Scaling the training of particle classification on simulated MicroBooNE events to multiple GPUs

A Hagen, E Church, J Strube, K Bhattacharya, and V Amatya
Pacific Northwest National Laboratory, Richland, WA, USA
E-mail: alexander.hagen@pnnl.gov

Abstract. Measurements in Liquid Argon Time Projection Chamber (LArTPC) neutrino detectors, such as the MicroBooNE detector at Fermilab [1], feature large, high fidelity event images. Deep learning techniques have been extremely successful in classification tasks of photographs, but their application to LArTPC event images is challenging, due to the large size of the events. Events in these detectors are typically two orders of magnitude larger than images found in classical challenges, like recognition of handwritten digits contained in the MNIST database or object recognition in the ImageNet database. Ideally, training would occur on many instances of the entire event data, instead of many instances of cropped regions of interest from the event data. However, such efforts lead to extremely long training cycles, which slow down the exploration of new network architectures and hyperparameter scans to improve the classification performance. We present studies of scaling a LArTPC classification problem on multiple architectures, spanning multiple nodes. The studies are carried out on simulated events in the MicroBooNE detector. We emphasize that it is beyond the scope of this study to optimize networks or extract the physics from any results here. Institutional computing at Pacific Northwest National Laboratory and the *SummitDev* machine at Oak Ridge National Laboratory's Leadership Computing Facility have been used. To our knowledge, this is the first use of state-of-the-art Convolutional Neural Networks for particle physics and their attendant compute techniques onto the DOE Leadership Class Facilities. We expect benefits to accrue particularly to the Deep Underground Neutrino Experiment (DUNE) LArTPC program, the flagship US High Energy Physics (HEP) program for the coming decades.

1. Introduction

Use of convolutional networks to analyze time projection chamber data is often performed on cropped data because of large image sizes. Training and inference on uncropped TPC data is desired to minimize physics information loss before training. The high fidelity and large size of the image data requires scaling of computing resources past the 1s to 10s and 100s of GPUs.

1.1. The MicroBooNE Detector and data format used

This work uses data from a toy Monte Carlo simulation resembling MicroBooNE experiment data [1]. MicroBooNE is a 170 tonne liquid argon time projection chamber (LArTPC) with the express interest of analyzing neutrino physics. Readout of MicroBooNE consists of 2 induction planes with 2400 wires each and 1 collection plane with 3156 wires. Readout occurs every 4.8 ms (which is 2.2× the TPC drift time) for 9600 digitizations.

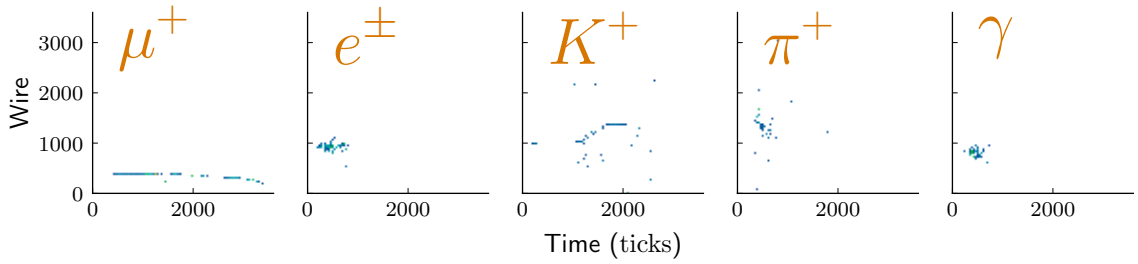


Figure 1: Example of simulated single particle interactions for classification.

This work presents classification on simulated single particle events in a format similar to that of MicroBooNE [1]. GEANT4 Monte Carlo particle transport code was used to simulate interactions with the wires in a LArTPC. The seed particles were single particles of type μ^+, e^+, e^-, K^+, π^+, and γ. The interactions with the collection plane were then tallied across 3200 time ticks and padded with zeros to increase the time dimension to 3600. The collection plane dimension was also padded with zeros to reach size 3600. The final simulated data was a 3600×3600 "image", with examples for each particle shown in Figure 1.

