

Physics Community Needs, Tools, and Resources for Machine Learning

Philip Harris, Erik Katsavounidis, William Patrick McCormack, Dylan Rankin
MASSACHUSETTS INSTITUTE OF TECHNOLOGY, CAMBRIDGE, MA 02139

Yongbin Feng, Abhijith Gandrakota, Christian Herwig, Burt Holzman, Kevin Pedro, Nhan Tran, Tingjun Yang, Jennifer Ngadiuba
FERMI NATIONAL ACCELERATOR LABORATORY, BATAVIA, IL 60510

Michael Coughlin

UNIVERSITY OF MINNESOTA, MINNEAPOLIS, MN 55455

Scott Hauck, Shih-Chieh Hsu, Elham E Khoda

UNIVERSITY OF WASHINGTON, SEATTLE, WA 98195

Deming Chen, Mark Neubauer

UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN, URBANA, IL 61801

Javier Duarte

UNIVERSITY OF CALIFORNIA SAN DIEGO, LA JOLLA, CA 92093

Georgia Karagiorgi

COLUMBIA UNIVERSITY, NEW YORK, NY 10027

Mia Liu

PURDUE UNIVERSITY, WEST LAFAYETTE, IN 47907

ABSTRACT

Machine learning (ML) is becoming an increasingly important component of cutting-edge physics research, but its computational requirements present significant challenges. In this white paper, we discuss the needs of the physics community regarding ML across latency and throughput regimes, the tools and resources that offer the possibility of addressing these needs, and how these can be best utilized and accessed in the coming years.

Submitted to the Proceedings of the US Community Study
on the Future of Particle Physics (Snowmass 2021)

Contents

1	Introduction	4
2	Needs	4
2.1	Collider Physics	5
2.2	Neutrino Physics	7
2.3	Astrophysics	7
3	Tools	8
3.1	Hardware	8
3.1.1	GPUs	9
3.1.2	FPGAs	10
3.1.3	ASICs	11
3.1.4	IPUs	12
3.1.5	Other specialized processors	13
3.2	Software	13
3.2.1	Integration into existing workflows	13
3.2.2	Services for Optimized Network Inference on Coprocessors	14
3.2.3	Industry tools	15
3.3	Lessons from industry	16
4	Resources	17
4.1	On- and off-detector real-time electronics	17
4.2	Cloud providers	18
4.3	High Performance Computing (HPC) Centers	19
5	Applications	20
5.1	Colliders	20
5.2	Neutrinos	20
5.3	Astrophysics	21

1 Introduction

The field of machine learning (ML) has exploded in the last decades. Many of the advances in the field have come hand in hand with advances in computing as algorithms have become increasingly complex, and ML is now in use across a wealth of research domains and industries. In physics, specifically, ML has become a widely used tool for applications ranging from event classification to particle identification to energy regression. New ML algorithms have improved both the performance and scope of these algorithms, but at the cost of algorithm complexity. This evolution results in algorithms that can become computationally intensive or slow, potentially to an extreme degree. As more complex algorithms are developed, and as the ML needs of physics research grow in the next decade, it will be vital for the physics community to be prepared with the computing tools and resources necessary to handle this growth.

Just as no single ML architecture is most appropriate for all problems, no single tool or resource for performing inference will address every physics use case effectively. Indeed, as technologies and fields evolve so too will the methods most adapted to each use. The goal of this white paper is not to lay out exactly how ML should be done for every physics application, but instead to present ideas that are capable of collectively addressing the needs of current and future physics experiments across frontiers. These options come in many forms. Hardware that can be faster and more efficient than traditional CPUs for inference is one possibility for reducing the overall computing load of ML. Indeed in some cases specialized hardware is the only possible way to perform inference fast enough to be used effectively. Other options for improved inference come in the form of software and computing paradigms. These tools can improve the overall latency and throughput of inference and also reduce the computing complexity, significantly lowering the cost for users to develop optimized workflows. Finally, there are multiple resource groups capable of enabling access to make use of these tools. Cloud computing and in-house clusters are both options, as are high-performance computing (HPC) centers. For environments that require microsecond-scale inference latencies, specialized resources for heavily leveraging FPGAs are necessary. Taken together, and used effectively, these tools and resources can enable the novel ML that we believe will help drive the field of physics forward in the next decade.

The layout of this white paper is as follows. Section 2 discusses the needs for ML in different frontiers, including colliders, neutrinos, and astrophysics. Section 3 presents the different hardware and software tools that are available for accelerated ML and their advantages. The necessary resources to properly utilize these tools for different use-cases are examined in Section 4, and some potential exciting applications are presented in Section 5. A summary and outlook is provided in Section 6.

2 Needs

The ML needs of every experiment are inherently different. These needs are dictated by many factors, such as the data formats, the system latencies, and the existing resources and

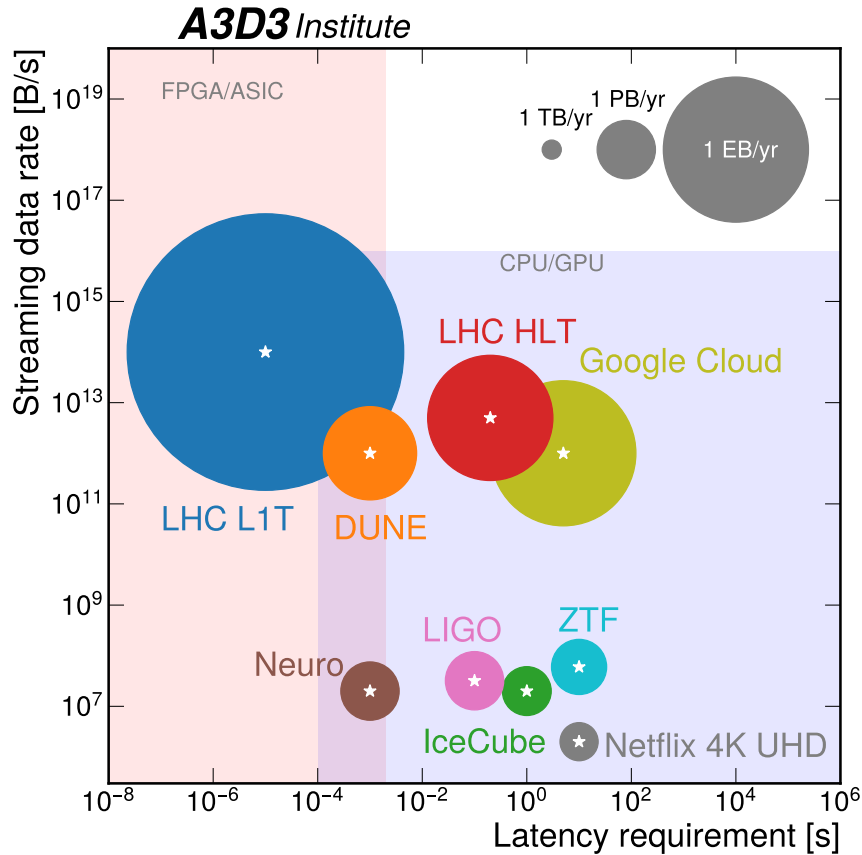


Figure 1: Plot of the streaming data rate in bytes per second and latency requirements in seconds for various experiments. Points of comparison from industry and other scientific fields are also included. The size of the bubbles represents the total per year data volume. Taken from Ref. [1].

workflows. In order to frame the discussion of tools and resources we present the ML needs in three distinct fields (collider physics, neutrino physics, and astrophysics). In each case ML has already begun to be widely adopted in certain contexts, and the needs are only expected to grow in the future. A summary of the approximate data rates and latencies for experiments across these frontiers is shown in Fig. 1.

