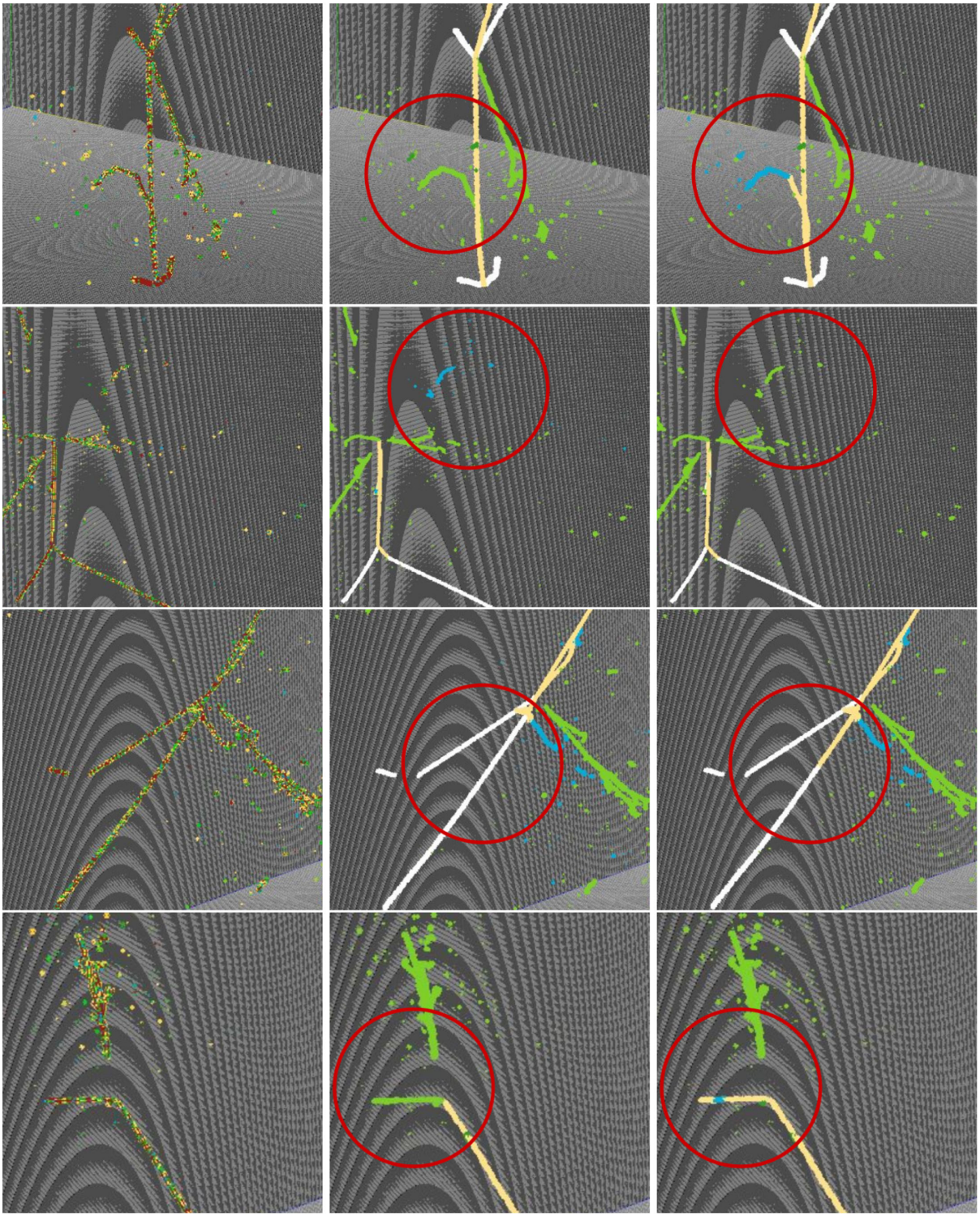

FIG. 13. Typical mistakes of sparse U-ResNet [3D, 512 px, 5–16]. Images are selected among the worst 0.05% with respect to the nonzero accuracy metric. Mistakes are circled in red. Left column: data. Middle column: labels. Right column: predictions of the network. First row: an electromagnetic shower is mistaken for a Michel electron. Second row: a Michel electron is mistaken for an electromagnetic shower. Third row: a part of a HIP is mistaken for a MIP. Fourth row: a short shower is mistaken for a (MIP) track.