Previous work showed not only decreased training time for simple convolution networks on 2 to 14 GPUs (on PNNL's Institutional Computing), but improved performance, achieving lower losses with 14 GPUs versus with only 1 [2]. This work extends upon that scaling study, moving from 1 to 10s of GPUs to the 100s.

2. Methodology

A simplified convolutional neural network (CNN) model was developed for testing[2]. As shown in figure 2, the network architecture used two subsequent blocks consisting of convolutional layers with kernel size 5, padding of 2, and exponential linear unit (ELU) activation; followed by max pool layers of pooling size 5. Then, two more subsequent blocks of convolutional layers with kernel size 4, padding of two, and ELU activation followed by max pooling layers of pool size 4. These four blocks have increasing numbers of convolutional filters with the scheme of 1, 10, 64, 128, 256. After these four blocks, a linear layer with 20736 inputs, 32 outputs and ELU activation was appended. A linear layer with 32 inputs, 5 outputs, and softmax activation, one output for each of the particle types (e^+ and e^- were grouped together), finished the network. For a dense data representation, this network architecture was implemented using `Torch`'s `Conv2d` and `MaxPool2d` layers and is hereafter called `JishNet`. For a sparse data representation, discussed further in section 2.2, this network used `SparseConvNet`'s `SubmanifoldConvolution` and `MaxPooling` layers and is hereafter referred to as `SCNet`. It should be noted that the padding behavior of the convolutions in `JishNet` and `SCNet` differ subtly.

2.1. Resources

Many industrial applications of convolutional networks can apply 10s to 100s of GPUs to speed training and enable fast iteration in network design [3, 4].

2.1.1. Hardware This work presents one of the first known instances of scaling a physics classificaton problem to this scale on Oak Ridge National Laboratory Leadership Computing

[1] Permission from MicroBooNE collaboration was granted to focus on compute resources and performance studies

[2] It is stressed that this model was generated to provide model results while focusing on studying the computatonal resources. The CNN itself could be improved in numerous ways, but that is past the scope of this work.

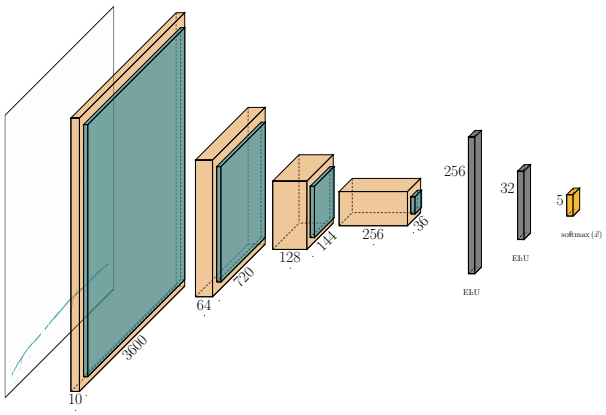


Figure 2: Design of "`JishNet`" and "`SCNet`" convolutional networks used for all testing presented on this poster

Facility's *SummitDev* computer. *SummitDev*, which is a test stand for the eventual *Summit* computer, was used in this study. The *SummitDev* machine boasts 50 nodes, each with 2 22-core IBM Power8NVL CPUs and 4 NVIDIA P100 GPUs. The interconnections between nodes are Mellanox EDR 100G InifiBand; interconnections between GPU to host node are NVLink. It should also be noted that there is a 4 hour time limit to all *SummitDev* jobs.

