Accelerating High-energy Physics Exploration with Deep Learning

Dave Ojika, Darin Acosta, Ann Gordon-Ross, Andrew Carnes, * Sergei Gleyzer
University of Florida, Gainesville FL, USA. * CERN, Geneva, Switzerland
{davido, acostad,anngordonross}@ufl.edu, acarnes@phys.ufl.edu, sergei.gleyzer@cern.ch

I. Introduction

Every year, up to 30 petabytes of data are captured from the large hadron collider (LHC) at CERN, the European Centre for Nuclear Research. 1 petabyte of this data is offline-processed everyday using 11,000 servers with 100,000 processor cores. This huge amount of data represents only a very small fraction of the total amount of raw data generated by sensors in the collider's trigger system at a rate of 40 million events per second. In order to adhere to the very strict experimental requirements, novel and efficient algorithms are needed to perform the required physics analytics very quickly during runtime. For the Compact Muon Solenoid (CMS) experiment [1], the design, construction, and processing of a large portion of the Level-1 trigger data for muon detection will become even more challenging with nearly 1 billion collisions occurring per second as a result of the increased luminosity from proposed LHC upgrades.

Recently, machine learning has gained tremendous popularity in the research community and is used in many data-processing-intensive applications, ranging from recommendation engines [10, 11] and large-scale recognition applications to sequencing and identification systems [9]. Deep learning, which is a branch of machine learning, is based on a set of algorithms that model high-level data abstractions. While traditional machine learning algorithms (e.g., SVM, KNN, and BDT) [17] require a prior pre-processing step that manually extracts salient features from the data, deep learning obviates the need for this pre-processing step by dynamically performing feature extraction on the raw input data during runtime, which makes deep learning attractive for identifying physics particles based on raw detector readings from particle colliders [18].

An important aspect of data processing for the trigger system of LHC detectors is classifying collision events as signal ("an interesting event") or noise ("an uninteresting event"). Interesting events are saved for later processing, and uninteresting events are discarded. This classification must be performed very quickly to keep up with the continuous stream of real-time data samples. In this work, we present our approach to using deep learning for identification of rarely produced physics particles (such as the Higgs Boson) out of a majority of uninteresting, background or noise-dominated data. A fast and efficient system to eliminate uninteresting data would result in much less data being stored, thus significantly reducing processing time and storage requirements. In this paper, we present a generalized preliminary version of our approach to motivate research interest in advancing the state-of-the-art in deep learning networks for other applications that can benefit from learning systems.

PEARC17, July 09-13, 2017, New Orleans, LA, USA

ACM ISBN 978-1-4503-5272-7/17/07.
http://dx.doi.org/10.1145/3093338.3093340

II. Background

A. Deep Learning

Deep Learning is a branch of artificial neural networks (ANN) where a computational model is composed based on the structure and behavior of biological neural networks. This computational model is composed of multiple layers that are interconnected to form a network. Each layer contains multiple processing nodes (often referred to as neurons). Information is propagated through the network via a set of non-linear transformations called activation functions that describe the output of a given node based on an input or set of inputs. By observing intricate structures in large datasets, deep learning can "learn" representations of data with multiple levels of abstraction.

Fig. 1 depicts the two phases of deep learning: the *training* and *prediction* phases. The training phase uses a method called back propagation to indicate how the internal parameters (weights) of the deep neural network should change to more accurately reflect the representations in each layer from the representation in the previous layer. Since there is typically millions of training examples in which to train the deep neural network, and thus millions of adjustable weights, the learning phase is a lengthy process.

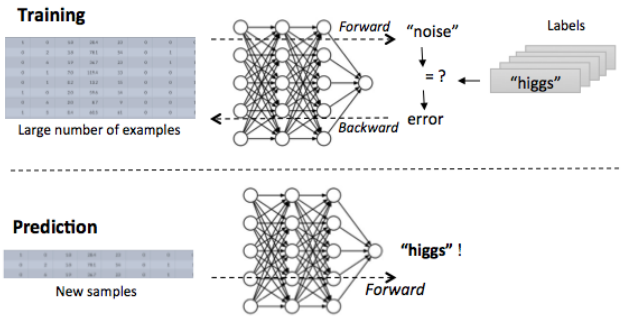

Fig 1: Phases of deep learning: The training phase (top) uses a large set of examples to learn data representations. The trained network is then used to determine information about new data in the prediction phase (bottom).