2.1 Collider Physics

Data acquisition at collider experiments roughly fall into three separate tiers where reconstruction is applied at varying levels. At each tier the latency and throughput demands of the reconstruction are quite different making the deployment of machine learning algorithms also quite different. Moreover, the computing hardware between the different reconstruction

tiers is also quite different. The combination of both of these demands make it such that the deployment of ML algorithms, and both the size and usage of ML algorithms varies greatly across reconstruction tier.

In the first tier of reconstruction at a collider experiment, data is often collected at an incredible rate. At the Large Hadron Collider (LHC), collisions occur at a rate of 40 MHz. With a single collision carrying several megabytes of raw information, overall data rates can approach a petabit per second. As a result of the large data rate and the harsh collision environment, the full event information can only be stored for roughly 10 microseconds [2, 3]. As a result, an interconnected set of field-programmable gate arrays (FPGAs) are required to allow for the reconstruction of full events within the 10 microsecond window. This stage of reconstruction is known as the Level-1 trigger (L1T). As a result of these constraints, ML algorithms that are developed for this tier of reconstruction need to be small, and, depending on the reconstruction tier, to involve only local information. Furthermore ML algorithms are required to run within the strict latency constraints, typically less than a microsecond.

By selecting, on average, the most interesting collision amongst one hundred, the first tier of reconstruction reduces the overall data rate by two orders of magnitude. The second tier of reconstruction at a collider experiment, known as the high-level trigger (HLT), then takes this data and performs a much more thorough event reconstruction. Data selected for the HLT are shipped from the detector to a computing farm [4, 5]. As a result, the data are not stored in electronics subject to the harsh collider collision environment. These conditions along with the reduced data rate allow for a loosening of the latency constraint from a fixed 10 microseconds to roughly 1 second or less in some cases. Furthermore, the lower data rate allows for the use of more conventional CPUs and graphics processing units (GPUs) to perform the reconstruction. The larger latency and access to conventional processing hardware allows for a large array of ML algorithms, provided they can be run within the latency constraints. Since these constraints are softer, algorithms can batch inputs together to process many collisions in parallel, allowing for the possibility of additional throughput gains when deploying these algorithms.

The final tier of reconstruction, typically referred to as offline reconstruction, takes in collisions selected by the HLT at a rate of a few kHz. At this tier, there is no latency requirement, and algorithms can take up to seconds per collision to run. ML algorithms thus have no constraints. Furthermore, recent developments in offline computing have allowed for the possibility of GPUs or other heterogeneous computing to be deployed for ML inference [6].

New generations of collider experiments are starting to pursue variations on these ML approaches. In particular, the LHCb experiment has largely eliminated the first tier of reconstruction and instead built a significantly larger HLT [7]. Additionally, future nuclear physics experiments are pursuing “full-streaming” readout, whereby they aim to read out all of the data in its raw form without a trigger, and then reconstruct the data offline over a substantially longer timescale [8]. These variations limit the need for FPGA-based or readout-based ML, at the cost of demanding more complicated, higher throughput ML computing downstream.

2.2 Neutrino Physics

In neutrino physics, ML and in particular applications of image recognition for neutrino identification have been growing [9–11], due to the increasing use of large, high-resolution tracking calorimeters as neutrino detectors. Deep learning applications now span the full extent of data processing for neutrino experiments, including data acquisition [12–15], final data analysis (see, e.g. Ref. [16–18]), and in particular data reconstruction (see, e.g. Ref. [19–21]).

A detector technology ideally suited for computer vision applications in neutrino physics is that of liquid argon time projection chambers (LArTPCs)—employed by the future DUNE [22], current MicroBooNE [23], and upcoming SBN [24] experiments. These detectors function as stereoscopic image streaming devices, offering the possibility of direct application of image recognition at as early as the data acquisition stage, for triggering purposes. For the time being, however, applications of deep learning algorithms for these experiments have predominantly been focused on data reconstruction and final analysis tasks (see, e.g. Ref. [25–34]). Beyond LArTPC experiments, other neutrino experiments such as NOvA [35], MINERvA [36], Daya Bay [37], KamLAND-Zen [38], KM3NeT [21], and IceCube [39, 40] are also making extensive use of ML for reconstruction and analysis purposes.

Event record sizes for neutrino detectors can be as large as 100 TB (for a supernova burst in the DUNE far detector), and typically of order of ≤ 1 GB for the current generation of LArTPC experiments, challenging data reconstruction work flows. With the use of increasingly larger and higher-resolution detectors, and the need for continuous readout for off-beam, rare-event physics searches, data readout and trigger challenges for neutrino experiments also begin to approach those of current collider experiments. For example, the future DUNE [41] will be generating raw data rates of several terabytes per second, and plans to be operated for at least a decade and with a 100% live-time in order to be sensitive to supernova neutrinos and other rare and stochastic beam-unrelated signals. New developments in ML applications for neutrino physics target GPU-accelerated ML inference as-a-service (see Sec. 3.2) for computing in neutrino experiments for data reconstruction purposes (see, e.g. Ref. [42]), as well as GPU- or FPGA-based acceleration of ML algorithms such as 1D or 2D CNNs in real-time or online processing of raw LArTPC data at the data acquisition and trigger level [12–14].

2.3 Astrophysics

The growing number of astronomical instruments, including both ground- and space-based observatories, e.g. the Zwicky Transient Facility [43], the Submillimeter Array [44], the Very Large Array [45], the Neutron star Interior Composition Explorer (NICER) [46], the Neil Gehrels Swift Observatory [47, 48], amongst many others all across the electromagnetic spectrum, and starting recently also in gravitational waves [49], are challenging the existing data analysis paradigms.

So far, the usage of CPU and GPU clusters, combined with some level of human su-

pervision, has been generally sufficient to perform the required source detection, classification, and parameter estimations. However, there are several considerations that call for a paradigm shift. These include the exponential growth of data sets provided by the instruments, the interconnections between observations with all messengers (light, neutrinos, and gravitational waves), and the intrinsic timescales over which violent transient astrophysical sources with multi-messenger signatures develop, all of which make unattainable any human supervision/human-in-the-loop models, including requiring $O(1s)$ latencies for processing data and disseminating results within the broader community for follow-up.

Some areas of transient astronomy have long been using machine learning techniques, e.g. Ref. [50]. However, newly established gravitational-wave astronomy with the ground-based km-scale interferometers like Advanced LIGO [51], Advanced Virgo [52], and KAGRA [53] need a general infrastructure where ML models can be trained, deployed, and used for data analysis in a transparent way for the end-user, while factoring out the data acquisition and packaging protocols used in interferometry (see Ref. [54] for a summary of the various applications). Altogether, it is the combination of providing accessibility and enabling ML algorithms with dedicated hardware that will yield the necessary automation, scalability, and latencies to meet the challenges and discovery potential of multi-messenger astronomy and astrophysics.

3 Tools

The needs of physics experiments for fast and large-throughput ML inference will require a range of tools to address. These tools are emerging from both industry and within physics. We divide them into two categories: hardware and software. The first category encompasses specialized hardware that is capable of improved performance with respect to traditional CPUs, as well as the software that exists for enabling the deployment of algorithms to run on these specialized processors. The second category encompasses software capable of integrating ML inference into existing workflows and supporting at-scale computing of ML inference. We also present lessons from the deployment of ML in industry. We note that, although some tools for ML in industry may differ from those that are most useful in physics research, there is still much that can be learned.

3.1 Hardware

While CPUs represent the core technology for computing, they are not particularly efficient for ML inference. In order to enable low latency and high throughput inference alternative hardware will be a crucial addition to computing workflows. A summary of the most common hardware is shown in Fig. 2. GPUs are the most well-known hardware capable of performing ML inference faster than CPUs. Other alternative architectures like FPGAs, application-specific integrated circuits (ASICs), and new neuromorphic architectures (tensor processing units (TPUs) [55], intelligence processing units (IPUS) [56], and more) have also become popular more recently due to their low power usage and speed.



