PROCEEDINGS OF SCIENCE

Low Power Real-Time Muon Track Reconstruction for Water(Ice) Cherenkov Neutrino Telescopes

Miaochen Jin[a,*] **and Carlos A. Argüelles**[a]

[a] *Harvard University,*
18 Oxford St., Cambridge, USA

E-mail: miaochenjin@g.harvard.edu

In recent years, there have been enormous improvements in event reconstruction and signal identification for neutrino experiments via deep learning methods. Despite the success in accuracy of current ML algorithms, the power-efficiency of such methods calls out for improvement for any hope at realizing real time machine learning data processing. Solutions are two-fold: either resorting to software, using a faster algorithm and gaining absolute speed under the same power consumption, or bringing into play a hardware alternative that consumes less energy in performing the same operations. In this work, we turn to the hardware alternative, presenting the first attempt at accelerating deep learning methods for muon event reconstruction on Tensor Processing Units (TPU's). We use an LSTM-based recursive neutral network with CNN-based data encoding that is compatible with Google TPU hardware requirements. We show that this algorithm is capable of achieving similar angular resolution in reconstruction compared to State of the Art ML peers, and reaches a performance per watt of 100 Hz/Watt on a TPU accelerator. This opens up an entire world of chances at integrating machine learning capacity into detectors and electronics that go deep into even the most power-restricted environments.

Corresponding authors: Miaochen Jin[1*]
[1] *Department of Physics and Laboratory for Particle Physics and Cosmology, Cambridge 02138, MA, United States*

[*] Presenter

The 38th International Cosmic Ray Conference (ICRC2023)
26 July – 3 August, 2023
Nagoya, Japan

[*]Speaker

PoS(ICRC2023)1081

1. Introduction

Recently, there has been many attempts to improve reconstruction algorithm performance for neutrino telescopes in many fronts, including low energy events reconstruction using graph neural networks in [1] and faster reconstruction using sparse architectures in [2]. In this article, we further explore the possibility to run real time reconstruction algorithms under stringent power consumption restrictions, introducing the first attempt at accelerating neutrino event reconstruction on edge computing devices using a recursive neural network (RNN) method with a convolutional encoding. We will present the performance of our method on both water and ice based neutrino telescopes, providing a general picture of how this technology could be integrated into any such detector alike. In Section 2 we introduce the detectors and data simulation used in this work; in Section 3 we briefly discuss the various hardware architectures we test our algorithm on and their reported power consumption specifications, and lay out specifically the constraints of the TPU hardware; in Section 4 we motivate the data pre-processing and network architecture in the context of edge computing hardware limitations, and explain our methods and solutions, including discussions regarding TPU deployment and quantization caveats; in Section 5 we evaluate the accuracy and power performance of our approach; finally, in Section 6 we discuss the various venue this work opens up, and encourage further exploration in the direction of low power computing in neutrino detectors.

2. Detector and Data Simulation

In this work, we will be testing the performance and accuracy of our network on two example detectors of the same optical module (OM) geometry in water and in ice, to which we assigned the names of WaterHex and IceHex respectively, where the hexagonal geometry is inpired by that of IceCube. Inter-string distance is set to 100 meters, and inter-OM distance along a string is set to 17 meters. A total of 114 strings are deployed, with each string containing 60 OMs, summing to a total of 6840 OMs. Data simulation used as training, testing and validation datasets are generated using the Prometheus package [3].

In Figure 1 we show the energy and cosine zenith distribution of the simulated events at different quality cut levels, the levels are defined as follows:

- Generation level: Neutrinos are injected using LeptonInjector to the target detector. Events are range-injected: this means at injection level, we only include events whose closest point of approach to the target distance lies within the detector geometry volume plus two sccaterring lengths of photon in the target detector medium. This choice of injection method results in a considerable fraction of event at generation level to be useful for training, while still considering the physical-ness of the source.

- Light level: this level includes all the events that resulted in some light deposit in the target detector.