We train a sparse U-ResNet using the relabeled data set and optionally the pixel-wise loss weighting scheme. The results are presented in Table V. First, regardless of whether the U-ResNet was trained on the regular or relabeled data set, the nonzero accuracy on Michel electrons increased by more than 40%. This implies that the algorithm did learn

TABLE V. A comparison of classwise nonzero accuracies between three flavors of sparse U-ResNet: regular, trained with relabeled data set, and trained with both the relabeled data set and the weighting scheme [3D, 512 px, 5–32]. We also compare the performance of the regular sparse U-ResNet on a test relabeled data set.

Train data	Regular	Regular	Relabeled	Relabeled+Weights
Test data	Regular	Relabeled	Relabeled	Relabeled
HIP	98.0%	98.1%	98.1%	99.3%
MIP	99.4%	99.2%	99.4%	98.1%
Shower	99.4%	97.9%	99.2%	99.2%
Delta rays	85.7%	94.8%	96.0%	97.2%
Michel e^-	**56.6%**	**94.4%**	**94.7%**	**95.7%**
Overall	99.2%	99.2%	99.6%	99.1%

the distinctive features of Michel electrons and delta rays without relabeling. Second, we see a slight improvement for delta rays and EM shower pixels by training with the relabeled data set. Finally, pixel-wise loss weighting further improved the accuracy of both Michel and delta ray classes as expected.

VII. MICHEL ELECTRON RECONSTRUCTION

Finally, we present a study on reconstructing Michel electron energy spectrum using the public simulation sample. Michel electron is one of the well-understood physics signals and thus useful for detector calibrations. This analysis has been done by LArTPC experiments with real data including MicroBooNE and ICARUS [19,20]. Our contribution is to show the first ML-based approach with quantification of both efficiency and purity of reconstructed signal.

A. Reconstruction method

Our goal is to quantify the efficiency and purity of clustering Michel electron energy depositions by only using the primary ionization component of its trajectory. We use the 3D 512 px images from the relabeled sample presented in the previous section. After running the U-ResNet for semantic segmentation, we isolate pixels that belong to each of the five classes. We run a common density-based spatial clustering algorithm DBSCAN [21] to identify different predicted Michel electron clusters and MIP clusters, with parameters $\epsilon = 2.8$ and $min_{Pts} = 5$ and 10, respectively. We then select the candidate Michel electron clusters that are attached to the edge of a predicted MIP cluster. Here "attached" is defined as less than 1 px distance between the nearest pixels of a Michel electron and MIP clusters. The "edginess" of a given pixel is evaluated by masking surrounding pixels within the radius of 15 px and making sure that the DBSCAN algorithm only finds one cluster when run over the remaining MIP cluster pixels.

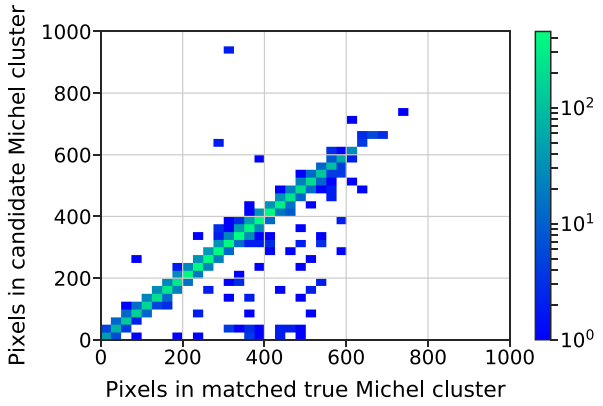

FIG. 14. A comparison of pixel counts between the true and candidate Michel electron clusters.

B. Performance metrics

After identifying candidate Michel electron clusters, we match each of them to a true Michel cluster by maximizing the overlap pixel count between true and predicted Michel cluster. We can then define several performance metrics. Let us define notations: N_i^{pred} is the total number of pixels in the predicted Michel electron cluster i, N_i^{true} the total number of pixels in the matched true Michel electron cluster i, and N_i the number of pixels which belong to the intersection of both candidate and matched Michel electron clusters. Then we define clustering efficiency and purity as N_i/N_i^{true} and N_i/N_i^{pred}, respectively. Similarly, if N^{true} is the total number of true Michel electron clusters in the sample, N_{pred} is the total number of candidate Michel electron clusters, and $N_{\text{pred}}^{\text{true}}$ is the number of matched candidate Michel electron clusters over the whole sample, then we define ID efficiency and purity as $N_{\text{pred}}^{\text{true}}/N^{\text{true}}$ and $N_{\text{pred}}^{\text{true}}/N_{\text{pred}}$, respectively.

