Scaling the training of particle classification on Liquid Argon Time Projection Chamber events to multiple GPUs

An application of Convolutional Neural Networks on US Leadership Class computing facilities

Alex Hagen*, Eric Church, Jan Strube, Kolahal Bhattacharya, Vinay Amatya - *Pacific Northwest National Laboratory, Richland, WA, USA*

19th International Workshop on Advanced Computing and Analysis Techniques in Physics Research - Saas Fee, Switzerland - March 11th - 15th, 2019

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.

LArTPC Classification with Convolutional Neural Networks

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 of GPUs.

The MicroBooNE Experiment and Data

- The Detector
  - MicroBooNE is a 170 Tonne LArTPC [1]
  - Readout consists of
    - 2 induction planes with 2400 wires each
    - 1 collection plane with 3156 wires
    - 9600 digitizations over 4.8 ms (which is $2.2\times$ the TPC drift length)
- The Data
  - We use simulated events for single particle interactions. Permission from the MicroBooNE collaboration for this study is granted to focus on compute resource and performance studies.
  - One event image is 3600×3600, created by padding the 3156-wire collection plane wire data for the time from time tick 3200 to 6400 with zeros.

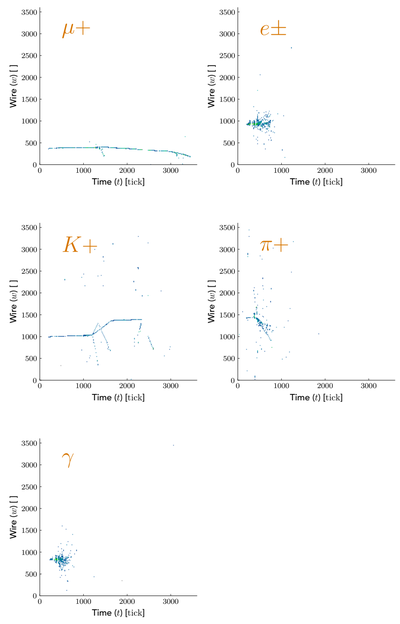

Figure 1: Example of simulated single particle interactions for classification

Event Classification with Convolutional Neural Networks

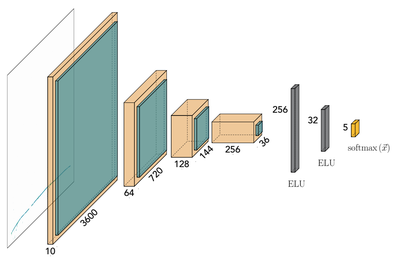

Figure 2: Design of "`JishNet`" and "`SCNet`" convolutional networks used for all testing presented on this poster

- A simple convolutional network was designed for testing of multi-GPU scaling, called `JishNet` (and a sparse convolutional equivalent named `SCNet`).
- The network included four dimensional convolutions and max pooling layers of size 5×5 (in the first two blocks) and 4×4 (in the second two blocks).
- Two fully connected layers of size 256 and 32, respectively, followed the convolutions.
- The network output used a dimension 5 softmax layer (for labels γ, $e\pm$, $\mu+$, $\pi+$, $K+$).

Previous Scaling Studies with multiple GPUs

- This group (Bhattacharya, Church, Strube, et al.) presented scaling of this task at CHEP 2018 [2].
- A PNNL developed scaling framework named `MaTEx` [3] was used to for scaling.
- Scaling was performed on PNNL's *Marianas* cluster. A training speed up was observed, as shown in Figure 3.
- The loss after 50 epochs was almost $6\times$ lower when using 14 GPUs (vs. 1 GPU). See Figure 4.

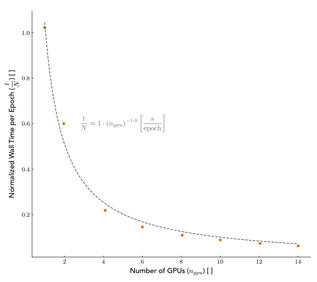

Figure 3: Normalized time to train an epoch on varying numbers of GPUs using `MaTEx`

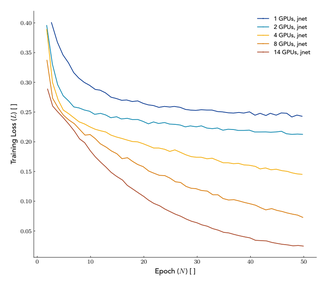

Figure 4: Training loss during training on varying numbers of GPUs using `MaTEx`

Towards High Performance Computing Scale

Many industrial applications of convolutional networks can apply 10s to 100s of GPUs to speed training and enable fast iteration in network design [4, 5]. This poster presents the first known instance of scaling of the above test training case to this scale on Oak Ridge National Laboratory Leadership Computing Facility's *SummitDev* computer.