Once a neural network is trained, the network can be used in the prediction phase (sometimes referred to as inference or scoring), where the network attempts to map arbitrary input instances of data to the learned weights, which can generate predictions about that instance. For example, in convolutional neural networks (a type of neural network used for image recognition), when a dataset of cats are used in the training phase, the neural network could then be used to predict whether a given image does or does not contain a cat.

Deep neural networks (DNN) differ from traditional machine learning algorithms in that deep neural networks have several (hidden) layers. The *depth* of the network, often indicated by the number of layers, generally improves the prediction accuracy [12, 13, 14] at the cost of increased computation time. However, in contrast to CPU-only solutions, graphics processing units (GPUs) and other parallel architectures, such as the Intel Xeon Phi [16] and field-programmable logic (FPGAs) [15], can be used to speedup these computations.

III. Approach

In this section, we describe our computational approaches, including the techniques, algorithms, software, models, and data that we will evaluate (Section V) using our experimental infrastructure (Section IV).

A. Data Collection

We initially validate our design using a common dataset for benchmarking Higgs classification solutions [2], which appeared in a challenge at the Kaggel competition [19]. The dataset, produced using Monte Carlo simulations, contains 21 low-level features for each event recorded from the particle physics experiment, as well as 7 high-level features along with a classification label that indicates if the event is signal or noise. There are a total of 11 million events and all values are real floating-point numbers, except the class label, which has a binary value of 0 or 1 indicating if that sample is signal or noise, respectively. We arbitrarily partition the dataset to a 9:1 ratio for representing the training and test datasets, respectively.

B. Software Framework

We create prototype software models using Keras APIs (application programming interfaces), which are used for creating neural networks. As shown in Fig. 2, we use this API to build two processing engine data paths (*Data Path 1* and *Data Path 2*) to interface between the user and platform levels using TensorFlow [5] and Theano [6]. TensorFlow and Theano represent a model's computational structure as a computation graph of data. The user level abstracts these processing engines and the underlying hardware at the platform level to present a consistent holistic system view for model design exploration, enabling users to easily construct and evaluate different neural network models based on different design scenarios (e.g., accuracy versus computational costs) (Section III-C). We note that no source code change is required for any model at the user level, regardless of the data path and processing engine.

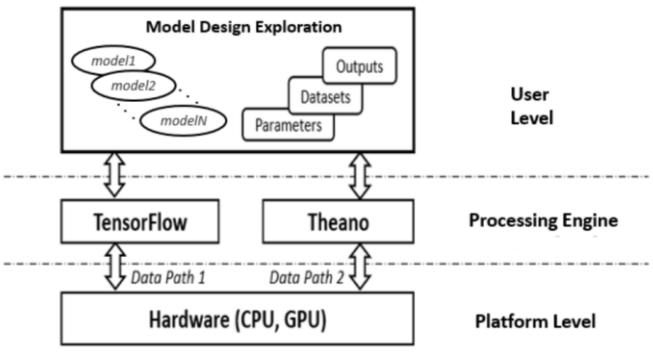

Fig 2: Design exploration using different DNN configurations. Computations can be executed by default on the CPU, or accelerated on the GPU for any given processing engine through any of the data paths.

C. Models

In order to evaluate different design scenarios, users can create different system models during model design exploration, where each model can be uniquely identified with a number ranging from 1 to *N* (maximum number of models explored). Each model can represent a different neural network configuration or optimization technique, where aspects, such as the number of layers and layer composition, can be varied. In this work, we evaluate two neural network models, *model1* and *model2*, using different numbers of sequential (dense) layers. *Model1* has two layers and *model2* has one additional layer to represent a deeper network than *model2*. This additional layer uses a higher number of neurons based on the implementation in [7]. Both models are configured with Stochastic Gradient Descent (SGD) as the optimizer, with a *cross-entropy* loss function. Even though we present a simple two-model evaluation, the fundamental design exploration practices are easily extendable to any number of models to determine an effective balance between model accuracy and computational costs.

IV. Experimental Infrastructure

We evaluate *model1* and *model2* using the two configurations represented by *Data Path 1* and *Data Path 2*, respectively, in Fig 2. In this section, we describe the infrastructure used for training and evaluating these models.

A. XSEDE Supercomputer

We ran our experiments on the SDSC (San Diego Supercomputing Center) Comet cluster [4], which is a dedicated cluster on XSEDE (Extreme Science and Engineering Development Environment) and is capable of delivering up to 2 petaflops. XSEDE is a single virtual system consisting of clusters of supercomputers, software tools, and shared data [8]. We used SDSC's Oasis distributed storage system for storing the large amount of experimental data.

