Trigger-level event reconstruction for neutrino telescopes using sparse submanifold convolutional neural networks

Felix J. Yu,[1,*] Jeffrey Lazar,[1,2,†] and Carlos A. Argüelles[1,‡]

[1]*Department of Physics and Laboratory for Particle Physics and Cosmology, Harvard University, Cambridge, Massachusetts 02138, USA*

[2]*Department of Physics and Wisconsin IceCube Particle Astrophysics Center, University of Wisconsin–Madison, Madison, Wisconsin 53706, USA*

(Received 27 March 2023; accepted 23 August 2023; published 12 September 2023)

Convolutional neural networks (CNNs) have seen extensive applications in scientific data analysis, including in neutrino telescopes. However, the data from these experiments present numerous challenges to CNNs, such as nonregular geometry, sparsity, and high dimensionality. Consequently, CNNs are highly inefficient on neutrino telescope data, and require significant preprocessing that results in information loss. We propose sparse submanifold convolutions (SSCNNs) as a solution to these issues and show that the SSCNN event reconstruction performance is comparable to or better than traditional and machine learning algorithms. Additionally, our SSCNN runs approximately 16 times faster than a traditional CNN on a GPU. As a result of this speedup, it is expected to be capable of handling the trigger-level event rate of IceCube-scale neutrino telescopes. These networks could be used to improve the first estimation of the neutrino energy and direction to seed more advanced reconstructions, or to provide this information to an alert-sending system to quickly follow-up interesting events.

DOI: 10.1103/PhysRevD.108.063017

I. INTRODUCTION

Gigaton-scale neutrino telescopes have opened a new window to the Universe, allowing us to study the highest-energy neutrinos. While there are a variety of proposed designs, many follow the detection principle outlined by the DUMAND project [1] and consist of an array of optical modules (OMs) deployed in liquid or solid water. This detector paradigm shows great promise, and analyses by these experiments have already provided the first evidence of astrophysical neutrino sources [2,3]. Before they can be analyzed, however, high-energy neutrinos must be isolated from the immense cosmic-ray-muon-induced background. While a high-energy neutrino may trigger a detector once every few minutes, cosmic-ray muons typically induce a trigger rate on the order of kHz.

Since they are unable to traverse a substantial portion of the Earth without coming to rest, cosmic-ray muons have a distinct zenith dependence. This allows them to be removed by cutting on the reconstructed direction of an event. Thus, a reliable reconstruction that is capable of keeping up with the ∼kHz background rate is the first step in isolating neutrinos. Moreover, a rapid reconstruction method could serve as part of an alert system that notifies researchers of events that are highly likely to be astrophysical neutrinos. For example, the real-time follow-up of such an IceCube event led to the observation of the first astrophysical neutrino source candidate, TXS $0506+056$, by detecting a neutrino in coincidence with a gamma-ray flare [2]. Along this line, similar efforts are underway in water-based detectors such as ANTARES, see Ref. [4] for a recent review.

At the trigger level, a simple but fast reconstruction is typically done by solving a least squares problem via matrix inversion, as is the case for LineFit [5] in IceCube or QFit in ANTARES [6]. Machine learning has shown promise by delivering a comparable quality reconstruction with less run-time requirements [7,8]; however, the fastest convolutional neural network (CNN) developed for high-energy neutrinos is not able to keep pace with a kHz scale trigger-level rate. In this article, we introduce a reconstruction method using a sparse submanifold CNN (SSCNN), which overcomes this run-time issue. Our SSCNN achieves better angular resolutions than methods such as LineFit while requiring a comparable run-time, enabling improved trigger-level cuts and serving as a better seed for the likelihood-based reconstruction. Figure 1

[*]felixyu@g.harvard.edu
[†]jlazar@icecube.wisc.edu
[‡]carguelles@g.harvard.edu

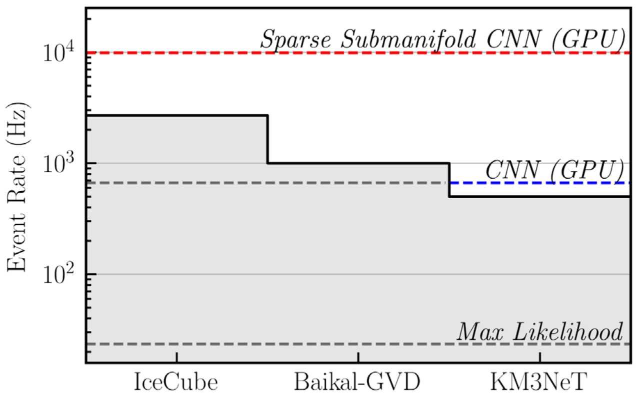

FIG. 1. Event rates of triggers in different neutrino telescopes [9–11] compared to the run times of various reconstruction methods. Sparse submanifold CNNs and their performance are detailed in this article (the rate shown comes from the angular reconstruction model). The CNN and maximum likelihood method run times are taken from [7]. Notably, sparse submanifold CNNs can process events well above standard trigger rates in both ice- and water-based experiments.

