Accelerated Machine Learning as a Service for Particle Physics Computing

Javier Duarte
University of California San Diego
Fermi National Accelerator Laboratory

Burt Holzman, Sergo Jindariani, Thomas Klijnsma, Benjamin Kreis, Mia Liu, Kevin Pedro, Nhan Tran, Aristeidis Tsaris
Fermi National Accelerator Laboratory

Phil Harris, Dylan Rankin
MIT

Vladimir Loncar, Jennifer Ngadiuba, Maurizio Pierini
CERN

Suffian Khan, Brian Lee, Brandon Perez, Ted W. Way, Colin Versteeg
Microsoft

Scott Hauck, Shih-Chieh Hsu, Matthew Trahms, Dustin Werran
University of Washington

Zhenbin Wu
University of Illinois Chicago

Abstract

Large-scale particle physics experiments face challenging demands for high-throughput computing resources both now and in the future. New heterogeneous computing paradigms on dedicated hardware with increased parallelization, such as Field Programmable Gate Arrays (FPGAs), offer exciting solutions with large potential gains. The growing applications of machine learning algorithms in particle physics for simulation, reconstruction, and analysis are naturally deployed on such platforms. We demonstrate that the acceleration of machine learning inference as a web service represents a heterogeneous computing solution for particle physics experiments that requires minimal modification to the current computing model. As an example, we retrain the `ResNet-50` convolutional neural network to demonstrate state-of-the-art performance for top quark jet tagging at the LHC. Using Microsoft Azure Machine Learning deploying Intel FPGAs to accelerate the `ResNet-50` image classification model, we achieve average inference times of 60 (10) milliseconds with our experimental physics software framework deployed as a cloud (edge or on-premises) service, representing an improvement by a factor of approximately 30 (175) in model inference latency over traditional CPU inference in current experimental hardware. A single FPGA service accessed by many CPUs achieves a throughput of 600-700 inferences per second using an image batch of

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

one, comparable to large batch-size GPU throughput and significantly better than small batch-size GPU throughput. Deployed as an edge or cloud service for the particle physics computing model, coprocessor accelerators can have a higher duty cycle and are potentially much more cost-effective.

1 Introduction

With large datasets and high data acquisition rates, high-performance and high-throughput computing resources are an essential element of the experimental particle physics. These experiments are constantly increasing in both sophistication of detector technology and intensity of particle beams. As such, particle physics datasets are growing in size just as the algorithms that process the data are growing in complexity. For example, the high luminosity phase of the Large Hadron Collider (HL-LHC) will deliver 15 times more data than the current LHC run and the data itself will be at least an order of magnitude more complex due to higher number of readout channels and particles per collision. Such a data rate leads to datasets that are exabytes in scale [1]. Future neutrino experiments such as Deep Underground Neutrino Experiment (DUNE) [2] and cosmology experiments like Square Kilometre Array (SKA) [3] are expected to produce datasets at the exabyte scale. Concurrently, improvement in single processor performance is stalling due to changes in the scaling of power consumption [4]. The current particle physics computing paradigms will not suffice to simulate, process, and analyze the massive datasets that the next-generation experimental facilities will deliver. New technologies that provide order-of-magnitude improvements are needed.

The ubiquity of sophisticated detectors with complex outputs has led to the quick adoption of machine learning (ML) algorithms as tools to reconstruct physics processes. Neutrino experiments currently use state-of-the-art convolutional neural networks (CNNs) [5–7] for neutrino event reconstruction and identification. At the LHC, ML methods are used in all stages of the experiments, from sub-microsecond online filtering applications [8, 9] to low-level calibration of individual reconstructed particles [10] to high-level optimization of final-state event topologies [11]. Across big science, such as cosmology and large astrophysical surveys, similar trends exist as the experiments grow and the data rates increase. While the computing challenge in particle physics is a vital concern for current and future experiments, it is not unique. With the rise of so-called "big data" across a wide range of scientific fields, the sophisticated large-scale processing of big data has become a global challenge. At the forefront of this trend is the need for new computing resources to handle both the training and inference of large ML models.

In this paper, we focus on the **inference** of deep ML models as a solution for processing large datasets [12]; inference is computationally intensive and runs repeatedly on hundreds of billions of events. A growing trend to improve computing power has been the development of hardware that is dedicated to accelerating certain kinds of computations. Pairing a specialized coprocessor with a traditional CPU, referred to as *heterogeneous computing*, greatly improves performance. These specialized coprocessors, including GPUs, Field Programmable Gate Arrays (FPGAs), and Application Specific Integrated Circuits (ASICs), utilize natural parallelization and provide higher data throughput. ML algorithms, and in particular deep neural networks, are at the forefront of this computing revolution due to their high parallelizability and common computational needs.