B. Computation Resources

Due to resource limitations, a single dedicated node on Comet is used to train each model, however, multiple models could be used trained simultaneously across multiple nodes. This limitation increases the total computation time reported in Section V, but does not affect the analysis. The node contains two Intel E5-2680v3 processors, 4 NVIDIA Tesla K80 GPUs,

and 128 GB of DDR4 memory. Since TensorFlow did not support all of Keras's features at the time of this work, *Data Path 1* is limited to GPU resources only, whereas *Data Path 2* accesses both the GPU and CPU resources.

V. Results

The increase of a single layer from *model1* to *model2* expresses the relative performance of both models, which can be evaluated using two metrics: (1) the computation time during the training phase quantified in hours, minutes, or seconds; and (2) the achieved accuracy during the prediction phase, quantified as an AUC (area under curve) score. A high accuracy metric for a model is our primary objective of the evaluation, but it can become computationally expensive to achieve a high accuracy with respect to model complexity. While the prediction accuracy of respective models remain the same irrespective of data path (*Data Path 1 or Data Path 2),* we explore possibilities of speeding up the training process via each data path.

Fig. 3 depicts our experimental results for the training time (left) and prediction accuracy (right). As shown, *Data Path 2* (Theano with the CPU / GPU) is about 5X faster than *Data Path 1* (TensorFlow with the GPU) as a result of the increased number of CPU cores. The low performance of *Data Path 1* may have also been contributed by the ongoing performance improvements to TensorFlow as indicated in [5] where the authors suggest the use of a just-in-time compiler to achieve a number of optimizations such as loop fusion, blocking and tiling for locality, specialization for particular tensor shapes and sizes, etc.

For the prediction accuracy, the results show a 13% improvement in AUC for *model1* as compared to *model2,* which suggests that a single-layer increment in the neural network depth can greatly impact prediction accuracy. We are currently performing additional benchmarking to evaluate the performance accuracy trends to evaluate the benefits of additional layers (up to *N*), and whether the improvement in accuracy eventually plateaus. We are also evaluating the potential consequence of implementing more advanced optimizations, such as using *AdaGrad* as the network optimizer instead of SGD.

VI. Conclusions and Discussions

In this work, we described our evaluation of how using a development environment, such as XSEDE, facilitates efficient model design exploration for deep learning. We used Comet to explore different design options (accuracy vs training-time tradeoff), exemplifying how users can use our methodology to efficiently explore different model configurations and identify configurations that meet the user's required design goals, such as performance and accuracy. Our methodology allows users to identify the computational composition of models that meet design goals.

This work presents a preliminary and generalized methodology and evaluation, and leaves much future work. For example, due to limited access to processing hardware, we were unable to train deeper networks with more layers. Even though

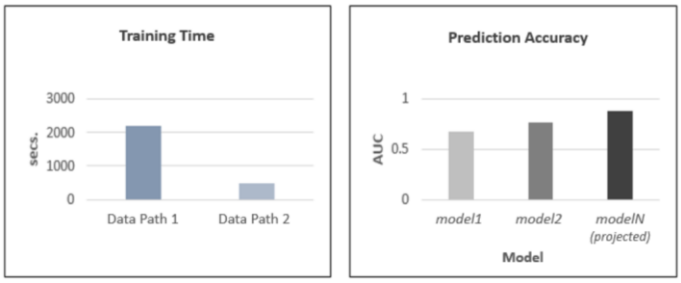

Fig 3: Training time (left) and prediction accuracy (right) on a node with 4 Tesla GPUs and 24 Xeon CPU cores. *Data Path 1* and *Data Path 2* are configured for GPU and CPU runs respectively. The prediction accuracy of a given model is the same irrespective of data path.

the main contribution of this work is flexible design assistance for exploring deep neural network models that achieve high accuracy at minimal processing cost, we also plan to explore how parallelizing the training phase across a large cluster of compute nodes on Comet deduces the train phase runtime.

While applications like the ATLAS [21] and CMS experiments require real-time performance for many classification tasks, designing an efficient hardware architecture requires software simulations and testing before any actual hardware is built. Our future work aims to enable hardware implementers and system designers with a platform for realizing hardware versions of DNN models that meet user-specified requirements. We also plan to conduct further studies to quantify the computational composition of newly developed models, which will further help in mapping models more efficiently to hardware platforms such as FPGAs and application-specific integrated circuits (ASICs).