summarizes typical event rates found in neutrino telescopes and compares these to the execution rate of various reconstructions. SSCNN is also capable of reconstructing the neutrino energy, a task which has not been done at trigger level.

While the rest of this article will concentrate on the implementation of SSCNN in an ice-embedded IceCube-like detector, it should be noted that our method is also applicable to water-based neutrino telescopes, where we expect similar performance gains. The rest of this article is organized as follows: In Sec. II we motivate and introduce sparse submanifold convolutions; in Sec. III we describe the data sets used for training and testing; in Sec. IV we evaluate the performance of the network. Finally, in Sec. V we conclude with some parting words. The code detailing our implementation of SSCNN has been made available at Ref. [12].

II. METHODS

A. Sparse submanifold convolutions

Convolutional neural networks (CNNs) have become the staple architecture for image-like data, and have achieved great success in a wide range of applications, including neutrino physics [13–16]. However, data from neutrino telescopes presents inherent challenges to CNNs. In particular:

(i) *Nonregular geometry*: CNNs are designed to operate on images, which are arranged on Cartesian grids. Neutrino telescope sensors are typically spaced irregularly [10,17–19], with varying distances and arrangements in between each sensor.

(ii) *Sparsity*: Traditional CNNs use convolutions which operate on all points in the given input data. This leads to computational inefficiencies when the data is sparse.

(iii) *High dimensionality*: Events occur in large spatial and temporal scales. This makes using traditional CNNs computationally unfeasible on raw 4D data (three spatial and one time) without information loss or significant preprocessing.

In this article, we propose a solution to these challenges using sparse submanifold convolutions [20]. This strategy has already shown success in liquid argon time-projection chamber neutrino experiments [21,22]. The usage of sparse submanifold convolutions in our network naturally solves the challenges laid out above. Sparsity and high dimensionality are no longer a concern, as the number of computations performed will depend only on the number of OM hits. With this improved computational efficiency, we can also handle nonregular geometries more smoothly by using the spatial coordinates of each OM hit (in meters from the center of the detector). This allows us to consider data of any shape or arrangement, without restricting ourselves to a Cartesian grid; thus our algorithm can be easily adapted from our IceCube-like test case to, e.g., IceCube, KM3NeT, P-ONE, etc.

Our SSCNN replaces traditional convolutions with sparse submanifold convolutions. While a traditional convolution extracts features by mapping a learned kernel over all input data, a sparse submanifold convolution operates only on the nonzero elements. This circumvents the inefficiency of using CNNs on sparse data, wherein the vast majority of operations are wasted multiplying zeros together. Furthermore, to preserve the sparsity of the data after applying multiple layers in succession, sparse submanifold convolutions enforces that the coordinates and number of output activations matches those of the input. In other words, the features do not spread layer after layer, as shown in Fig. 2. This is crucial for the efficiency of SSCNNs that are very deep, as the data would otherwise become progressively less sparse throughout the network. The lack of feature spreading will have a minimal impact on performance as long as the network can rely on local information. It should be noted that SSCNNs still compute over a grid-like structure, but this structure can be arbitrarily large because the network only operates on a submanifold of it.