To capitalize on this new wave of heterogeneous computing and specialized hardware, particle physicists have two primary options:

1. *Adapt domain-specific algorithms to run on specialized accelerator hardware.*
   This option takes advantage of specific human expert knowledge, but can be challenging to implement on new and ever-changing hardware platforms with different computing paradigms. New portable development environments (e.g. `OpenCL` [13]) can potentially provide cross-hardware solutions.
2. *Design ML algorithms to replace domain-specific algorithms.*
   This option has the advantage of running natively on specialized hardware using open-source software stacks, but it can be a challenge to map specific physics problems onto ML solutions.

In this paper, we explore how such heterogeneous computing resources can be deployed within the current computing model for particle physics in a scalable and non-disruptive way. We will present

physics results for a publicly available top quark tagging dataset for the LHC [14]. This study focuses on the newly available Microsoft Azure ML platform that deploys FPGA coprocessors as a service at datacenter scale [15]. Azure provides a first scalable platform to study, though other such options exist. Results from this study will serve as a performance benchmark for any similar systems and will provide valuable lessons for applying new technologies to particle physics computing.

2 Benchmark Model Performance

The computing model for many large scale physics experiments is based on processing events. An event here is defined as a measurement of some physical process of interest; in the case of the LHC, it is a collision of bunches of protons every 25 ns. The event consists of complex detector signals that are filtered, combined, and analyzed; typically, the raw signal inputs are converted into objects with a more physical meaning. There is both online processing, in which the event is selected from a buffer and analyzed in real time, and offline processing, in which the event has been written to disk and is more thoroughly analyzed with less stringent latency requirements. It is important to note that the basic processing unit is a single event and performing the same task for multiple events (batching) becomes significantly more complex to manage. Because each event contains potentially millions of channels of information, it is optimal to load the needed components of that event into memory and then execute all desired algorithms for that event. The tasks themselves can be very complex, either with time-consuming physics-based algorithms, or, as is becoming more popular, machine learning algorithms. There may be dozens or even hundreds of modules executed for each event.

For this paper, we demonstrate performance for one such task of identifying top quarks at the LHC through the production of collimated sprays of particles in the detector called *jets*. Jets are common products for many physics processes, and only jets consistent with a certain structure correspond to a top quark. Because this task involves highly-correlated and high-dimensionality inputs of lists of particles, it is an active area of R&D for ML algorithms in particle physics. For this example, we choose the jet to be represented as a 2D image and use `ResNet-50` [16] for this specific task. We use `ResNet-50` as the primary featurizer, then we add our own custom classifier, which comprises one fully connected layer of width 1024 with ReLU [17] activation and another fully connected layer of width 2 with softmax activation. The training is performed by minimizing the categorical cross-entropy loss function using the Adam algorithm [18] with an initial learning rate of 10^{-3} and a minibatch size of 64 over 10 epochs on an NVIDIA Tesla V100 GPU. The training for this particular `ResNet-50` model is unique, and performed in steps, because there is a particular *quantized* version of `ResNet-50` with reduced precision for the FPGA and needs to be "fine-tuned," or trained with a smaller learning rate. Therefore, the quantized model is initialized using the weights from the trained floating point model. Finally, as the quantized model evaluated with the Azure FPGA service differs numerically from the quantized model evaluated on the local GPU, an additional fine-tuning is applied to the classifier after evaluating the final `ResNet-50` features.

After training, we evaluate the performance of our trained `ResNet-50` top tagger. The receiver operator characteristic (ROC) curve is a graph of the false positive rate (the background QCD jet efficiency ε_B) as a function of the true positive rate (top quark jet efficiency ε_S). It is customary to report three metrics for the performance of the network on the top tagging dataset: model accuracy, area under the ROC curve (AUC), and background rejection power at a fixed signal efficiency of 30%, $1/\varepsilon_B(\varepsilon_S = 30\%)$. Fig. 1 shows the ROC curve comparison for `ResNet-50` as well as the fully retrained featurizer with custom weights. With a background rejection power of $1/\varepsilon_B(\varepsilon_S = 30\%) \approx 1000$, the retrained `ResNet-50` outperforms the other models developed for this dataset.