Acknowledgment

This work was supported by the U.S Department of Energy, Office of Science, Basic Energy Sciences. The work also used the Extreme Science and Engineering Discovery Environment (XSEDE) and we thank Mahidhar Tatineni for his assistance with setting up TensorFlow on SDSC's Comet. We also acknowledge the use of University of Florida's HiperGator supercomputer for parts of this work, and thank the Compact Muon Solenoid (CMS) group at University of Florida and at CERN for their support.

References

[1] D. Acosta, N. Adams, A. Atamanchouk, R.D. Cousins, M.I. Ferguson, V. Golovtsov, J. Hausera, A. Madorsky, M. Matveev, J. Mumford, T. Nussbaum, P. Padley, B. Razmyslovich, V. Sedova, W. Smith, B. Tannenbaum ., "Development and Test of a Prototype Regional Track-Finder for the Level-1 Trigger of the Cathode Strip Chamber Muon System of CMS," Nuclear Instruments and Methods A496 (2003) 64-82

[2] http://archive.ics.uci.edu/ml/datasets/HIGGS

[3] Chollet, F keras, (2015), https://github.com/fchollet/keras

[4] SDSC Comet, https://portal.xsede.org/sdsc-comet

[5] Mart´ın Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Man´e, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Vi´egas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng , "TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems", 2015

[6] Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, James Bergstra, Ian Goodfellow, Arnaud Bergeron, Nicolas Bouchard, David Warde-Farley, Yoshua Bengio, "Theano: new features and speed improvements". NIPS 2012 deep learning workshop

[7] Abhinav Vishnu, Charles Siegel, Jeffrey Daily, "Distributed TensorFlow with MPI", Cornell University Library

[8] John Towns, Timothy Cockerill, Maytal Dahan, Ian Foster, Kelly Gaither, Andrew Grimshaw, Victor Hazlewood, Scott Lathrop, Dave Lifka, Gregory D. Peterson, Ralph Roskies, J. Ray Scott, Nancy Wilkins-Diehr "XSEDE: Accelerating Scientific Discovery", Computing in Science & Engineering

[9] Po-Hsien Liu; Shun-Feng Su; Ming-Chang Chen; Chih-Ching Hsiao, "Deep learning and its application to general image classification," Informative and Cybernetics for Computational Social Systems (ICCSS), 2015

[10] Jonathan L. Herlocker, Joseph A. Konstan, Loren G. Terveen, and John T. Riedl. 2004. Evaluating collaborative filtering recommender systems. *ACM Trans. Inf. Syst.* 22, 1 (January 2004), 5-53

[11] Chumki Basu, Haym Hirsh, and William Cohen. 1998. Recommendation as classification: using social and content-based information in recommendation. In *Proceedings of the fifteenth national/tenth conference on Artificial intelligence/Innovative applications of artificial intelligence* (AAAI '98/IAAI '98). American Association for Artificial Intelligence, Menlo Park, CA, USA, 714-720.

[12] Deep Residual Learning for Image Recognition. Kaiming He. Xiangyu Zhang. Shaoqing Ren. Jian Sun. Microsoft Research

[13] Going Deeper with Convolutions, Christian Szegedy

[14] Imagenet classification with deep convolutional neural networks Alex Krizhevsky

[15] C. Wang, L. Gong, Q. Yu, X. Li, Y. Xie and X. Zhou, "DLAU: A Scalable Deep Learning Accelerator Unit on FPGA," in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 36, no. 3, pp. 513-517, March 2017.

[16] A. Viebke and S. Pllana, "The Potential of the Intel (R) Xeon Phi for Supervised Deep Learning," *2015 IEEE 17th International Conference on High Performance Computing and Communications, 2015 IEEE 7th International Symposium on Cyberspace Safety and Security, and 2015 IEEE 12th International Conference on Embedded Software and Systems*, New York, NY, 2015, pp. 758-765.

[17] C. Crisci, B. Ghattas, G. Perera, A review of supervised machine learning algorithms and their applications to ecological data, Ecological Modelling, Volume 240, 10 August 2012, Pages 113-122

[18] Searching for exotic particles in high-energy physics with deep learning, P Baldi

[19] Kaggle Higgs Challenge. https://www.kaggle.com/c/higgs-boson

[20] Keras. http://kera.io

[21] G. Cataldi, "Muon identification with the event filter of the ATLAS experiment at CERN LHC's," 14th IEEE-NPSS Real Time Conference, 2005., Stockholm, 2005