B. Input format

As input, the SSCNN takes in two tensors: a coordinate tensor C, and a feature tensor F. In symbols,

$$C = \begin{bmatrix} x_1 & y_1 & z_1 & t_1 \\ \vdots & \vdots & \vdots & \vdots \\ x_n & y_n & z_n & t_n \end{bmatrix}, \qquad F = \begin{bmatrix} h_1 \\ \vdots \\ h_n \end{bmatrix}, \qquad (1)$$

where the coordinate tensor is a $n \times 4$ tensor representing the space-time coordinates of the OMs in which there were

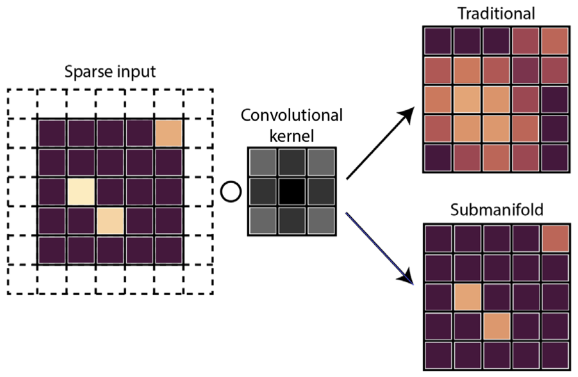

FIG. 2. Comparison of conventional and submanifold convolution with a Gaussian kernel. The submanifold convolution maintains the sparsity of the input, while the traditional convolution blurs the input, making it less sparse. In this example, a traditional convolution would require 18 or 25 matrix multiplications for sparse and nonsparse convolution, respectively, whereas the bottom image only requires three matrix multiplications.

a nonzero number of photon hits. The symbol n represents the total number of photon hits in the event, which is variable and can typically range from one to hundreds of thousands. The feature tensor contains the number of photon hits which occurred within a 1 ns time window on that OM, starting from the time indicated in the coordinate tensor. Nanosecond units were chosen for very fine timing resolution, as we aim to deploy the network on both low- and high-energy events. However, depending on the application, the 1 ns time window can be expanded to trade-off timing resolution for even better run-time and memory efficiency.

It is worth noting that each coordinate can be associated with any feature vector, offering flexibility for various configurations to encode neutrino telescope data. For instance, we can approach this as a 3-dimensional problem, where we only consider the spatial positions as coordinates and use the timing information as features. However, we decided to treat the timing as coordinates to take advantage of the time structure inherent in neutrino telescope data.

III. EVENT SIMULATION

Our benchmark case follows an ice-embedded IceCube-like geometry, where the OMs are spaced our approximately 125 meters horizontally and 17 meters vertically. The events used in this work are μ^- from ν_μ charged-current interactions. The initial neutrino sampling, charged lepton propagation, and photon propagation were simulated using the PROMETHEUS package [23]. The incident neutrinos have energies between 10^2 GeV and 10^6 GeV sampled from a power-law with a spectral index of -1. Since most of the events that trigger neutrino telescopes are downward-going cosmic-ray muons, we generated a downgoing dataset. Specifically, the initial momenta have zenith angles between 80° and 180°. It is worth noting that this definition of zenith angle is different from the convention which is typically used by neutrino telescopes, which take 0° to be downgoing. Internally, LeptonInjector [24] samples the energy, direction, and interaction vertex. PROPOSAL [25] then propagates the outgoing muon, recording all energy losses that happen within 1 km of the instrumented volume. PPC [26] then generates the photon yield from the hadronic shower and each muon energy loss, and propagates these photons until they either reach an OM or are absorbed. If a photon reaches an OM, the module ID, module position and time of arrival are recorded.

We then add noise in the style of [27] to the resulting photon distributions. This model accounts for nuclear decays in the OMs' glass pressure housings and thermal emission of an electron from a PMT's photocathode. The latter process strongly depends on the ambient temperature near the OM and varies between PMTs. Since this information is not publically available, we simplify the model and vary the thermal noise rate linearly from 40 Hz at the top of the detector to 20 Hz near the bottom, which approximately agrees with the findings from [27]. We then take the nuclear decay rate to be 250 Hz and generate a number of photons drawn from a Poisson distribution with a mean of 8 for each decay.

Before moving on, it is important to note that the photons generated in the previous steps are only tracked to the surface of the OM. In a full simulation of the detection process, one would need to simulate the electronics inside the OM, which could introduce timing uncertainties. Furthermore, the digitized signal reported by the e.g., the IceCube OMs must be unfolded to get the number of photons per unit time. These steps require access to proprietary information that is not available externally. Thus, we cannot include the effects from these detail detector performance in our simulation.

For example, the process by which IceCube unfolds the photon arrival times is described in [28]. They find that this process introduces a timing uncertainty typically on the order of 1 ns but that may grow up 10 ns under certain conditions. While this may affect our results, we expect the impact to be small since by group the photon arrival times into nanosecond-wide bins, we are introducing a timing uncertainty with a similar scale.