Figure 2: Silicon alternatives from most flexible (left) to most efficient (right), including CPUs, GPUs, FPGAs, and ASICs. TPUs (not pictured) are a relatively new development, which have flexibility similar to GPUs, but may have higher efficiency for ML inference. Figure adapted from Ref. [57].

3.1.1 GPUs

GPUs are specialized electronic circuits designed to rapidly manipulate and alter memory to accelerate the creation of images in a frame buffer intended for output to a display device. Figure 3 shows the architecture differences between CPUs and GPUs. Compared with CPUs, which are composed of a few cores with lots of cache memory good for serial processing, GPUs are composed of thousands of cores. This enables GPUs to perform *parallel* operations on very large sets of data with much lower power consumption. While individual CPU cores are faster and more flexible, the large number of GPU cores and massive amount of parallelism make up for the single-core speed difference. GPUs are therefore well-suited for repetitive and highly parallel computing tasks.

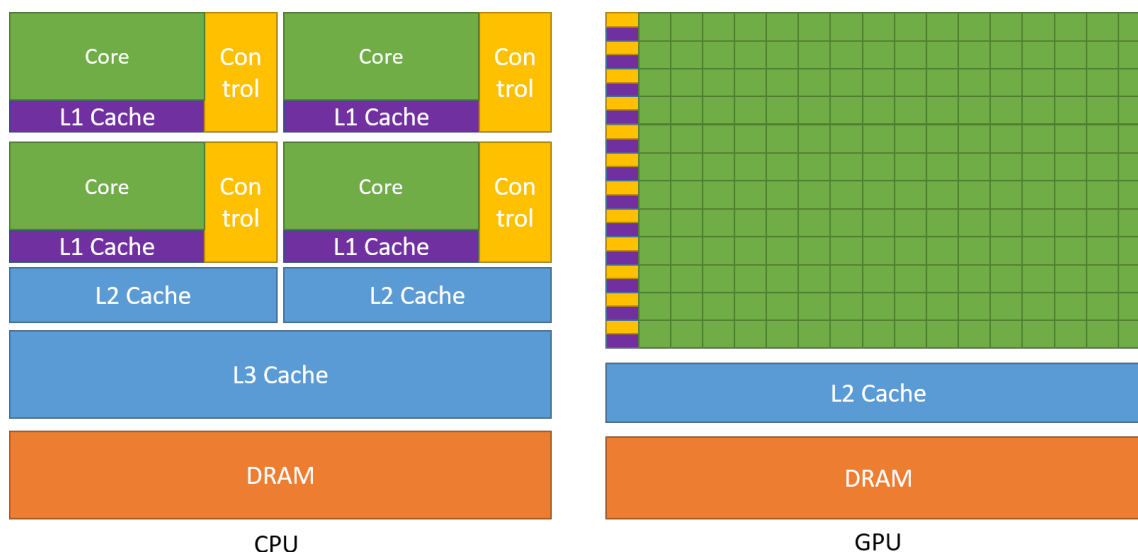


Figure 3: Architecture difference between CPUs and GPUs. Figure taken from Ref. [58].

Inference, as well as training, of ML models essentially consists of a large amount of

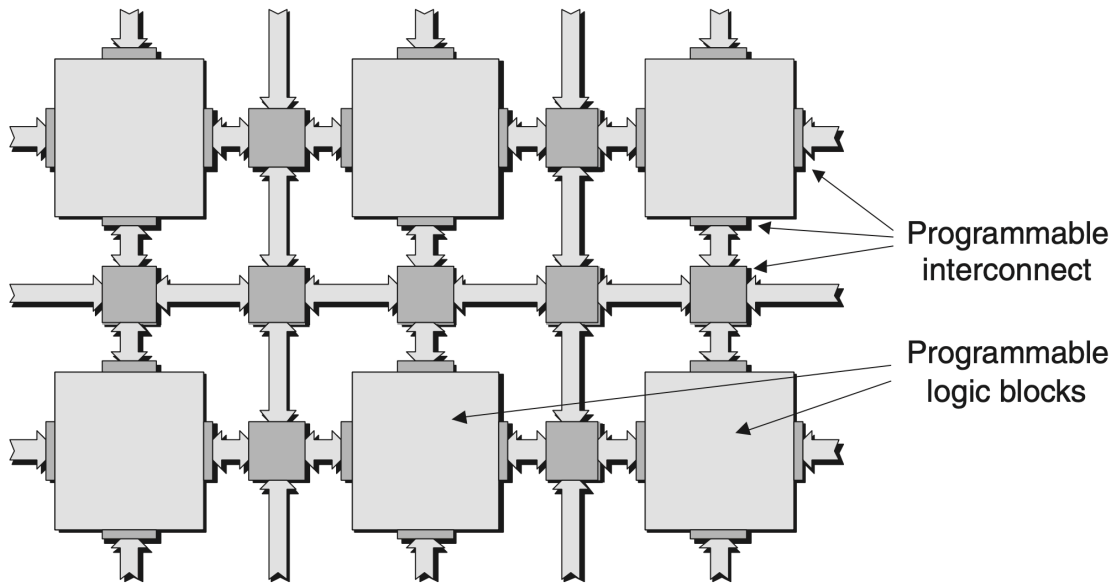


Figure 4: Simplified view of an FPGA architecture with programmable logic blocks and programmable interconnects. Figure adapted from Ref. [66]

complicated matrix operations. These operations are a very good fit for GPUs, especially when large batch sizes can be enabled. In the latest MLPerf inference benchmarks [59], the throughputs on NVIDIA A100 GPUs are about $\mathcal{O}(100)$ times faster than Intel Xeon CPUs. The inference latency is typically around $\mathcal{O}(1)$ ms to $\mathcal{O}(10)$ ms per batch, well below the HLT latency requirement ($\mathcal{O}(100)$ ms). Accelerating both HLT and offline productions with GPUs is therefore a very promising direction.

Currently widely-used GPU programming languages include CUDA [60], OpenCL [61], and OpenACC. Major ML frameworks are well-supported on GPUs, such as Tensorflow [62], PyTorch [63], and ONNX [64]. Besides quantization and pruning (discussed in the following section), there are also well-established and supported tools to optimize and accelerate ML inference on GPUs, such as TensorRT [65]. These tools have the potential to dramatically increase the throughputs and reduce the cost.

3.1.2 FPGAs

Field-programmable gate arrays (FPGAs) are digital integrated circuits that contain configurable (i.e., *programmable*) blocks of logic along with configurable interconnects between these blocks as shown in Fig. 4. Algorithm designers can program such devices to perform an array of tasks. Modern FPGAs also feature high bandwidth I/O connections and specialized components for multiplications (DSPs) or storing memory (block RAM).

Relative to CPUs, FPGAs can achieve much lower latencies for certain algorithms. In particular given the latency demands of Level-1 trigger systems at the LHC [2, 3, 67, 68],

which are of order microseconds, an FPGA-based design is necessary. Certain changes need to be made to the ML algorithms to enable efficient processing on an FPGA. Namely, these are: *quantization* [69–73], or the use of reduced precision operations, e.g. 16-bit fixed-point precision instead of 32-bit floating point precision, and *compression* [74–77], or the removal of unnecessary or redundant operations. As an example, a reference benchmark autoencoder for anomaly detection runs on a CPU in 10 ms, whereas a modified version of the same algorithm can be run on an FPGA in 9.6 μ s [78, 79].