- Trigger level: The detector is triggered when at least 8 hard local coincidences are detected within the detector, where each hard local coincidence is defined at two photon deposits in

Hardware Architecture	Reported Efficiency	Total Power	ML Accelerator Power
A100 GPU	165 TFLOPS	2400W	400W
RTX3080 GPU	29.8 TFLOPS	1000W	320W
M1 Pro chip	5.3 TFLOPS	100W	15W
Google Edge TPU	4TOPS	3W	2W

Table 1: Summary of the hardware utility data used in evaluation of the model deployed by this work. The power consumption sepcifications listed above are approximate values of peak consumption, and therefore should be taken as a rough estimate and indication of the scale of the computing system to some extend of accuracy. Even considering this limitation in exactness, the comparison listed above still shows significant differences between different classes of hardware architectures.

one or two optical modules within 150 meters of range with each other within a time window of 5 nanoseconds.

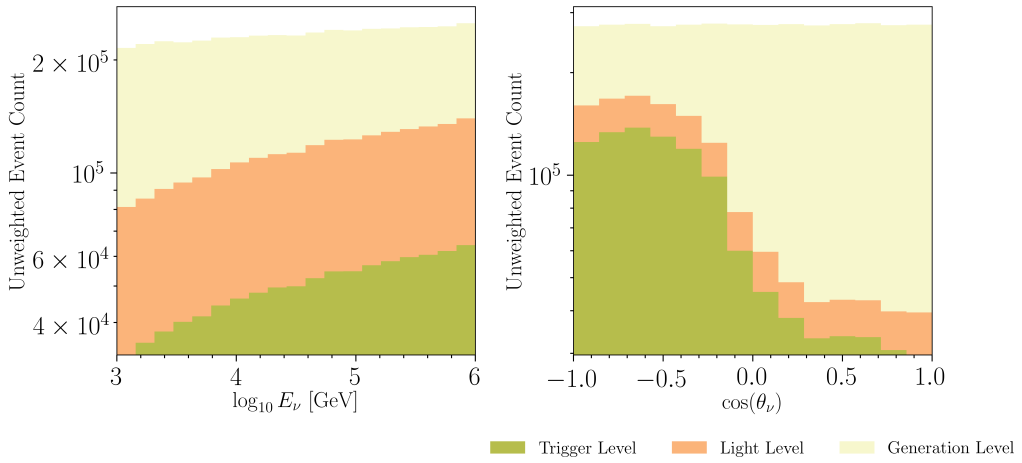

Figure 1: Energy and Zenith angle distribution of data set at various selection levels

3. Hardware Setup

For this work, we evaluate the performance, both accuracy and power efficiency, of our method on thre different classes of hardware architectures spanning three different orders of magnitude in maximum power consumption. In Table 1 we show these hardware architectures: specifically, we show the reported Trillion Operations per Second (TOPS) / Trillion Floating Point Operations per Second (TFLOPS), total power of the entire hardware form factor, and fraction of the power going into the the ML accelerator component.

4. Software Methods

4.1 Data and Network Input

Water(Ice)-Cherenkov detector data consists at the lowest level PMT waveforms recorded by optical modules in water or ice. This yields a set of waveforms $\{q_\alpha(t)\}_\alpha$ corresponding to each OM located at $(x_\alpha, y_\alpha, z_\alpha)$. In order for machine learning algorithms to work with these data, at least some extent of preprocessing is needed, the specific method of which differs per ML algorithm and architecture employed. We take IceCube as an example, where the prominent reconstruction algorithm DNN reco[4] is a Convolutional Neural Network approach, where data is preprocessed to form a 4 dimensional tensor, treated equivalent to an "image." Spatial coordinates are embedded into a 3 dimensional tensor $(x_\alpha, y_\alpha, z_\alpha) \rightarrow V^{(i_\alpha, j_\alpha, k_\alpha)}$, while for the time component, the PMT waveforms $q_\alpha(t)$ are extracted to give 9 distinct features from the waveform of a DOM across the time of an event. This results in an input of the format $\vec{T}_i \oplus \vec{x}_i$ where the distinct time parameters are given as

$$T_i = (c_{\text{total}}, c_{500\text{ns}}, c_{100\text{ns}}, t_{\text{first}}, t_{\text{last}}, t_{20\%}, t_{50\%}, t_{\text{mean}}, t_{\text{standard}})$$