PNNL Institutional Computing

- Previous work utilized PNNL Institutional Computing's *Marianas* cluster for scaling studies.
- *Marianas* is a cluster built specially for GPU-heavy work.
- *Marianas* has 7 nodes with:
  - 2 Intel Broadwell E5-2620 v4 @ 2.10GHz CPUs
  - 2 NVIDIA P100 12GB PCI-e based GPUs

The SummitDev Computer

- Oak Ridge National Laboratory's Leadership Computing Facility is commissioning a HPC GPU optimized machine named *Summit*. *SummitDev* is the test stand for *Summit* and has 50 nodes with:
  - 2 22-core IBM Power8NVL CPUs and 4 NVIDIA P100s
  - Mellanox EDR 100G InfiniBand node to node connections and NVLink node to node and GPU to host node communication
- There is a 4 hour time limit on all *SummitDev* jobs.

Implementation on *SummitDev*

Implementation of training on multiple GPUs is a non-trivial detail that many commercial entities are tackling. This poster presents one implementation using common open source and industry standard packages.

Frameworks

Deep Learning Framework - `PyTorch` [6]

In Memory Compression for Batching - `BLOSC` [7]

Data Parallelization Framework - `Horovod` [5]

Batching

- Initialization of batching used `BLOSC` to compress images into memory, images were uncompressed only at batch training time.
- With our network and image size of 3600×3600, only 7 images could fit into a single NVIDIA P100's memory, so this was our chosen minibatch size.
- Batching using sampling-**without**-replacement strategy was developed to ensure no data duplication across GPUs [8].
- Tensors provided to each GPU were tested to ensure uniqueness across all GPUs in all batches.

Training

- A typical supervised learning training loop was implemented within `PyTorch`.
- At the end of each batch, loss and accuracy were saved locally to each machine.
- `Horovod`'s implementation of the `allreduce` function was used to average loss and accuracy across GPUs for each batch, and more importantly to accumulate gradients across GPUs.

Optimizer Updates

- 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].
- Distributed Adam optimization was tested and ultimately successful; thereafter, all results used Adam optimization [9].
- The training dynamics are highly sensitive to parameters of Adam optimizer, final parameters were $lr = 0.001$, $\beta = [0.9, 0.999]$, $\epsilon = 1 \times 10^{-8}$, and $decay = 0.01$.

Scaling Results

- Scaling to multiple GPUs again shows a speed up with more GPUs (Figure 5). The speed up is approximately linear and is expected to be linear if more GPU sizes were tested.
- Again, training with more GPUs allows for a lower loss in the same amount of time (all training on *SummitDev* was limited to 240 minutes) as shown in Figure 6.
- Data loading time (DLT) - the time required to load all events for training into memory before training begins, is shown in the inset in Figure 6, and decreases with increasing GPUs.

Table 1: 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%

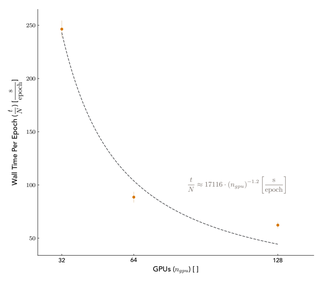

Figure 5: Data throughput versus number of GPUs using `JishNet` with `Horovod`

- Time to obtain a loss of 1.2 (TTL) is shown in the inset in Figure 6, and decreases with increasing GPUs.
- Final training accuracy after 4 hours increases with more GPUs, as shown in Figure 7.
- Validation accuracy is $78.8\% \pm 3.8\%$, with most confusion occuring in $K+$ and $\pi\pm$ discrimination, shown in Table 1.

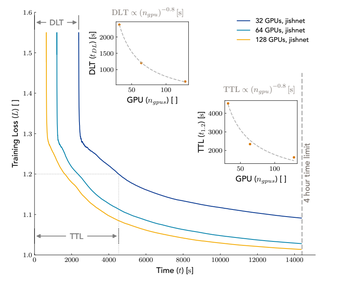

Figure 6: Loss over time with varying numbers of GPUs using `JishNet` with `Horovod` on 15000 images - Time to reach a loss value of 1.2 (TTL) and data loading time (DLT) are plotted in the insets

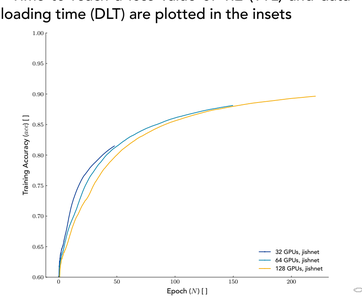

Figure 7: Training accuracy with varying numbers of GPUs using `JishNet` with `Horovod` on 15000 images

An Efficient Alternative: Sparse Convolutional Networks

Most of the data passed to the dense convolutional network are zeros, as no events happen in large parts of the TPC. Sparse convolutional networks are a more efficient network architecture for LArTPC event classificaton. Laura Domine and Kazuhiro Terao are presenting more extensively on sparse convolutional networks in Track 2: Data Analysis - Algorithms and Tools. We use Facebook's SparseConvNet [10] codebase for this work.