C. Results

Figure 14 shows the pixel count of matched true Michel electron clusters against the pixel count of reconstructed Michel electron clusters. As expected, most of the clusters lie on the diagonal. The majority of strayed clusters are present below the diagonal and are underclustered.

TABLE VI. ID purity and efficiency as well as cluster purity and efficiencies of reconstructed Michel electrons. The sample size is the number of true positives. The cluster efficiency and purity are averaged over all reconstructed Michel electron clusters.

Cut	None	10
Sample size	6998	6961
ID purity	96.7%	97.3%
ID efficiency	93.9%	93.4%
Cluster efficiency	95.4%	96.0%
Cluster purity	95.5%	96.0%

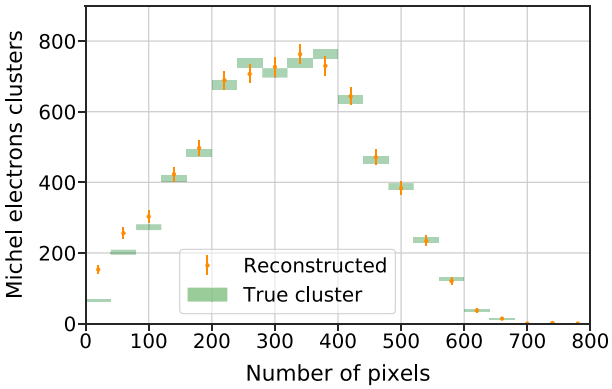

FIG. 15. Pixel counts of Michel electrons clusters. The true cluster pixel count comes from the primary ionization of the Michel electron after relabeling. The reconstructed pixel count comes from the candidate Michel clusters [3D, 512 px, 5–32].

Table VI shows the evaluation metrics with and without an analysis quality cut, which requires reconstructed Michel electron clusters to contain 10 or more pixels. We find that 89.8% of the reconstructed Michel electrons have both cluster efficiency and purity above 95%. MicroBooNE Collaboration has published Michel electron reconstruction study with 2% ID efficiency and 80%–90% ID purity where the focus of the analysis was to maximize the purity of the sample for accurate energy reconstruction [19]. The outcome of this study with the public simulation sample cannot be directly compared with others using real detector data because the public simulation sample lacks complicated detector effects. However, the results are compelling to show the promise of ML-based reconstruction approach.

Finally, using the candidate Michel electrons, we present the reconstructed and true spectrum in the primary ionization component. We used the pixel count in each cluster as the measure of energy. Figure 15 shows a reasonable agreement except for low pixel count bins where there is an excess of reconstructed candidates. These are misreconstructed candidates that have no true Michel electron counterpart. The simple reconstruction procedure introduced in this paper can be improved to reduce such artifacts. Furthermore, in order to reconstruct the total true Michel electron energy, the reconstruction step needs to account for EM shower pixels resulting from Bremsstrahlung radiation as described in the MicroBooNE publication [19]. This is out of the scope of this paper.

VIII. CONCLUSIONS

In this paper, we demonstrated the strong performance of SSCN against our baseline dense CNN for LArTPC data reconstruction, specifically for the task of semantic segmentation to identify five particle classes at a pixel level. We employed U-ResNet, an architecture pioneered by MicroBooNE Collaboration, and showed that the implementation using SSCN makes a drastic improvement in the computational resource usage. For U-ResNet under the same condition of batch size 4 with 192 px 3D images, SSCN reduces the computational cost in memory and wall time at inference by a factor of 354 and 33, respectively, as shown in Table II. For 2D samples, using batch size 88, those reduction factors are 93 and 3.1, respectively. While a naive application of standard CNN for 3D data (e.g., the DUNE near detector) comes with prohibitive and extremely inefficient computational resource usage, we demonstrated that SSCN can mitigate such costs and generalize U-ResNet for 3D data samples without loss in the algorithm performance.