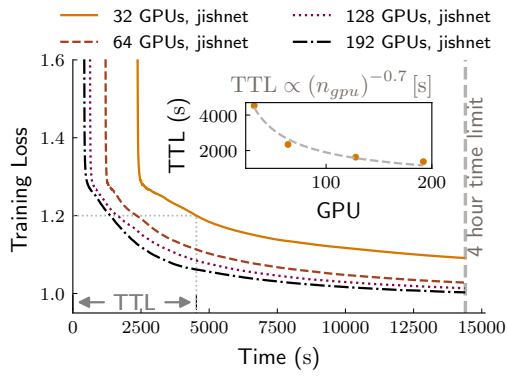
2.1.2. Software There is an embarassment of options for the development of neural networks and data parallel training of these neural networks. While previous scaling studies used Pacific Northwest National Laboratory's `MaTEx` [2, 5] code for data parallel training, this work was based on the use of `Torch` [6] for neural network definition and training, and `Horovod` [4] for data parallelization.

Compression was required when loading the dataset into CPU memory due to their size and number, and `Blosc` [7] was therefore used when loading images from file. At training time, a minibatch of these images were uncompressed from the host memory and transferred to GPU memory. Due to the large size of each datum, only 7 images alongsize the `JishNet` model could fit into a single NVIDIA P100's memory. Thus, all training was performed with a minibatch size of 7. After training of a single minibatch on all GPUs allocated for each training instance, `Horovod`'s `allreduce` function was used to transfer and allocate all gradients to the head node. This has the effect of generating a batch size of $7 \times N_{GPU}$. The loss and accuracy local to each GPU was written to disk before this `allreduce` operation.

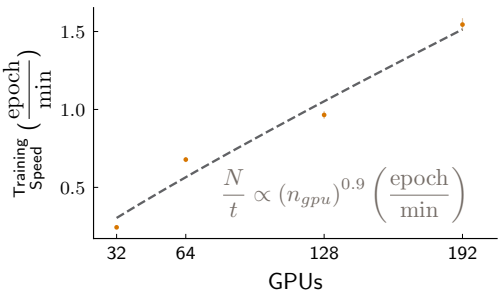
The use of many GPUs significantly affects the training dynamics of the CNN in this training example. A prolonged study was performed examining the training dynamics of this task using stochastic gradient descent (SGD) optimizer and warmup and scaling suggested in Goyal [8]. The SGD optimization study ultimately underperformed, so other options were explored. Distributed Adam optimization was tested and ultimately successful; thereafter, all results used Adam optimization [9]. The training dynamics were highly sensitive to the Adam optimizer parameters, final parameters were $lr = 0.001$, $\beta = [0.9, 0.999]$, $\epsilon = 1 \times 10^{-8}$, and $decay = 0.01$. Figures 3a and 4a illustrate the expected training behavior and verify the use of the Adam optimizer.

2.2. Sparse

Most of the data passed to the dense convolutional network is zeros, as no events happen in large parts of the TPC. Sparse convolutional networks are a more efficient network architecture



(a) Training dynamics of `JishNet` using increasing number of GPUs on *SummitDev*. Time to a loss (TTL) of 1.2 is shown in the inset.



(b) Speedup of training `JishNet` with increasing number of GPUs on *SummitDev*.

Figure 3: Results from training the dense convolutional network `JishNet` using various numbers of GPUs on *SummitDev*.

for LArTPC event classificaton. After conversion from a dense format to a sparse data format, training using `SubmanifoldConvolution` layers from Facebook's `SparseConvNet` [10] codebase and the modified CNN `SCNet` was performed.

3. Results

Results show that increasing the number of GPUs for both dense (`JishNet`) and sparse (`SCNet`) CNNs was successful in decreasing the training time on large images in large datasets.