Programming these devices requires highly specialized knowledge of register transfer-level (RTL) languages, like Verilog or VHDL, and vendor tools. For this reason, custom source-to-source compilers, also known as transpilers, have been developed that can lower the barrier to entry for deploying ML algorithms on FPGAs by generating FPGA-ready firmware directly from trained ML models. Many of these compilers leverage high-level synthesis (HLS) [80–83], which is an alternative way of generating hardware modules from code written in high-level programming languages like C/C++. Two such compilers are `hls4ml` [84] and FINN [85], although many others exist in the literature [86–94]. In particular, `hls4ml` has enjoyed wide usage within the particle physics community thanks to its flexibility and ease of use, especially for L1 trigger applications [3]. Current applications include binary and ternary neural networks [71], boosted decision trees [95], quantization-aware training [72], convolutional neural networks [96], graph neural networks [97, 98], and (variational) autoencoders [99].

Finally, FPGAs are not only applicable in the case of ultralow latency ML inference. They also have use in datacenters for high-throughput inference tasks [100, 101], such as for processing of large experimental datasets. This is discussed further in Sec. 3.2.2.

3.1.3 ASICs

For many particle physics experiments, the on-detector “front end” electronics consist of one or more ASICs working together to perform tasks such as data readout, calibration, and compression. Often an ASIC is the only choice for these tasks, due to strict requirements on power consumption, radiation tolerance, and latency. As detectors of the near future will generate data at unprecedented rates, these custom chips must be relied on to perform increasingly sophisticated processing in order for total recorded data volumes to be kept manageable, making ML algorithms a natural choice.

Here as in the case of FPGAs, specialized tools are required to port models trained with standard methods to a hardware design, specified in RTL. However, recent capabilities of certain transpilation tools to specifically target ASIC architectures has enabled the use of similar toolkits such as `hls4ml` to specify completely custom circuits [102]. In particular, this HLS flow enables quick prototyping so that designers can understand tradeoffs between model performance and the total chip area needed and estimated power consumption for their design.

Finally, the complete customizability of the ASIC provides for interesting options to allow for future flexibility of the ML model even once the final ASIC has been fabricated.

Ref. [103] illustrates an intermediate choice in which the CNN architecture is fixed, while weights may be reset via I2C registers, enabling reconfigurability to adapt to changing conditions over the lifetime of the experiment. For applications requiring radiation tolerance, triplication of these weight parameters can ensure robustness to single-event effects [104, 105].

3.1.4 IPU

Intelligence Processing Units (IPUs) are massively parallel processors developed by GraphCore, with the design aim of efficient executions of fine-grained operations across a relatively large number of parallel threads [106]. In contrast to GPUs, IPUs offer “Multiple Instruction Multiple Data” (MIMD) parallelism and adapt well to fine-grained, irregular computations that exhibit irregular data access. Fig. 5 provides a simplified view of an MK2 GC200 IPU architecture. Each IPU contains 1472 processing elements (“tiles”); each tile consists of one computing core and around 600KB of local memory. IPU tiles communicate among themselves via *links* and with CPU-based hosts via *exchange*.

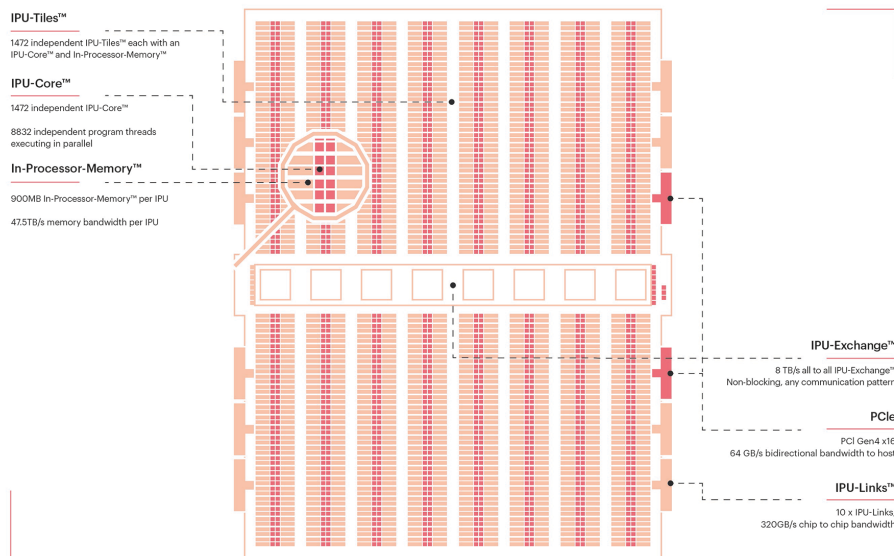


Figure 5: Simplified view of an IPU architecture. Figure taken from Ref. [56].

IPU programming is based on the Poplar SDK [107], co-designed with the IPU hardware. Currently the popular ML frameworks, such as Tensorflow [108], PyTorch [63], and ONNX [64], are well-supported with Poplar on IPUs. In the latest benchmark results released by GraphCore [109] on the training and inference of various popular ML models with different frameworks, IPUs outperform the state-of-art GPUs with higher throughputs and lower costs. Similar to GPUs, IPUs can potentially contribute to ML algorithm acceleration

for both HLT and offline computing in HEP.

3.1.5 Other specialized processors

Other next-generation processor technologies are starting to emerge that could potentially dramatically improve the ability to perform ML computations. By restricting the processor technology to target these computations, optimized processors can be developed that explicitly target ML operations. One intriguing future possibility comes from optical based ML processors [110–112]. Optical ML processors can operate at THz frequencies, leading to the possibility of running deep neural network inference within one nanosecond. Recent developments have allowed manufacturing these processors with conventional CMOS technology. Current optical-based technology is capable of running neural network inference with small networks ($\mathcal{O}(1,000)$ weights). Newer innovations could potentially allow for running inference for larger deep learning algorithms.

Tensor processing units (TPUs) are another type of optimized processor, designed by Google particularly for use with TensorFlow [55]. Like many other specialized processors, they make heavy use of quantization techniques to reduce latency and resources. TPUs are capable of processing hundreds of thousands of operations in a single clock cycle [113], and are optimized to perform operations like large matrix multiplications for training neural networks and image-based network inference, especially for very large networks. While TPUs are extremely performant at these tasks, these tasks do not represent a significant component of expected ML inference needs for physics.

3.2 Software

The architectures discussed above are capable of major performance improvements over standard CPU workflows, but achieving these performance improvements can be difficult. Industry tools have been designed in many cases to provide fast inference for single processes and maintaining this performance across the potential range of architectures and system designs is difficult. As a result of the popularity of GPUs, many industry tools are designed especially to make use of GPUs, but other tools exist from industry and in physics to make use of other architectures. Integrating these tools and architectures into existing workflows can be non-trivial, and as a result methods to simplify this process are vital, particularly at production scale. Paradigm shifts in computing such as the as-a-service model can enable more effective workflows not only for heterogeneous computing but also for CPU-only inference.

3.2.1 Integration into existing workflows

In a traditional computing model, a given CPU is responsible for processing the entirety of a given workflow, with each CPU capable of processing data in parallel to the others. The integration of alternative architectures, or coprocessors, into this paradigm can take

multiple different forms. The simplest option conceptually is to connect a coprocessor to each CPU, and to use existing tools to offload part of the workflow to the coprocessor. In the case that ML inference is a significant component of the workflow in terms of latency, the offloading of the inference to the coprocessor can provide a large reduction in the overall processing time. While this is effective for smaller systems, its use in larger systems with many CPUs requires many coprocessors. This can incur significant costs and also make it difficult to make full use of each coprocessor. An alternative paradigm, called computing “as-a-service” (aaS), attempts to overcome these limitations by eliminating the need for direct connection between each CPU and coprocessor [114–116]. In the aaS paradigm, coprocessors are connected to CPUs known as servers, and other CPUs known as clients communicate with the server CPUs over a network connection. Clients directly perform the non-accelerated components of the workflow and send the inputs necessary to perform the accelerated components to the server. This paradigm allows the management of the coprocessor to be separated from the clients, which can alleviate conflicts between the needs of ML inference and the needs of an existing system and workflow. Additionally, the aaS paradigm simplifies the addition or removal of coprocessors from the workflow, which can be extremely beneficial as architectures improve or are phased out of service.