3 Implementation, Latency, and Throughput

One challenge is to integrate FPGA coprocessors into the computing model without disrupting the current multithreaded paradigm, where several modules process an event in parallel. A natural method for integrating heterogeneous resources is via a *network service*. This client-server model is flexible enough to be used locally by a single user or within a computing farm where a single thread communicates with the server. In the particular case investigated here, we use the `gRPC` package [19], an open-source Remote Procedure Call (RPC) system, interfaced with the Azure system. This setup defines a communication method between the FPGA coprocessor resources and an experiment's

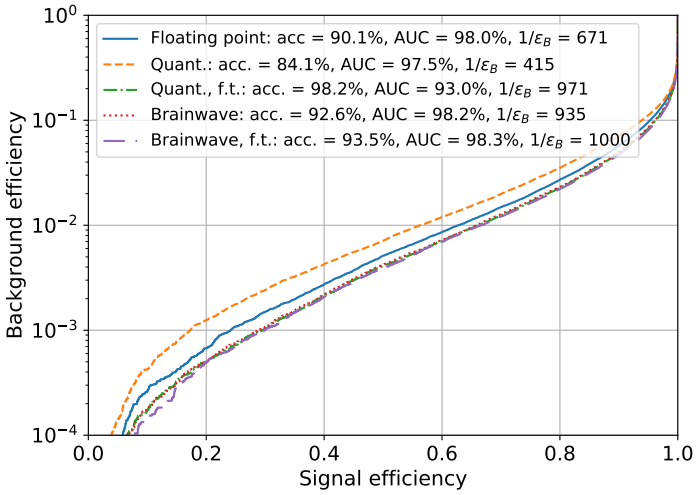

Figure 1: ROC curves showing performance of the floating point and quantized versions (before fine-tuning, after fine-tuning, and using the Brainwave service) of the `ResNet-50` top tagging model.

primary computing CPU-based datacenters. We test the performance of a single task which makes a request to a single service which performs an access to the Azure platform. We also scale up the number of simltaneous requests to the Azure system, which is capable of load balancing of service requests.

For our demonstration study, we use the CMS experiment software framework, `CMSSW` [20]. A typical module has a *produce* function that obtains data from an event, operates on it, and then outputs derived data. Our goal is to utilize the Brainwave hardware as a service to perform inference of a large ML model such as `ResNet-50`. Within `CMSSW`, a hook to the `gRPC` system is established using a special feature called `ExternalWork` [21].

To summarize the results, for total inference time for a batch of one image, we present Azure, CPU, and GPU performance in Table 1. The most straightforward comparison with the current CMSSW performance of 1.75 seconds is the 10 (60) ms *on-prem* (*remote*) that it would take to perform inference with Azure. This represents a factor of 175 (30) speedup for Azure *on-prem* (*remote*) over current `CMSSW` CPU performance. We can extrapolate from Table 1 that, for more modern versions of `TensorFlow` and CPUs, the `CMSSW` CPU inference time could improve to approximately 500 ms with a single core. GPU comparisons can be more nuanced depending on the model implementation and batch sizes. However, for a batch of one image, we can say that the Azure inference latencies, both *on-prem* and *remote* including network latencies, are of a similar order to local, physically connected GPU inference times. The GPU and Azure have similar maximum throughput, about 660 images per second, though the former only achieves this with large batch size and the latter achieves this when accessed with many CPUs simultaneously.

Table 1: A summary comparison of total inference time for Brainwave, CPU, and GPU performance using the same `ResNet-50` model.

Type	**Hardware**	⟨**Infer. time**⟩	**Max throughput**	**Setup**
CPU	Xeon 2.6 GHz, 1 core	1.75 seconds	0.6 img/s	`CMSSW`, `TF v1.06`
CPU	i7 3.6 GHz, 1 core	500 ms	2 img/s	`python`, `TF v1.10`
CPU	i7 3.6 GHz, 8 core	200 ms	5 img/s	`python`, `TF v1.10`
GPU (b=1)	NVidia GTX 1080	100 ms	10 img/s	`python`, `TF v1.10`
GPU (b=32)	NVidia GTX 1080	9 ms	111 img/s	`python`, `TF v1.10`
GPU (b=1)	NVidia GTX 1080	7 ms	143 img/s	`TF v1.10` (internal)
GPU (b=32)	NVidia GTX 1080	1.5 ms	667 img/s	`TF v1.10` (internal)
Azure (FPGA)	Altera Artix	10 ms	660 img/s	`CMSSW`, *on-prem*
Azure (FPGA)	Altera Artix	60 ms	660 img/s	`CMSSW`, *remote*