We presented the first demonstration of reconstructing Michel electron clusters, defined as the primary ionization component of a trajectory, using a primarily ML-based method. Our result using the public simulation sample shows a naive approach with DBSCAN on U-ResNet output can yield 93.9% Michel electron identification efficiency with 96.7% true positive rate. Pixel clustering efficiency for reconstructed Michel electrons is found to be 95.4% with the purity of 95.5%. In particular, 89.8% of reconstructed Michel electrons are found to carry both the efficiency and purity of clusters above 95%.

SSCN is a solution to address scalable CNN applications for LArTPC data, which is generically sparse but locally dense. Furthermore, SSCN is a generic alternative to dense CNN and can be applied to tasks beyond semantic segmentation including image classification, object detection, and more. Its performance depends on the sparsity of data, which may vary among detectors. For LArTPC detectors, however, a simple thresholding technique can achieve a high sparsity without loss of signal information as demonstrated in MicroBooNE experiment [2]. We consider that the technique remains relevant for the future accelerator-based LArTPC oscillation experiments. For other imaging detectors, the presence of noise which may result in less sparsity must be taken into account. We strongly recommend SSCN for any CNN applications that exist for LArTPC experiments including MicroBooNE, ICARUS, SBND, and DUNE.

ACKNOWLEDGMENTS

This work is supported by the U.S. Department of Energy, Office of Science, Office of High Energy Physics, and Early Career Research Program under Contract No. DE-AC02-76SF00515.

[1] Y. LeCun, Y. Bengio, and G. Hinton, Nature (London) **521**, 436 (2015).
[2] R. Acciarri *et al.* (MicroBooNE Collaboration), J. Instrum. **12**, P03011 (2017).
[3] C. Adams *et al.* (MicroBooNE Collaboration), Phys. Rev. D **99**, 092001 (2019).
[4] A. Radovic, M. Williams, D. Rousseau, M. Kagan, D. Bonacorsi, A. Himmel, A. Aurisano, K. Terao, and T. Wongjirad, Nature (London) **560**, 41 (2018).
[5] R. Acciarri *et al.* (MicroBooNE Collaboration), J. Instrum. **12**, P02017 (2017).
[6] M. Antonello *et al.* (MicroBooNE, LAr1-ND, and ICARUS-WA104 Collaborations), arXiv:1503.01520.
[7] S. Amerio *et al.* (ICARUS Collaboration), Nucl. Instrum. Methods Phys. Res., Sect. A **527**, 329 (2004).
[8] R. Acciarri *et al.* (DUNE Collaboration), arXiv:1601.02984.
[9] B. Graham and L. van der Maaten, arXiv:1706.01307.
[10] B. Graham, M. Engelcke, and L. van der Maaten, in *Proceedings of the IEEE Computer Vision and Pattern Recognition CVPR, Salt Lake City, UT, USA* (Institute of Electrical and Electronics Engineers (IEEE), 2018), p. 18.
[11] K. G. Sochat, V. V. Sochat, and C. J. Prybol, PLoS One **12**, e0188511 (2017).
[12] C. Adams, K. Terao, and T. Wongjirad, arXiv:2006.01993.
[13] S. Agostinelli *et al.* (GEANT4 Collaboration), Nucl. Instrum. Methods Phys. Res., Sect. A **506**, 250 (2003).
[14] E. L. Snider and G. Petrillo, J. Phys. Conf. Ser. **898**, 042057 (2017).
[15] O. Ronneberger, P. Fischer, and T. Brox, in *International Conference on Medical Image Computing and Computer-Assisted Intervention* (Springer, Cham, Switzerland, 2015), Vol. 9351, pp. 234–241.
[16] K. He, X. Zhang, S. Ren, and J. Sun, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (Institute of Electrical and Electronics Engineers (IEEE), 2016), pp. 770–778.
[17] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, L. Desmaison, Alban Antiga, and A. Lerer, in *NIPS-W* (Neural Information Processing Systems Foundation, Inc. (NIPS), 2017).
[18] D. P. Kingma and J. Ba,arXiv:1412.6980.
[19] R. Acciarri *et al.* (MicroBooNE Collaboration), J. Instrum. **12**, P09014 (2017).
[20] S. Amoruso *et al.* (ICARUS Collaboration), Eur. Phys. J. C **33**, 233 (2004).
[21] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, in *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96 (AAAI Press, Palo Alto, California, 1996), pp. 226–231.