3.2.2 Services for Optimized Network Inference on Coprocessors

Services for Optimized Network Inference on Coprocessors (SONIC) is a software design pattern to integrate a client-server approach for inference as a service into experiment software frameworks (which are usually based on C++). It offers useful abstractions to minimize dependence on specific features of the client interface provided by a given server technology. SONIC has been implemented in the CMS software [6, 100, 101] and in LArSoft for pro-toDUNE [42]; it is being explored by other experiments including ATLAS. Asynchronous, non-blocking calls are the most efficient approach [117], because the communication between the client and the server proceeds in parallel with other work continuing on the local CPU, therefore hiding the impact of transmission latency. However, synchronous, blocking calls may still provide significant speedups if a workflow is dominated by ML algorithm inference that can be offloaded to faster coprocessors. Here, we provide a brief overview of the advantages of SONIC. More details can be found in the aforementioned references, which also provide performance results showing speedups by more than an order of magnitude for a variety of ML algorithms.

- **Flexibility:** Allowing multiple clients to connect to multiple coprocessors enables many arrangements to ensure optimal usage of all devices.
- **Cost-effectiveness:** Related to flexibility, using coprocessors optimally reduces the number of coprocessors that must be purchased to support algorithm inference.
- **Symbiosis:** SONIC facilitates the use of existing industry tools and developments (see Section 3.2.3), rather than requiring HEP software developers to reimplement common tasks such as ML algorithm inference repeatedly for different ML frameworks, coprocessors, etc.

- **Simplicity:** SONIC modules only implement conversions of input and output data, which reduces the amount of code necessary to develop and maintain in order to perform ML algorithm inference.
- **Containerization:** The client-server model keeps the ML frameworks separate from the experiment software framework, which eliminates the significant workload needed to integrate two software systems that each have their own complicated dependencies.
- **Portability:** SONIC enables experiment software workflows to swap between CPUs, GPUs, FPGAs, and other coprocessors such as IPU without any code changes. In some cases, even the choice of ML framework can be changed with no other modifications.

It should be understood that the flexibility of SONIC is solely an advantage, in that it provides options, but does not require them to be used. In particular, SONIC facilitates the use of co-located or remote (e.g. cloud) coprocessor resources that may not be directly connected to worker CPUs. However, many of the other advantages of SONIC can still be realized even if it is just used as an abstraction to interact with directly-connected coprocessors. SONIC can also offload non-ML algorithms, but this may lack the advantage of automatic portability that comes with ML. While the use of separate servers and/or separate inference processes does add some complexity to experiment workflows, this is of a similar magnitude as, for example, conditions databases or remote data access protocols that are already widely used.

3.2.3 Industry tools

Existing implementations of SONIC [6, 42] focus on the open-source Triton inference server from Nvidia [118]. It offers a number of useful features, including:

- Support for all modern ML frameworks
- Support for non-ML algorithms and even non-Nvidia coprocessors via custom backends loaded by the server
- Standardized protocols, http or gRPC [119], to communicate between the client and the server
- Load balancing for multi-GPU servers
- Dynamic batching: combining multiple requests for the same algorithm to be processed more efficiently in GPU memory
- Usage of shared memory for faster communication with directly-connected processors (CPU or GPU)
- Input/output compression to conserve bandwidth

- Tools to optimize algorithm deployment on the server

The extensible nature of the server, with its standardized protocols and custom backends, has already been used to deploy servers that conduct inference on FPGAs or IPU rather than Nvidia GPUs. This enables the automatic portability that is a key advantage of the SONIC approach. Previous implementations of SONIC have used the Microsoft Brainwave API [120] as well as wrapping the TensorFlow C++ API. In the future, other client-server technologies such as the interprocess communication (IPC) provided by Apache Arrow [121] could be considered.

The use of industry client-server technologies enables, though it does not require, the use of commercial cloud resources and associated software tools. All major providers—Google Cloud, Amazon Web Services, and Microsoft Azure—have been used successfully to host coprocessor servers that provide GPUs or FPGAs. The option to use cloud computing is also useful to test new devices such as IPU [122]. Kubernetes [123] is an extremely powerful and versatile tool to orchestrate these cloud servers, and it is therefore worth investigating seriously for HEP-specific grid computing centers as well.

3.3 Lessons from industry

Significant effort in industry has been dedicated to developing software frameworks, computing and networking systems, and hardware accelerators to meet the broad range of demands for ML solutions in various fields. These range from smart Internet of Things (IoT) domains (e.g., smart city, smart manufacturing, wearable healthcare devices, and security surveillance) to computer vision (e.g., autonomous driving, unmanned aerial vehicles, and augmented/virtual reality) to data analytics applications (e.g., those encountered in particle physics for scientific discovery). Many industry tools focus on ease of use, and as such development flows have been streamlined and been made highly efficient. The importance of these easy-to-use frameworks and tools such as PyTorch [63], TensorFlow [62, 108], and ONNX [64] remaining open source is of vital importance to the physics community. Tools that are not open source can be difficult to use due to a lack of ability to adapt to the specifics of physics workflows.

For low-latency solutions that may require FPGAs as hardware accelerators, the commercial HLS tools from major FPGA vendors, such as AMD/Xilinx and Intel/Altera, are of critical importance. The broad usage of FPGAs (and ASICs) for ML in physics is highly dependent on these HLS tools. These vendors also offer ML-specific design flows. For example, AMD/Xilinx offers the Vitis AI development environment for AI inference on AMD/Xilinx hardware platforms, that contains optimized IP, tools, libraries, models, and example designs. It also offers AI Model Zoo, which provides optimized and retrainable AI models for fast deployment and high-performance accelerations. It is useful to note that the low-latency regime for industry is typically focused on latencies of approximately 1 ms. While these latencies are appropriate for many physics applications, there also exists cases with latency needs in the sub-microsecond regime, far below the target of most industry tools.

Industry has seen much success through efforts coordinated between users, application developers, hardware vendors, and computing and networking solution providers to help create a highly functional ecosystem. This is essential for the fast growth of ML-driven businesses. The scientific community can learn from this and carry out coordinated efforts among researchers and practitioners in physics, computer science, engineering, and hardware systems to achieve our ultimate research goals. We should also continue to develop dedicated and customized HLS solutions, compilers, and hardware systems in order to achieve high computation efficiency and low latency for the ML algorithms specific to our scientific domains.

4 Resources

While some physics workflows have already begun to incorporate coprocessors, many are still limited only to CPUs. The variety of resources for accessing coprocessors will be important as ML inference and computing needs expand. While some experiments are able to make use of more generic options, many applications will require special resources.

4.1 On- and off-detector real-time electronics

Machine learning integrated into the detector real-time electronics provides powerful data reduction capabilities, particularly for very high data rate or low latency applications at the LHC, DUNE, particle accelerators, and many more [124]. We are considering here applications ranging from on-detector data concentrator or aggregation in ASIC or FPGAs to off-detector real-time trigger or filtering based on feature extraction of physics observables. To consider such a wide range of custom requirements for each specific application—with different bandwidths, latencies, interfaces, hardware platforms—makes developing common resources challenging. Therefore we broadly consider two types of resources needed for the development of custom electronics ML solutions: hardware platforms and electronic design automation (EDA) tools.