With such an approach, the data is primarily seen by the network as an image, where on each “pixel," now an OM location, the information of “color channels" is replaced by the distinct time parameters. In this work, we rethink the formulation of this problem, phrasing it primarily as a time series analysis and utilize Recursive Neural Network (RNN) architectures, this choice wil be elaborated upon in Section 4.3. We take the detector data after pulse extraction to obtain the individual hit times of photons on OMs, $\{H_i = (t_i, x_i, y_i, z_i)\}_{i=1}^{N}$, and group them imagining that we are taking snapshots of the event at a fixed timestep. For an event consisting of N hits that spans T nanoseconds, we break it into 30 separate frames each with $T/30$ nanoseconds, containing an aggregate of $\{N_j\}_{j=1}^{30}$ hits with $\sum_j N_j = N$. This results in 30 distinct 3 dimensional arrays T_j, with each entry $T_j^{(x,y,z)}$ encoding the number of total hits on the corresponding DOM with in the time window, with $\sum_{(x,y,z)} T_j^{(x,y,z)} = N_j$. This method of data encoding can be understood visually from the preprocessing portion of the network illustration in Figure 2.

4.2 Edge TPU Limitation and Methodology Adaptation

While the Edge TPU is capable of runninng inference at a very high speed and low power consumption, being a edge computing device implies limitations on network architecture as well as numerical accuracy. In order to fully understand the design of the network and data preprocessing, we need to first elaborate theses restrictions.

For network architecture limitations, only a subset of commonly used ML-relevant operations is allowed [5]. On top of this, all tensors have to be kept at or below dimensionality $D \leq 3$, thereby also implying that any convolutional operation can only reach up to 2 dimensional. For numerical accuracy limitations, quantization is required to ensure all weights are computed to 8-bit accuracy [5]. We adapt our architecture as well as employ post-training quantization to overcome these limitations: these will be explored in Section 4.3 and Section 4.4 respectively.

4.3 Recursive Network with Convolutional Embedding

In Fig. 2 we show the architecture of the network this article develops. The input as described in section Section 4.1 is first fed into $T = 30$ identical Convolutional Neural Networks (CNN). Each input into the CNN is three dimensional and has dimensions reflecting the optical modules' grid geometry padded into a regular cuboid shape. Here the the depth of the cuboid is treated as different channels, leaving the task as a 2D CNN network, where a one dimensional output is yielded. This step serves as an embedding, as the resulting $T = 30$ vectors are then used as inputs into a Long Short Term Memory (LSTM) cell that predicts the output.

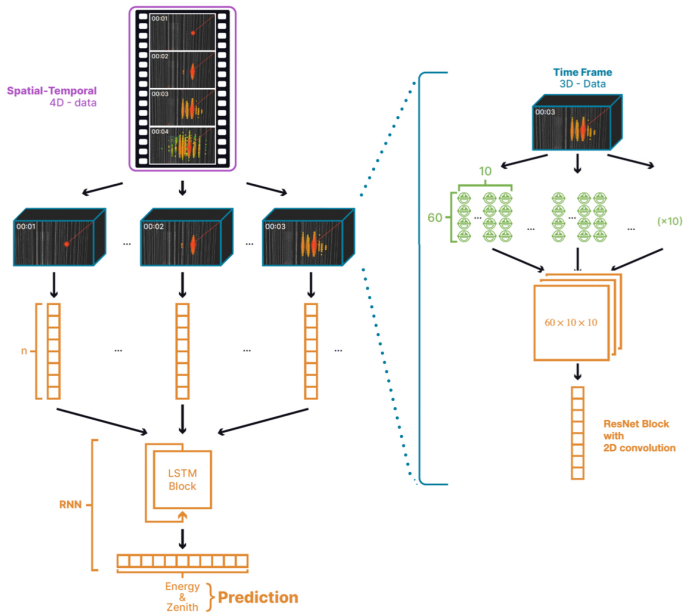

Figure 2: Recursive Neural Network with Convolutional Encoding

4.4 TPU Deployment Caveats and Quantization

There are a couple caveats in the design and quantization of the network, which we will discuss and explore here.

Firstly, we would like to note the difference between training on a GPU setup and inferencing on a TPU setup and how that leads to the particular design of the CNN Embedding layer. While training on a GPU architecture, the $T = 30$ temporal slices can be simultaneously handled with no additional overhead time usage, since TPU can only handle vector dimensions of size $D \leq 3$ as mentioned previously, this requires 30 full iterations, taking 30 times the amount of time compared to embedding only one temporal slice of data. This introduces an important hyperparameter: the number of temporal slices. In general we have noticed a trade-off between efficiency and accuracy, and $T = 30$ is chosen as a sensible candidate for now. This is also the reasoning behind taing one dimension of each temporal slide cuboid and treating it as the channel dimension in the CNN input. If this were not done, then we would need to achieve a 3 dimensional convolution by two subsequent 2 dimensional convolutions, which, while not costing more time in a GPU setup, will suffer in efficiency in a multiplicative manner on a TPU setup.