Once all photons have been added, we then implement a trigger criteria similar to the one described in [29]. This requires that a pair of neighboring or next-to-neighboring OMs see light in a 1 μs time window. If 8 such pairs are achieved in a 5 μs time window, we consider the trigger to be satisfied. As before, the exact details of the triggering process require access to proprietary information; however, the events which pass our trigger should be qualitatively similar to those which would trigger IceCube. After this cut, we are left with 462,892 events from 3 million simulated events, which we split between the training

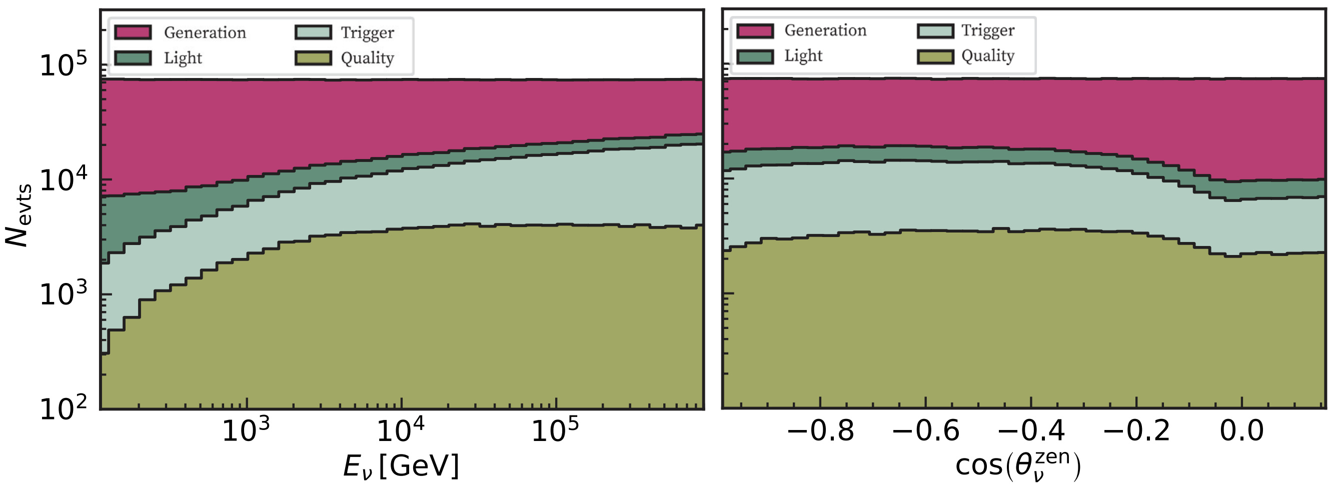

FIG. 3. Distributions of events in true energy and zenith. Left: The distribution of events used as a function of true neutrino energy. As expected, the generated distribution is flat when binned logarithmically since the generation was sampled according to a E_ν^{-1} distribution. Furthermore, the fraction of generated events which produce light, and the fraction of light-producing events which pass the trigger threshold increase with energy. Right: The same distribution as a function of true neutrino zenith angle. Once again the generated distribution is flat in the cosine of this angle, which is proportional to the differential solid angle. nearly flat, with slightly lower efficiency near the horizon.

and test data sets of 412,892 and 50,000 sizes respectively. One can see distributions of the events which pass this trigger as a function of true energy, zenith, and azimuth in Fig. 3.

In addition to the trigger-level dataset, we also evaluate the network performance on a dataset with further quality cuts, so that we can understand performance on events which are more likely to make it into a final analysis. In order to do this, we consider three quantities: $N_{\rm OM}$, $r_{\rm COG}$, and $R_{\rm ell}$. The first two quantities—the number of distinct OMs that saw the light and the distance between the charge-weighted center of gravity and the center of the detector—are fairly straightforward, but the last requires more explanation. To compute $R_{\rm ell}$, we fit a two-parameter ellipsoid to all OMs which saw light, and then take the ratio of the long axis to the short axis. A ratio close to one indicates a spherical events, whereas larger ratios indicate longer, tracklike events.

We perform straight cuts on these variables, requiring events have $N_{\rm OM} > 11$, $r_{\rm COG} < 400$ m, and $2 < R_{\rm ell} < 8$. The first cut removes low-charge events which are difficult to reconstruct, while the second removes "corner clipper" events caused by μ^- passing near the edge of the detector. The final cut on $R_{\rm ell}$ helps ensure that the events have a long lever arm for angular reconstruction.