Hardware platforms can vary from custom ASIC chips to custom FPGA systems to off-the-shell electronics boards. Custom hardware solutions are tailored to specific experiments, particularly in the case of ASICs, but there are some FPGA readout systems that can serve multiple experiments such as OTSDAQ/CAPTAN [125, 126], RCE [127], and FELIX [128]. Beyond that, flexible off-the-shelf solutions for use-cases with less stringent requirements are becoming increasingly popular. There are a wide range of examples that make compiling a complete list challenging. They can range from commercially available system-on-chip (SoC) or system-on-module (SoM) partial solutions that can be integrated into a larger system—for example, an Arria10 SoM [129] being explored for accelerator controls—to development kits to complete solutions like the Alveo [130]. The Alveo is even available in the cloud [131] and can be used for prototyping.

Resources for electronics synthesis and integration can be extremely expensive, including maintaining licenses and also support more generally. In the previous section, we discussed

tools for ML translation to hardware description language; however, integrating those algorithms into FPGA or ASIC systems requires licenses for expensive EDA tools. FPGA tools such as Vivado [132] and Quartus [133] are not relatively expensive compared to ASIC tools, but are much more widely used, and university and laboratories need to reserve resources to maintain licensing of those tools. For ASIC development, the cost can be an order of magnitude or more greater for EDA tools (which can also synthesize designs for FPGAs) from vendors like Cadence [134], Synopsys [135] and Siemens [136]. Two potential paths that should be supported to reduce resources include: (a) joint agreements across laboratories and universities and the vendors to reduce overall costs and (b) exploration of open-source solutions. For (b) in particular, the trend towards open-source hardware both for FPGAs and ASICs continues to grow with the development of tools like Symbiflow [137], Yosys [138], and SkyWater SKY130 PDK [139].

4.2 Cloud providers

Perhaps the simplest way to acquire large-scale computing resources on a short timescale is the utilization of cloud resources, such as Google Cloud Platform (GCP) [140] or Amazon Web Services (AWS) [141]. By coordinating with the provider, it is straightforward to acquire $\mathcal{O}(10,000)$ CPU threads and $\mathcal{O}(100)$ GPUs for use at a given time. There is also flexibility to choose from a variety of CPU platforms and GPU types [142–144], change the platform at will, and create custom images to deploy in all instances.

To manage these large-scale resources, one can use pre-existing paradigms, such as HEPCloud [145], which emulates the HTCondor distributed computing system, dynamically allocating and removing worker nodes in the cloud as needed. It is also possible to build a custom SLURM workload manager in the cloud. The HEPCloud approach allows for greater ease of access, as relatively little user-side expertise is needed, while the SLURM approach allows for somewhat more control over resource configuration and the use of GPUs in worker nodes, though a greater deal of user-side effort and knowledge is required to correctly configure cluster networks and images.

While the cloud provides for quick large-scale access to computing resources, costs can be prohibitive for long term running. Rates for CPU-only nodes are in the range of $\mathcal{O}(\$1 - \$10)$ per day per thread, depending on the memory, disk, and platform specifics, and GPU-enabled nodes cost $\mathcal{O}(\$1,000)$ per month for continual operation. The cost of CPU-enabled nodes can be reduced by a factor of about 5 by using pre-emptible or ephemeral nodes (which can be “taken” by other users with higher priority), though the feasibility of this option depends on the use case.

In spite of this high cost, there are certain cases that are very well-suited to the cloud. Short-term development and testing can make effective use of the pool of heterogeneous resources, without requiring full purchases of a resource before its potential is understood. Several proof-of-principle studies have demonstrated the possibility of deploying heterogeneous computing platforms, such as GPU and FPGA coprocessors, in HEP software with an innovative “as-a-service” approach [6, 101]. The SONIC software framework was developed during these studies, which were made possible through large-scale heterogeneous cloud

resources, and the SONIC-based GPU- and FPGA-as-a-service toolkits have been released as open source code. Scale-up tests of computing scenarios are another scenario for which the cloud is an appealing option. The cloud can be used in these cases to make heavy use of a huge amount of resources for a short period of time. For example, an initial test using a small number of simulated ProtoDUNE events showed a viable, cost-effective way to use the SONIC framework to solve the computing challenges facing the neutrino experiments [42]. In 2021, to take advantage of an improved machine learning model, the entire 7 million ProtoDUNE beam data events were reprocessed using this framework. The acceleration provided by GPUs accessed through Google Cloud reduced the total processing time by half compared to the CPU-only approach. Similar tests have also shown the possibility of large speed-ups for gravitational-wave data processing by using cloud resources [146].

In the future, it is likely that cloud computing costs will decrease. Cloud computing providers are able—in principle—to procure computing resources at a fraction of our cost due to the very large scale of their procurement processes. They also can benefit from economies of scale in operation and development. Additionally, they rent computing power in a free market (sized at more than \$300 billion), competing with not only on-premises resources but other cloud computing companies.

4.3 High Performance Computing (HPC) Centers

There are a number of large HPC centers supported by the National Science Foundation and the Department of Energy. While many of these facilities are optimized for traditional high-performance computing jobs (requiring low-latency communication between nodes), they are also capable of executing high-throughput computing jobs (requiring little or no intranode communication). Work is executed using a fair-share scheduler. There are several challenges with federating these resources across multiple sites. The network connectivity between the compute nodes and the outside world may be underprovisioned (or in some cases, non-existent). Some systems are secured with multi-factor authentication and many do not have API endpoints for workload submission available off-site. Many centers use an annual proposal/review/award allocation model, which is not a good fit for HEP experiments that can process data over the course of decades.

Some HPC facilities deploy significant numbers of GPUs (e.g. Summit Oak Ridge National Laboratory, 27k NVIDIA V100 GPUs; Perlmutter NERSC, 6,000 NVIDIA A100 GPUs). Training of machine learning models, which can make efficient use of GPU to CPU ratios of 2:1 (or significantly more, depending on the model), can be executed on the machines using standard fair-share scheduling. However, processing data with the inference-as-a-service model—which may only need GPU to CPU ratios of 1:300 or less [6]—requires different co-scheduling of GPU and CPU resources. Addressing this difference may require direct coordination with HPC staff and changes to the facility scheduling model.

5 Applications

Although by no means a comprehensive summary, we offer a select few applications of ML that can be enabled through the effective use of the tools and resources described above. As in Section 2, we divide the applications by frontier.

5.1 Colliders

The stringent latency and resource constraints imposed by the L1 trigger at the LHC make it an ideal target for applying innovative ML methods to embedded systems. There have been several examples of the application of machine learning models based on high-level synthesis tools [84, 95, 96, 147–149] from FPGA vendors for tasks such as the reconstruction and calibration of final objects or lower-level inputs like trajectories [98], vertices, calorimeter clusters [3], and identification of long-lived particles [150]. Alternative approaches are being considered based directly on hardware description languages, such as VHDL [151], for example, the real-time signal processing of the ATLAS Liquid Argon calorimeter [152, 153]. Anomaly detection techniques such as autoencoders are being explored to efficiently suppress the SM background contribution without imposing stringent kinematic constraints [99, 154–156]. Deploying such a triggering mechanism at L1 trigger on FPGAs using the `hls4ml` tool can significantly enhance the sensitivity to new physics and rare SM processes. Modern architectures such as graph neural networks (GNNs) are being explored for the reconstruction of particle trajectories, showers in the calorimeter as well as of the final individual particles in the event. Key GNN applications can only be realized at scale with optimized hardware inference for GNNs, which is an innovative topic within the field of ML algorithm-hardware co-design.