3.1. Dense

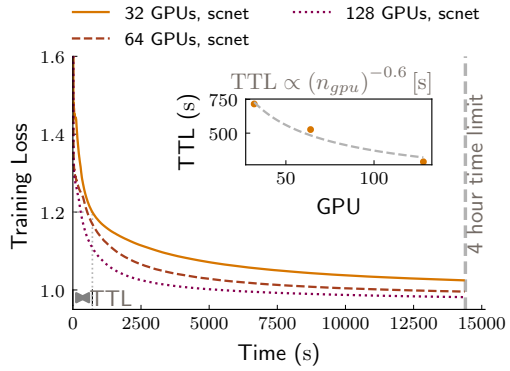
Scaling to multiple GPUs shows an approximately linear speedup with respect to the number of GPUs, as shown in Figure 3b. Figure 3a shows the loss versus training wall time with increasing GPUs. With more GPUs, a lower loss can be achieved in the allotted 4 hours, and the time to a loss (TTL) of 1.2 decreases with the number of GPUs used (shown in inset).

In Figure 3a, there is a marked difference in the time at which the first loss was reported with different number of GPUs. This time, the data loading time (DLT), is affected by how many images must be loaded into CPU before beginning training. Each training example was performed on 15,000 images, so each CPU had to load and compress $^{15{,}000}/_{N_{GPU}}$ into memory. Optimization of this compression routine is left for future work. It is hypothesized that the TTL speedup is sublinear ($\propto N_{GPU}^{0.7}$) because of this DLT effect.

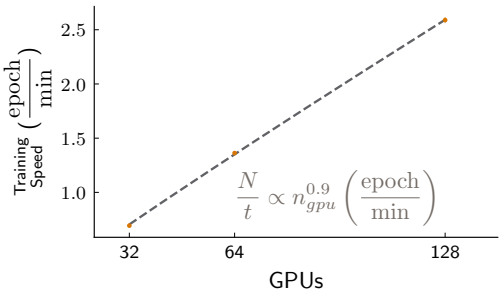
3.2. Sparse

`SCNet` shows interesting behavior during training compared to `JishNet`. Figure 4b again shows an approximately linear speedup when trained on increasing numbers of GPUs, however the highest speed is ~ 2.5 $^{\text{epochs}}/_{\text{min}}$, which is 60 % faster than `JishNet`. However, the TTL for `SCNet`, shown in Figure 4a is 4× lower than that for `JishNet`, shown in Figure 3a.

Figure 4a shows comparable loss to `JishNet` after the full training set, however there are several interesting characteristics. Compared to Figure 3a, there is very little evident DLT in Figure 4a. Despite this, the TTL speedup is only $\propto N_{GPU}^{0.6}$.



(a) Training dynamics of `SCNet` using increasing number of GPUs on *SummitDev*. Time to a loss (TTL) of 1.2 is shown in the inset.



(b) Speedup of training `SCNet` with increasing number of GPUs on *SummitDev*.

Figure 4: Results from training the sparse convolutional network `SCNet` using various numbers of GPUs on *SummitDev*.

Table 1: Confusion matrices from training of `JishNet` and `SCNet` with 128 GPUs for 4 hours.

(a) Confusion matrix for `JishNet` trained with 15000 images for 4 hours on 128 GPUs.

		Predicted				
		γ	$e\pm$	$\mu+$	$\pi+$	$K+$
True	γ	66.7%	33.0%	0.0%	0.0%	0.3%
	$e\pm$	1.6%	98.0%	0.0%	0.4%	0.0%
	$\mu+$	0.0%	0.1%	98.9%	0.0%	1.0%
	$\pi+$	1.6%	1.7%	1.3%	37.5%	57.8%
	$K+$	0.6%	0.1%	2.1%	26.3%	71.0%

(b) Confusion matrix for `SCNet` trained with 15000 images for 4 hours on 128 GPUs.