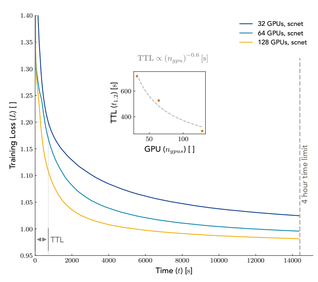

Figure 8: Training loss over time for varying numbers of GPUs using `SCNet` with `Horovod` using 15000 images - Time to train to a loss of 1.2 is shown in the inset axis

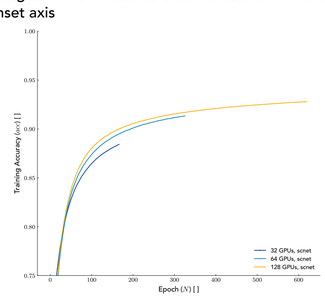

Figure 9: Training accuracy during training with varying numbers of GPUs using `SCNet` with `Horovod` on 15000 images

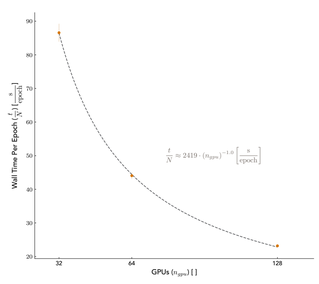

Figure 10: Data throughput versus number of GPUs *SCNet* with varying number of GPUs trained on 15000 images

- Because of the small size of a datum passed to `SCNet`, batch sizes of 7 events were no longer limiting.
- A further study on larger batch sizes for SCNs should be explored.

- SCNs greatly reduced the training time on 15000 LArTPC images ($\sim 5.2\times$ faster).
- SCNs also obtained comparable accuracy in the same training time, as shown in Figure 9.
- The final validation accuracy for `SCNet` was $76.8\% \pm 3.3\%$, which is close to that for `JishNet`.
- Training with SCNs greatly reduced loading time, as shown in the left part of Figure 8.
- Time to obtain a loss of 1.2 (TTL) is shown in the inset in Figure 8, and decreases with increasing GPUs. For reference, it is $\sim 4\times$ lower than that for dense CNNs.

Table 2: 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%

References

[1] R. Acciarri et al. "Design and construction of the MicroBooNE detector". In: *Journal of Instrumentation* 12.02 (2017), P02017–P02017. DOI: 10.1088/1748-0221/12/02/p02017. URL: https://doi.org/10.1088%2F1748-0221%2F12%2F02%2Fp02017.

[2] Kolahal Bhattacharya et al. "Scaling studies for deep learning in LArTPC event classification". In: *International Conference on Computing in High Energy and Nuclear Physics*. Sofia, Bulgaria, 2018.

[3] Vinay Amatya. *Machine Learning Toolkit for Extreme Scale (MaTEx)*. 2018.

[4] Hiroaki Mikami et al. "ImageNet/ResNet-50 Training in 224 Seconds". In: (2018). arXiv: 1811.05233. URL: http://arxiv.org/abs/1811.05233.

[5] Alexander Sergeev and Mike Del Balso. "Horovod: fast and easy distributed deep learning in TensorFlow". In: (2018). arXiv: 1802.05799. URL: http://arxiv.org/abs/1802.05799.

[6] Adam Paszke et al. "Automatic differentiation in PyTorch". In: *NIPS-W*. 2017.

[7] Francesc Alted and Valentin Haenel. *python-blosc: a Python wrapper for the extremely fast Blosc compression library*. 2019. URL: https://github.com/Blosc/python-blosc.

[8] Priya Goyal et al. "Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour". In: (2017). ISSN: 2167-3888. DOI: 10.1561/2400000003. arXiv: 1706.02677. URL: http://arxiv.org/abs/1706.02677.

[9] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: (2014). arXiv: 1412.6980. URL: http://arxiv.org/abs/1412.6980.

[10] Benjamin Graham and Laurens van der Maaten. "Submanifold Sparse Convolutional Networks". In: *arXiv preprint arXiv:1706.01307* (2017).

OLCF Oak Ridge Leadership Computing Facility

Pacific Northwest National Laboratory