Accelerated ML can be used within the LHC HLT and offline computing workflows via heterogeneous computing platforms for algorithms with longer latencies. The proof-of-principle studies discussed in Section 4.2 provide an avenue toward “Big Data” science computing with scalability, low software maintenance cost, and maximized hardware flexibility. SONIC has been integrated into the CMS experiment software stack. The end-to-end GPU- and FPGA-based workflows for processing data in real-time heterogeneous systems is the practical approach to enable deployment of complexity algorithms needed for massive data processing.

5.2 Neutrinos

Machine learning algorithms are becoming increasingly prevalent and performant in the reconstruction of events in neutrino experiments. These sophisticated algorithms can be computationally expensive especially for detectors on the Earth’s surface or close to a neutrino source where there are lots of activities inside the detector. In order to improve the efficiency and speed of the inference of ML algorithms in a large-scale data processing, GPU acceleration specifically for the ProtoDUNE reconstruction chain has been integrated without disrupting the native computing workflow using SONIC [42] (as mentioned in Sec-

tion 4.2). With the integrated framework, the most time-consuming task, track and particle shower hit identification, is accelerated by a factor of 17. This results in a factor of 2.7 reduction in the total processing time when compared with CPU-only production. Future developments are expected to include: optimizing ML algorithms for GPUs with TensorRT using approaches such as quantization; studies using other hardware such as FPGAs, IPU (an initial test of an IPU in collaboration with GraphCore was shown to be very promising); and exploring deployment at scale at the Feynman Computing Center (at FNAL), NERSC, and other HPC centers.

On the latency and throughput front, the future DUNE Far Detector represents a special case, where information from any contiguous region of the detector, up to several GB per region, must be processed in parallel, with millisecond latency, and selected or triggered on with high accuracy. Due to the nature of neutrino interactions, computer vision algorithms such as CNNs were the first to be explored for this task. Techniques such as network quantization have been explored to further reduce latency and resource utilization for FPGA deployment. It has been shown that a relatively simple CNN can be implemented with reasonable FPGA resource utilization to select events with sufficiently low latency and with accuracy as required by the DUNE physics program [12, 157]. Future developments include demonstration of this application using real data, either at ProtoDUNE or the Short Baseline Neutrino Detector (SBND).

5.3 Astrophysics

Recent work has made possible a novel implementation and deployment of a deep learning inference infrastructure for real-time gravitational-wave data analyses pipelines [146]. This model enables easy integration of ML algorithms that can be used for denoising [158] and astrophysical source identification [159], including the ability to scale and incorporate hardware acceleration. Noise regression in gravitational-wave detectors is a challenging task and paramount to be performed as close to real-time as possible, since any denoising may directly improve astrophysical reach and thus discoveries that might require multi-messenger follow-up. The ML approach to this problem enables the identification and removal of subtle features in the data, going beyond linear couplings between the gravitational-wave channel and the environment/interferometry as captured in a wealth of auxiliary witness channels. Aside from denoising, ML algorithms for gravitational-wave source detection [159] and parameter estimation [160] are being benchmarked for full deployment during upcoming observing runs of the international network of gravitational-wave detectors. Use of ML approaches in traditional electromagnetic astronomy has a longer history than in gravitational waves. Supervised and unsupervised approaches have been used for object classification, including identification of artefacts, both at the pixel and light curve level. With the rate at which data sets are growing, ML has become commonplace. The ability to seamlessly incorporate ML models and further invoke hardware accelerators is expected to reduce the resource footprint and attain intrinsic latencies at sub-second levels.

6 Summary and Outlook

Machine learning is a powerful tool for research and offers real promise for many challenges that will face physics experiments in the coming decade. However, ML algorithms can become very expensive computationally, and while traditional CPU-based computing will be sufficient in some cases, many others will require the use of alternative hardware to meet power, throughput, and latency constraints. Of the options, it seems clear that GPUs will be the choice for many needs, as they offer the most mature tools for usage and offer large speedups for most medium to large models. FPGAs will also see use in certain situations, particularly for smaller networks and low latency applications where they are the only option such as the L1 trigger at the LHC. Although they are not expected to play a significant role in most computing for ML in physics, the use of ASICs will be absolutely necessary in some environments like those with high radiation that simply do not allow for the use of FPGAs or GPUs. The field should also be prepared to adapt to new coprocessors such as neuromorphic architectures like IPU, optical chips, or future technologies which may emerge as strong competitors to existing devices in the next decades.

In addition to the choice of hardware for ML applications, equally as important are the modes of model deployment and access. Alternative architectures that require specialized programming languages present a significant barrier to entry for most physics users. Compilers and transpilers from industry reduce this barrier significantly, as well as allow more efficient prototyping. Tools from inside physics will also be critical for uses that are distinct from those targeted most by industry. Two such tools of note are `hls4ml` and SONIC. `hls4ml` is already in wide use for applications with ultra-low latency requirements, and its potential use cases are only expected to grow as the bounds of experiment capabilities are pushed. SONIC and other tools to enable as-a-service computing are expected to be significant components of large computational workflows comprised of both ML and non-ML algorithms. These tools have many advantages that make them suited not only for single physics applications but a wide range of experiments and use cases across the frontiers we have highlighted. In all cases, both for tools from industry and from physics, it is very important that tools remain open-source to allow for collaboration and development that meets the specific needs of physics applications.

Access to the computing infrastructure necessary to execute these workflows will be critical for future success. For specialized on- and off-detector real-time electronics, the necessity of dealing with commercial entities for licensing will necessitate communication and collaboration between laboratories and universities or the exploration of open-source solutions. This extends to the use of hardware solutions that can be shared across experiments in the case of FPGA readout systems. As needs for experiments evolve, it is important to consider how different resources are able to scale and meet these needs. Elastic computing offered by cloud providers is clearly scalable; as we have noted, it is currently costly for sustained usage, but costs may decrease in the future. Collaboration with HPC centers will be important to ensure they can be utilized in ways most effective for experimental workflows that may not require their baseline of 2:1 ratios of CPU to accelerator.

The next decade will present significant challenges to physics research, and ML will be

a necessary component in overcoming some of these challenges. The tools and resources necessary to enable these ML solutions will come both from physics and industry, and will come in the form of both dedicated hardware and software and computing paradigms. Continued collaboration with industry and HPC centers will be critical to handling the rapidly changing landscape of ML. This can enable the not only the exciting applications discussed here but also those that are to come in the future.

References

- [1] A3D3 Institute, “A3D3 Institute.” <https://a3d3.ai/>, 2022.
- [2] ATLAS collaboration, *Operation of the ATLAS trigger system in Run 2*, *JINST* **15** (2020) P10004 [[2007.12539](#)].
- [3] CMS collaboration, *The Phase-2 upgrade of the CMS Level-1 trigger*, CMS Technical Design Report [CERN-LHCC-2020-004](#), [CMS-TDR-021](#) (2020).
- [4] CMS collaboration, *The Phase-2 Upgrade of the CMS Data Acquisition and High Level Trigger*, Technical Design Report [CERN-LHCC-2021-007](#), [CMS-TDR-022](#) (2021).
- [5] ATLAS collaboration, *Technical Design Report for the Phase-II Upgrade of the ATLAS TDAQ System*, Technical Design Report [CERN-LHCC-2017-020](#), [ATLAS-TDR-029](#) (2017).
- [6] J. Krupa et al., *GPU coprocessors as a service for deep learning inference in high energy physics*, *Mach. Learn. Sci. Tech.* **2** (2021) 035005 [[2007.10359](#)].
- [7] LHCb collaboration, *Design and performance of the LHCb trigger and full real-time reconstruction in Run 2 of the LHC*, *JINST* **14** (2019) P04013 [[1812.10790](#)].
- [8] F. Ameli et al., *Streaming readout for next generation electron scattering experiment*, [2202.03085](#).
- [9] A. Radovic, M. Williams, D. Rousseau, M. Kagan, D. Bonacorsi, A. Himmel et al., *Machine learning at the energy and intensity frontiers of particle physics*, *Nature* **560** (2018) 41.
- [10] F. Psihas, M. Groh, C. Tunnell and K. Warburton, *A Review on Machine Learning for Neutrino Experiments*, *Int. J. Mod. Phys. A* **35** (2020) 2043005 [[2008.01242](#)].
- [11] G. Karagiorgi, G. Kasieczka, S. Kravitz, B. Nachman and D. Shih, *Machine Learning in the Search for New Fundamental Physics*, [2112.03769](#).
- [12] Y.-J. Jwa, G.D. Guglielmo, L.P. Carloni and G. Karagiorgi, *Accelerating Deep Neural Networks for Real-time Data Selection for High-resolution Imaging Particle Detectors*, in *2019 New York Scientific Data Summit: Data-Driven Discovery in Science and Industry*, 6, 2019, DOI [[2201.04740](#)].