Secondly, we would like to discuss the quantization for such a network. In order to benefit from the full power of TPU acceleration, we need to quantize the network to uint-8 accuracy. We choose to perform post training quantization as discussed by Google in [6]. However, since our model contains two subsections, namely the CNN embedding section and the LSTM prediction section, we need the intermediate embedded outputs in order to quantize both networks. For the quantization of the CNN embedding section, we take the embedded temporal slice cuboids and treat these one dimensional vectors as the output for the CNN to quantize the CNN embedding network. Then, we use these intermediate results as the input into the LSTM section and quantize the latter half of the network. Current working progress is being performed on maximally preserving accuracy through the quantization progress. For this reason we will not discuss the quantized network's accuracy in this proceeding, but focus on the efficiency after quantization and deployment on TPU.

5. Results

5.1 Accuracy Performance of Network

In Figure 3 we show the median of the angular, zenith and azimuthal reconstruction errors for the detector in Ice medium. We have seen similar performances in water medium, but due to simulation time contraints we were not able to produce as large a dataset for the reconstruction in water. Therefore unfortunately for the moement we will report only on the detector geometry in ice. Due to differences in geometry setup, this is not to be compared directly, heads on, with current and next generation water(ice)-Cherenkov detectors' reconstruction accuracy, but nevertheless we can see approximately similar results. This shows that although the design of the network seems somehow unnatural in order to accomodate for TPU hardware architecture limitations, it is still a sensible design that shares the same level of accuracy performance with most currently used machine learning algorithms.