These cuts reduce the split training and testing dataset sizes to 108585 and 13183 events, respectively. The spatial sparsity of these improved quality events is about ∼3%, as there are 154 OMs that were hit on average, out of the 5,160 total OMs in our example detector. For the trigger-level events, the spatial sparsity is about ∼2%. The time dimension adds another level of sparsity, as typical events can last tens of thousands of nanoseconds compared to the microsecond time window.

IV. PERFORMANCE

A. Training and architecture details

We utilize a ResNet-based architecture, taking advantage of residual connections between layers to promote robust learning for deeper networks. More details on the network architecture can be found in Fig. 4. A typical block of the network consists of a sparse submanifold convolution, followed by batch normalization and the parametric rectified linear unit (PReLU) activation function. The selection of the activation function was determined after examining prevalent alternatives, such as the conventional ReLU or a smooth approximation, like the SELU. Downsampling is performed using a stride 2 sparse submanifold convolution. We use the PyTorch deep learning framework and the `MinkowskiEngine` [30] library to implement the network.

This article focuses on the training of three distinct models. The objective of two of these models is the prediction of the three components of the neutrino directional pointing vector (X_ν, Y_ν, Z_ν), with one model trained on the trigger-level dataset and the other trained on the quality dataset. The directional vector is learned rather than the zenith and azimuth angles because of complications with azimuthal periodicity and undesirable boundary condition behavior at large or small angles. Another model was trained to infer the primary neutrino energy E_ν. The network is trained to predict the logarithmic energy, $\log_{10}(E_\nu)$, as they vary over a wide range of magnitudes.

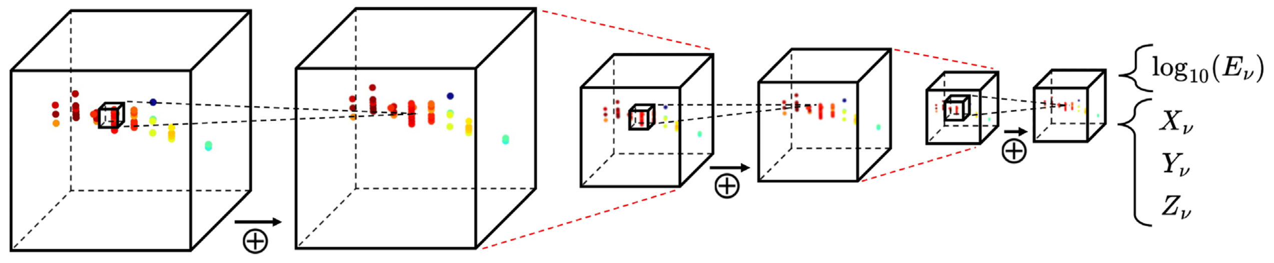

FIG. 4. Network architecture overview. The network accepts as input a 4D point cloud of photon OM hits, as shown by the colored points in the figure. The color indicates the timing (red is earlier, blue is later). Residual connections, denoted with ⊕, are used in between convolutions. Downsampling (dashed-red lines) is performed after a series of convolutions. The final layer of the network is a fully connected layer, which outputs the predicted quantity. In this work, we train the network to predict the logarithmic ν_μ energy, and the three components of the normalized ν_μ direction vector.

Each model was trained for 25 epochs using a batch size of 128 and the `Adam` optimizer. The initial learning rate was set at 0.001 and was reduced periodically during training. Dropout and weight decay are employed to prevent overfitting.

To train the energy reconstruction task, the `LogCosh` loss function is used, since it is more robust to outliers than the standard `MSE` loss. The loss function is defined as follows:

$$\mathcal{L}_E = \frac{1}{N}\sum_i^N \log\left(\cosh\left(x_i - y_i\right)\right), \tag{2}$$

where N is the number of events in the batch, x_i are the predictions, and y_i are the labels. For the angular reconstruction, an angular-distance loss function is used, namely,

$$\mathcal{L}_A = \frac{1}{N}\sum_i^N \arccos\left(\frac{\vec{X}_i \cdot \vec{Y}_i}{\|X_i\|\|Y_i\|}\right), \tag{3}$$

where $\vec{X}_i$ and $\vec{Y}_i$ are the predicted and true directional vectors, respectively.