4 Outlook

In this first study, we have explored the potential to accelerate computing for growing big science challenges using coprocessor hardware. Through the acceleration of machine learning algorithms, we can take advantage of developments beyond our domain in both new hardware paradigms such as GPUs, FPGAs, and ASICs, and open-source software stacks. Using a representative experimental software framework, `CMSSW`, and the accelerator hardware *as a web service*, we have found a large speed-up in inference latency for `ResNet-50` over what can be achieved with CPUs within `CMSSW`. The throughput of the FPGA as a service is also comparable to large-batch GPU throughput when accessing the FPGA over many CPUs simultaneously. This demonstrates a potentially cost-effective and non-disruptive integration of heterogeneous computing resources into the experimental particle physics computing paradigm. While this work is very promising as a solution for future computing challenges, we must continue to develop the algorithms, infrastructure, and hardware technologies in order to scale-out this solution for big science.

References

[1] HEP Software Foundation. `https://arxiv.org/abs/1712.06982`, 2017.

[2] DUNE collaboration, R. Acciarri et al. `https://arxiv.org/abs/1601.05471`, 2016.

[3] G. Mellema et al., *Reionization and the Cosmic Dawn with the Square Kilometre Array*, *Exper. Astron.* **36** (2013) 235 [`1210.0197`].

[4] National Research Council. 2011, 10.17226/12980.

[5] MicroBooNE collaboration, *Convolutional Neural Networks Applied to Neutrino Events in a Liquid Argon Time Projection Chamber*, *JINST* **12** (2017) P03011 [`1611.05531`].

[6] A. Aurisano, A. Radovic, D. Rocco, A. Himmel, M. D. Messier, E. Niner et al., *A Convolutional Neural Network Neutrino Event Classifier*, *JINST* **11** (2016) P09001 [`1604.01444`].

[7] K. He, X. Zhang, S. Ren and J. Sun, *Deep residual learning for image recognition*, *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016) [`1512.03385`].

[8] J. Duarte et al., *Fast inference of deep neural networks in FPGAs for particle physics*, *JINST* **13** (2018) P07027 [`1804.06913`].

[9] CMS collaboration Tech. Rep. CMS-CR-2017-361, CERN, Geneva, Oct, 2017.

[10] CMS collaboration, *Energy Calibration and Resolution of the CMS Electromagnetic Calorimeter in pp Collisions at $\sqrt{s} = 7$ TeV*, *JINST* **8** (2013) P09009 [`1306.2016`].

[11] T. Q. Nguyen, D. Weitekamp, D. Anderson, R. Castello, O. Cerri, M. Pierini et al. `https://arxiv.org/abs/1807.00083`, 2018.

[12] J. Duarte et al., *FPGA-accelerated machine learning inference as a service for particle physics computing*, *Comput. Softw. Big Sci.* **3** (2019) 13 [`1904.08986`].

[13] J. E. Stone, D. Gohara and G. Shi, *Opencl: A parallel programming standard for heterogeneous computing systems*, *Computing in science & engineering* **12** (2010) 66.

[14] Gregor Kasieczka, Michael Russell, Tilman Plehn, "Top Tagging Reference Dataset." `https://goo.gl/XGYju3`, 2017.

[15] A. Caulfield, E. Chung, A. Putnam, H. Angepat, J. Fowers, M. Haselman et al., *A cloud-scale acceleration architecture*, IEEE Computer Society, October, 2016, https://www.microsoft.com/en-us/research/publication/configurable-cloud-acceleration/.

[16] K. He, X. Zhang, S. Ren and J. Sun, *Deep residual learning for image recognition*, in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016, https://arxiv.org/abs/1512.03385.

[17] V. Nair and G. E. Hinton, *Rectified linear units improve restricted Boltzmann machines*, in *Proceedings of ICML*, vol. 27, pp. 807–814, 06, 2010.

[18] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization." https://dblp.org/rec/bib/journals/corr/KingmaB14, 2014.

[19] Google, "gRPC." version v1.14.0 https://grpc.io/, 2018.

[20] CMS Collaboration, "CMSSW." version CMSSW_10_2_0 https://github.com/cms-sw/cmssw, 2018.

[21] K. Pedro, "SonicCMS." version v3.1.0 https://github.com/hls-fpga-machine-learning/SonicCMS, 2019.