		Predicted				
		γ	$e\pm$	$\mu+$	$\pi+$	$K+$
True	γ	66.1%	33.2%	0.0%	0.7%	0.0%
	$e\pm$	0.9%	98.8%	0.0%	0.3%	0.0%
	$\mu+$	0.0%	0.0%	99.4%	0.0%	0.5%
	$\pi+$	0.0%	1.1%	1.0%	68.0%	29.9%
	$K+$	0.4%	0.4%	1.8%	67.1%	30.4%

It should be noted that the 7 image GPU memory limitation did not apply for `SCNet`, due to the small size of each datum, yet we kept the minibatch size at 7. This study did not explore varying the minibatch size (and in extension batch size), but the researchers hypothesize that increased speedup could be realized using more optimal batch sizes during training of `SCNet`.

4. Discussion and Conclusions

Training both dense and sparse convolutional networks on 10s to 100s of GPUs proved successful in this study. The wall time to a training loss decreased linearly with increased amounts of GPU resources, and allowed training on large images for physics related classificatoin studies.

The final accuracy metrics for both of these networks are shown in Tables 1a and 1b. The final validation accuracy for `JishNet` was 78.8% ± 3.8%, and for `SCNet` was 76.8% ± 3.3%. Particle labeling confusion is observed among species for which physics intuition suggests it is not unexpected. The relative differences between the two networks in the K-π submatrix can be attributed to the well-known similarities between those two hadronic interactions and the fact that some of the kaons in our simulated samples have decayed to pions. In some cases the decay topologies are inseparable. We expect different networks and different stages of training of the same network to trade-off identification of charged kaons and pions against each other. While it would be interesting to improve the performance of the networks in this corner of phase space, such an endeavor exceeded the scope of this study.

This study also elucidated several other important aspects related to training CNNs with large image sizes. These aspects included the sensitivity of training to the optimizer parameters. We also uncovered the importance of minibatch size and tradeoff between GPU memory resources and optimal training dynamics.

Acknowledgements
The authors gratefully acknowledge the MicroBooNE collaboration for permission to work on simulated LArTPC data to focus on compute resources and performance scaling. This research used resources of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725. Thus, the Oak Ridge Leadership Computing Facility and its staff are also gratefully acknowledged for use of the *SummitDev* computer. The authors also gratefully acknowledge Pacific Northwest National Laboratory's Institutional Computing for use of its resources.

References

[1] The MicroBooNE Collaboration 2017 *Journal of Instrumentation* **12** P02017–P02017 URL `https://doi.org/10.1088%2F1748-0221%2F12%2F02%2Fp02017`
[2] Bhattacharya K, Church E, Schram M, Strube J, Wierman K, Daily J and Siegel C 2018 Scaling studies for deep learning in LArTPC event classification *International Conference on Computing in High Energy and Nuclear Physics* (Sofia, Bulgaria)
[3] Mikami H, Suganuma H, U-chupala P, Tanaka Y and Kageyama Y 2018 (*Preprint* `1811.05233`) URL `http://arxiv.org/abs/1811.05233`
[4] Sergeev A and Del Balso M 2018 (*Preprint* `1802.05799`) URL `http://arxiv.org/abs/1802.05799`
[5] Amatya V 2018 Machine Learning Toolkit for Extreme Scale (MaTEx)
[6] Paszke A, Gross S, Chintala S, Chanan G, Yang E, DeVito Z, Lin Z, Desmaison A, Antiga L and Lerer A 2017 Automatic differentiation in pytorch *NIPS-W*
[7] Alted F and Haenel V 2019 python-blosc: a Python wrapper for the extremely fast Blosc compression library URL `https://github.com/Blosc/python-blosc`
[8] Goyal P, Dollár P, Girshick R, Noordhuis P, Wesolowski L, Kyrola A, Tulloch A, Jia Y and He K 2017 ISSN 2167-3888 (*Preprint* `1706.02677`) URL `http://arxiv.org/abs/1706.02677`
[9] Kingma D P and Ba J 2014 (*Preprint* `1412.6980`) URL `http://arxiv.org/abs/1412.6980`
[10] Graham B and van der Maaten L 2017 *arXiv preprint arXiv:1706.01307*