B. Run-time performance

We evaluate the run-time performance of SSCNN in terms of the forward pass duration on both CPU and GPU hardware. The CPU benchmark is performed on a single core of an Intel Xeon Platinum 8358 CPU, while the GPU benchmark uses a 40 GB NVIDIA A100. As is generally the case for neural networks, running on GPU is preferred due to its superior parallel computation capabilities. Additionally, the use of sparse submanifold convolutions has greatly enhanced the GPU memory efficiency, enabling us to run larger batch sizes during inference. SSCNN can reconstruct direction at a rate of 9901 Hz on a 40 GB NVIDIA A100 GPU, while handling a batch size of 12,288 events simultaneously. This is fast enough to handle the expected ∼kHz current and planned large neutrino telescopes.

The run-time on a single-core CPU is slower and largely dependent on the number of photons hits in the event due to the limited parallel computation capabilities. However, SSCNN run-time on a CPU core is comparable to that of the likelihood-based angular method and is more consistent, as indicated by the lower standard deviation on the run-time distribution. The run-time results on both GPU and CPU are summarized in Table I.

C. Reconstruction performance

We first test SSCNN on reconstructing the direction of the primary neutrino. We measure performance using the angular resolution metric, which is calculated by taking the angular difference between the predicted and true directional vectors. Figure 5 shows the angular resolutions as a function of the true neutrino energy. Lower-energy events generally produce less photon hits, leading to a shorter lever-arm and, consequently, worse resolution. As expected, the trigger-level events are harder to reconstruct due to the lower light yield and the presence of corner-clipper events. On this event selection, SSCNN is able to reach under 4° median angular resolution on the highest-energy events. Enforcing the previously described quality cuts improves the results of the SSCNN by roughly 2°

TABLE I. Per event average run-time performance. The forward pass run times (mean ± STD) for SSCNN was evaluated on trigger-level events. A likelihood-based method for energy and angular reconstruction was included for reference [7]. It should be noted that likelihood-based methods usually require seeding or initial estimates, meaning the actual run-time is longer.

SSCNN angular (GPU)	**0.101 ± 0.003 ms**
SSCNN energy (GPU)	**0.103 ± 0.008 ms**
SSCNN angular (CPU)	**37.7 ± 53.4 ms**
SSCNN energy (CPU)	**30.6 ± 48.9 ms**
Likelihood angular (CPU)	36 ± 152 ms
Likelihood energy (CPU)	6.58 ± 23 ms

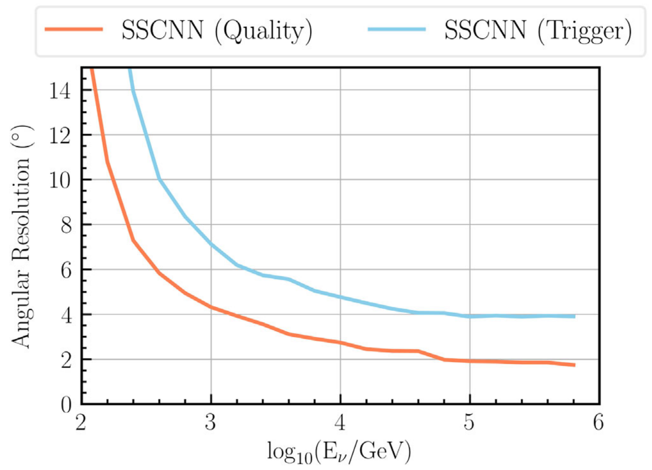

FIG. 5. Angular reconstruction performance as a function of the true neutrino energy. The angular resolution results are binned by the true neutrino energy, with the median taken from each bin to form the lines shown.

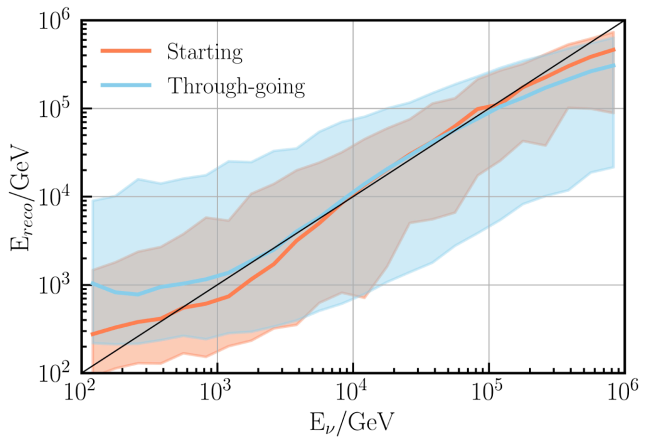

FIG. 6. Energy reconstruction performance at the trigger-level. The solid lines show the median of the predicted $\log_{10}(E_\nu)$, while the shaded regions are the 5% to 95% confidence level bands. The events in the test dataset are separated into starting and through-going events. The solid-black line serves as a reference for a perfect reconstruction.

across the entire energy range. This performance is comparable to or better than current trigger-level reconstruction methods used in neutrino telescopes. For example, the current trigger-level direction reconstruction at IceCube is done using the traditional LineFit algorithm [5], which has a median angular resolution of approximately 10° on raw data.