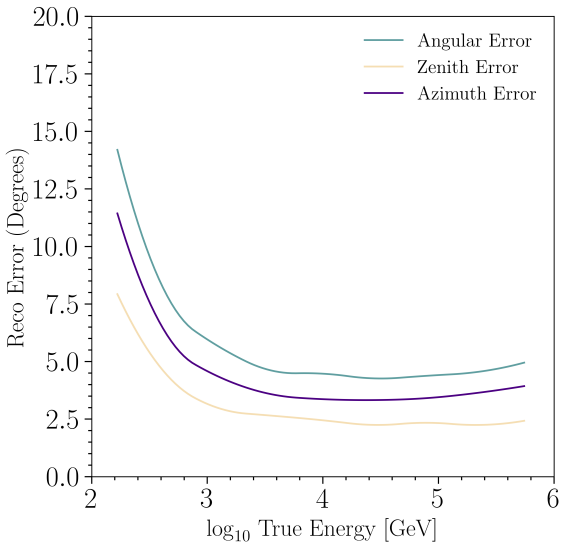

Figure 3: Neutrino Angular Reconstruction Error on full accuracy network

5.2 Efficiency Performance of Network

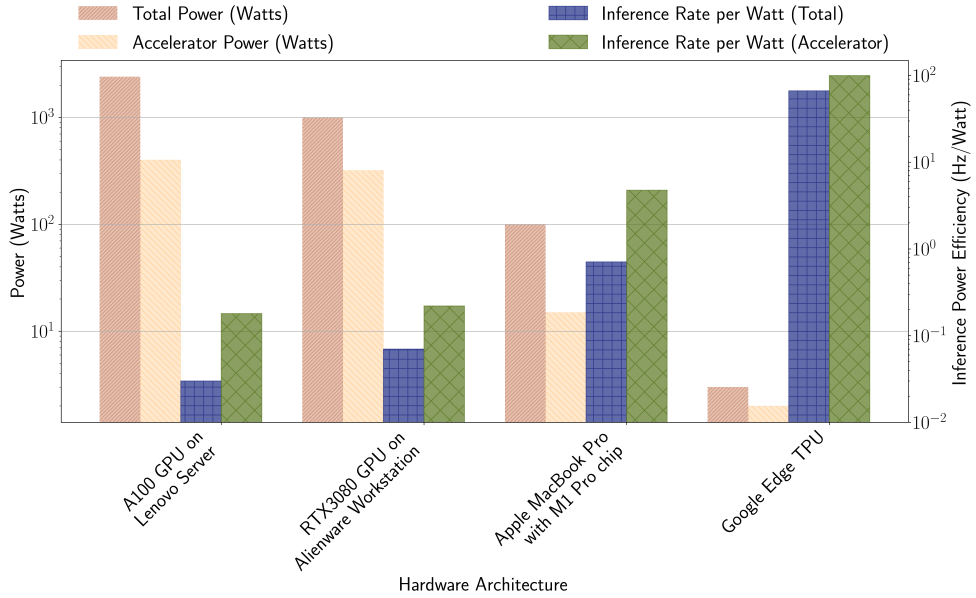

Figure 4: Power specification and efficiency of various hardware architectures. For each hardware architecture, the left two bars show their listed power consumption, the specifications are also listed in Table 1, and the right two bars show their power efficiency when running inference on batch sizes of 1. "Total" counts the power of the entire architecture, while "accelerator" counts just the power consumption of the ML accelerator unit, which could be a GPU or a TPU.

In Figure 4 we visualize the efficiency performance of the network under various architectures, especially the energy usage for each inference prediction on these hardware architectures. It is very important to note that, as shown in the table, while large scale computing systems and huge memory graphic cards enables large batch size parallelization and therefore much better efficiency in training, if we are aiming at a real time reconstruction system, we should always be considering inferences of batch size 1, where large scale architectures loose all their advantages, and we need to find smaller, more portable low power alternatives. The specific inference times and prediction per watt data can be found in Table 2.

Hardware Architecture	1 Prediction	Inference Speed (100/batch)	1 Prediction/Watt (Total)	1 Prediction/Watt (Accelerator)
A100 GPU	14ms	0.5ms	0.03 Hz/Watt	0.18 Hz/Watt
RTX3080 GPU	10ms	1.7ms	0.07 Hz/Watt	0.22 Hz/Watt
M1 Pro chip	14ms	3.8ms	0.71 Hz/Watt	4.76 Hz/Watt
Google Edge TPU	5ms	N/A	66.7 Hz/Watt	100 Hz/Watt

Table 2: Performance per Watt and total energy consumed for this network on different architectures.

6. Summary and Outlook

In this article, we have shown the accuracy and power efficiency of the novel approach deploying LSTM on an Edge TPU for water- and ice-Cherenkov neutrino telescope event reconstruction. We have demonstrated the capability of low power computing at a competent accuracy. We would like to motivate further exploration into this direction by noting some potential improvements to and applications of this approach. On the one hand, while this work utilizes simulated data that contains post-pulse-extraction detector data, by using the lower level PMT pulses, the advantage of the LSTM approach will be further amplified since it will be dealing with more time series-like input; this is even potentially capable to circumference the necessity of a convolutional embedding, further speeding up inference and thereby improving power efficiency. On the other hand, low power consumption implies the possibility of integrating the edge computing hardware into the optical modules, providing a chance at real time machine learning based processing of multiple PMT waveforms as an event selection alternative. There are definitely various other improvements and applications waiting out there to be explored. We would like to encourage more future work along this newly opened up venue.

References

[1] R. Abbasi, M. Ackermann, J. Adams, N. Aggarwal, J. Aguilar, M. Ahlers, M. Ahrens, J. Alameddine, A. Alves, N. Amin, *et al.* *Journal of Instrumentation* **17** no. 11, (Nov, 2022) P11003.

[2] F. J. Yu, J. Lazar, and C. A. Argüelles.

[3] J. Lazar, S. Meighen-Berger, C. Haack, D. Kim, S. Giner, and C. A. Argüelles.

[4] R. Abbasi, M. Ackermann, J. Adams, J. Aguilar, M. Ahlers, M. Ahrens, C. Alispach, A. Alves, N. Amin, R. An, *et al.* *Journal of Instrumentation* **16** no. 07, (Jul, 2021) P07041.

[5] "Edge tpu supported operations." https://coral.ai/docs/edgetpu/models-intro/#supported-operations. Retrieved: 05-25-2023.

[6] "Edge tpu quantization." https://coral.ai/docs/edgetpu/models-intro/#quantization. Retrieved: 05-25-2023.