We also test SSCNN on reconstructing the energy of the primary neutrino. Figure 6 summarizes the energy reconstruction results. Events where the interaction point of the neutrino occurs outside the detector, known as through-going events, make up the majority of our dataset. As a result, predicting the neutrino energy has an inherent, irreducible uncertainty produced by the unknown interaction vertex and the muon losses outside of the detector. This missing-information problem leads to an intrinsic uncertainty in the logarithmic neutrino energy of approximately 0.4 for a through-going event. The behavior observed between 100 GeV and 1 TeV is due to the muon being in the minimally ionizing regime, up to around 700 GeV. Additionally, the network has a tendency to overpredict at the lowest energies, and underpredict at highest energies. This can be attributed to the artificial energy bounds on the simulated training dataset.

V. CONCLUSIONS

In this article, we have demonstrated the application of an SSCNN for event reconstruction on neutrino telescopes. We have shown that these networks are capable of maintaining competitive performance on the tasks of energy and angular reconstruction while running on the μs time scale. The speedup enables the SSCNN to process events at a rate well above that of the current neutrino telescopes trigger rate, which is expected to be representative of other neutrino telescopes currently operating or under construction, such as IceCube, KM3NeT, P-ONE, and Baikal-GVD. Reaching this threshold makes the SSCNN a feasible option for online reconstruction at the detector site where resources are limited and where first guesses of the energy and direction of the neutrino are made. As discussed in the introduction, this can have a substantial impact on current real-time analyses, where our first estimations can also be utilized in an alert-sending system, which will notify collaborators if the detector sees an interesting event. Additionally, these reconstructions can serve as seeds for more time-consuming reconstructions, and thus improving these first estimations will be beneficial to all subsequent analyses.

As a promising future direction, the exploration of training SSCNN for other event reconstruction challenges, such as morphology and particle classification, holds great potential. Such tasks could significantly benefit from the accelerated run-time provided by SSCNN.

ACKNOWLEDGMENTS

We thank Tong Zhu, Alfonso Garcia Soto, and Miaochen Jin for useful discussions. Additionally, we would like to thank David Kim for help with LineFit and the rest of the PROMETHEUS authors. C. A. A., J. L., and F. J. Y. are supported by the Faculty of Arts and Sciences of Harvard University. Additionally, F. J. Y. is supported by the Harvard Physics Department Purcell Fellowship.

[1] Arthur Roberts, The birth of high-energy neutrino astronomy: A personal history of the DUMAND project, Rev. Mod. Phys. **64**, 259 (1992).

[2] M. G. Aartsen *et al.*, Neutrino emission from the direction of the blazar TXS 0506 + 056 prior to the IceCube-170922A alert, Science **361**, 147 (2018).

[3] R. Abbasi *et al.*, Evidence for neutrino emission from the nearby active galaxy NGC 1068, Science **378**, 538 (2022).

[4] A. Albert *et al.*, Review of the online analyses of multi-messenger alerts and electromagnetic transient events with the ANTARES neutrino telescope, J. Cosmol. Astropart. Phys. (to be published), https://inspirehep.net/literature/2182044.

[5] M. G. Aartsen *et al.*, Improvement in fast particle track reconstruction with robust statistics, Nucl. Instrum. Methods Phys. Res., Sect. A **736**, 143 (2014).

[6] J. A. Aguilar *et al.*, A fast algorithm for muon track reconstruction and its application to the ANTARES neutrino telescope, Astropart. Phys. **34**, 652 (2011).

[7] Mirco Hünnefeld, Online reconstruction of muon-neutrino events in IceCube using deep learning techniques, Master thesis, Technische Universität Dortmund, 2017.

[8] J. García-Méndez, N. Geißelbrecht, T. Eberl, M. Ardid, and S. Ardid, Deep learning reconstruction in ANTARES, J. Instrum. **16**, C09018 (2021).

[9] M. G. Aartsen *et al.*, Very high-energy gamma-ray follow-up program using neutrino triggers from IceCube, J. Instrum. **11**, P11009 (2016).

[10] Baikal-GVD Collaboration, BAIKAL-GVD: Gigaton volume detector in Lake Baikal, University of Amsterdam, FNWI, 2012, https://baikalgvd.jinr.ru/BAIKAL-GVD_En.pdf.

[11] Bardo Bakker, Trigger studies for the Antares and KM3NeT neutrino telescopes, 2011, https://indico.cern.ch/event/272703/attachments/490148/677542/BardoTriggerStudy_v2.pdf.

[12] Felix Yu and Jeffrey Lazar, https://github.com/felixyu7/nt_ssnet (2022).

[13] Alexander Radovic, Mike Williams, David Rousseau, Michael Kagan, Daniele Bonacorsi, Alexander Himmel, Adam Aurisano, Kazuhiro Terao, and Taritree Wongjirad, Machine learning at the energy and intensity frontiers of particle physics, Nature (London) **560**, 41 (2018).

[14] A. Aurisano, A. Radovic, D. Rocco, A. Himmel, M. D. Messier, E. Niner, G. Pawloski, F. Psihas, A. Sousa, and P. Vahle, A convolutional neural network neutrino event classifier, J. Instrum. **11**, P09001 (2016).

[15] R. Acciarri *et al.*, Convolutional neural networks applied to neutrino events in a liquid argon time projection chamber, J. Instrum. **12**, P03011 (2017).

[16] R. Abbasi *et al.*, A convolutional neural network based cascade reconstruction for the IceCube Neutrino Observatory, J. Instrum. **16**, P07041 (2021).

[17] P. Bagley *et al.*, KM3NeT: Technical design report for a deep-sea research infrastructure in the Mediterranean sea incorporating a very large volume neutrino telescope, 2009, https://www.km3net.org/wp-content/uploads/2023/05/KM3NeT_DS_TDR-published-in-2010.pdf.

[18] J. Ahrens *et al.*, IceCube preliminary design document, 2001, https://icecube.wisc.edu/wp-content/uploads/2020/11/IceCubeDesignDoc.pdf.

[19] Matteo Agostini *et al.*, The pacific ocean neutrino experiment, Nat. Astron. **4**, 913 (2020).

[20] Benjamin Graham and Laurens van der Maaten, Submanifold sparse convolutional networks, arXiv:1706.01307.

[21] P. Abratenko *et al.*, Semantic segmentation with a sparse convolutional neural network for event reconstruction in MicroBooNE, Phys. Rev. D **103**, 052012 (2021).

[22] Laura Dominé and Kazuhiro Terao, Scalable deep convolutional neural networks for sparse, locally dense liquid argon time projection chamber data, Phys. Rev. D **102**, 012005 (2020).

[23] J. Lazar *et al.*, PROMETHEUS: An open-source neutrino telescope simulation, 2023, https://inspirehep.net/literature/2655303.

[24] R. Abbasi *et al.*, LeptonInjector and LeptonWeighter: A neutrino event generator and weighter for neutrino observatories, Comput. Phys. Commun. **266**, 108018 (2021).

[25] Jan-Hendrik Koehne, Katharina Frantzen, Martin Schmitz, Tomasz Fuchs, Wolfgang Rhode, Dmitry Chirkin, and J Becker Tjus, Proposal: A tool for propagation of charged leptons, Comput. Phys. Commun. **184**, 2070 (2013).

[26] Dmitry Chirkin, PPC standalone code, https://icecube.wisc.edu/~dima/work/WISC/ppc/ (2020).

[27] Michael James Larson, Simulation and identification of non-Poissonian noise triggers in the IceCube neutrino detector, Ph.D. thesis, Alabama University, 2013.

[28] M. G. Aartsen *et al.*, Energy reconstruction methods in the IceCube Neutrino Telescope, J. Instrum. **9**, P03009 (2014).

[29] M. G. Aartsen *et al.*, The IceCube Neutrino Observatory: Instrumentation and online systems, J. Instrum. **12**, P03012 (2017).

[30] C. Choy, J. Gwak, and S. Savarese, 4d spatio-temporal convnets: Minkowski convolutional neural networks, In *Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA* (IEEE Computer Society, New York, 2019).