- [13] ARGONEUT collaboration, *A deep-learning based raw waveform region-of-interest finder for the liquid argon time projection chamber*, [2103.06391](#).
- [14] L. Uboldi, D. Ruth, M. Andrews, M.H.L.S. Wang, H.-J. Wenzel, W. Wu et al., *Extracting low energy signals from raw LArTPC waveforms using deep learning techniques – A proof of concept*, [2106.09911](#).
- [15] ARIANNA collaboration, *A novel trigger based on neural networks for radio neutrino detectors*, *PoS ICRC2021* (2021) 1074.
- [16] 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](#)].
- [17] NOvA collaboration, *Search for Active-Sterile Antineutrino Mixing Using Neutral-Current Interactions with the NOvA Experiment*, *Phys. Rev. Lett.* **127** (2021) 201801 [[2106.04673](#)].
- [18] MICROBOONE collaboration, *Search for an anomalous excess of charged-current quasi-elastic ν_e interactions with the MicroBooNE experiment using Deep-Learning-based reconstruction*, [2110.14080](#).
- [19] P. Baldi, J. Bian, L. Hertel and L. Li, *Improved Energy Reconstruction in NOvA with Regression Convolutional Neural Networks*, *Phys. Rev. D* **99** (2019) 012011 [[1811.04557](#)].
- [20] MICROBOONE collaboration, *Wire-Cell 3D Pattern Recognition Techniques for Neutrino Event Reconstruction in Large LArTPCs: Algorithm Description and Quantitative Evaluation with MicroBooNE Simulation*, [2110.13961](#).
- [21] KM3NET collaboration, *Event reconstruction for KM3NeT/ORCA using convolutional neural networks*, *JINST* **15** (2020) P10005 [[2004.08254](#)].
- [22] DUNE collaboration, *Deep Underground Neutrino Experiment (DUNE), Far Detector Technical Design Report, Volume I Introduction to DUNE*, *JINST* **15** (2020) T08008 [[2002.02967](#)].
- [23] MICROBOONE collaboration, *Design and Construction of the MicroBooNE Detector*, *JINST* **12** (2017) P02017 [[1612.05824](#)].
- [24] MICROBOONE, LAR1-ND, ICARUS-WA104 collaboration, *A Proposal for a Three Detector Short-Baseline Neutrino Oscillation Program in the Fermilab Booster Neutrino Beam*, [1503.01520](#).
- [25] DUNE collaboration, *Neutrino interaction classification with a convolutional neural network in the DUNE far detector*, *Phys. Rev. D* **102** (2020) 092003 [[2006.15052](#)].
- [26] DUNE collaboration, *Deep-Learning-Based Kinematic Reconstruction for DUNE*, [2012.06181](#).

- [27] MICROBOONE collaboration, *Convolutional Neural Networks Applied to Neutrino Events in a Liquid Argon Time Projection Chamber*, *JINST* **12** (2017) P03011 [[1611.05531](#)].
- [28] MICROBOONE collaboration, *Convolutional neural network for multiple particle identification in the MicroBooNE liquid argon time projection chamber*, *Phys. Rev. D* **103** (2021) 092003 [[2010.08653](#)].
- [29] MICROBOONE collaboration, *Deep neural network for pixel-level electromagnetic particle identification in the MicroBooNE liquid argon time projection chamber*, *Phys. Rev. D* **99** (2019) 092001 [[1808.07269](#)].
- [30] MICROBOONE collaboration, *Semantic segmentation with a sparse convolutional neural network for event reconstruction in MicroBooNE*, *Phys. Rev. D* **103** (2021) 052012 [[2012.08513](#)].
- [31] L. Dominé and K. Terao, *Scalable deep convolutional neural networks for sparse, locally dense liquid argon time projection chamber data*, *Phys. Rev. D* **102** (2020) 012005 [[1903.05663](#)].
- [32] SBND collaboration, *Cosmic Background Removal with Deep Neural Networks in SBND*, [2012.01301](#).
- [33] F. Drielsma, K. Terao, L. Dominé and D.H. Koh, *Scalable, End-to-End, Deep-Learning-Based Data Reconstruction Chain for Particle Imaging Detectors*, in *34th Conference on Neural Information Processing Systems*, 2, 2021 [[2102.01033](#)].
- [34] C. Adams, K. Terao and T. Wongjirad, *PILArNet: Public Dataset for Particle Imaging Liquid Argon Detectors in High Energy Physics*, [2006.01993](#).
- [35] NOvA collaboration, *The Convolutional Visual Network for Identification and Reconstruction of NOvA Events*, *J. Phys. Conf. Ser.* **898** (2017) 072053.
- [36] MINERvA collaboration, *Reducing model bias in a deep learning classifier using domain adversarial neural networks in the MINERvA experiment*, *JINST* **13** (2018) P11020 [[1808.08332](#)].
- [37] E. Racah, S. Ko, P. Sadowski, W. Bhimji, C. Tull, S.-Y. Oh et al., *Revealing Fundamental Physics from the Daya Bay Neutrino Experiment using Deep Neural Networks*, [1601.07621](#).
- [38] KAMLAND-ZEN collaboration, *Search for Majorana Neutrinos near the Inverted Mass Hierarchy Region with KamLAND-Zen*, *Phys. Rev. Lett.* **117** (2016) 082503 [[1605.02889](#)].
- [39] ICECUBE collaboration, *Graph Neural Networks for IceCube Signal Classification*, [1809.06166](#).
- [40] ICECUBE collaboration, *Reconstruction of Neutrino Events in IceCube using Graph Neural Networks*, *PoS ICRC2021* (2021) 1044 [[2107.12187](#)].

- [41] DUNE collaboration, *Deep Underground Neutrino Experiment (DUNE), Far Detector Technical Design Report, Volume III: DUNE Far Detector Technical Coordination*, *JINST* **15** (2020) T08009 [[2002.03008](#)].
- [42] M. Wang, T. Yang, M. Acosta Flechas, P. Harris, B. Hawks, B. Holzman et al., *GPU-Accelerated Machine Learning Inference as a Service for Computing in Neutrino Experiments*, *Front. Big Data* **3** (2021) 604083 [[2009.04509](#)].
- [43] M.J. Graham, S.R. Kulkarni, E.C. Bellm, S.M. Adams, C. Barbarino, N. Blagorodnova et al., *The Zwicky Transient Facility: Science Objectives*, *Publications of the ASP* **131** (2019) 078001 [[1902.01945](#)].
- [44] P.T.P. Ho, J.M. Moran and K.Y. Lo, *The submillimeter array*, *The Astrophysical Journal* **616** (2004) L1–L6.
- [45] R.A. Perley, C.J. Chandler, B.J. Butler and J.M. Wrobel, *The Expanded Very Large Array: A New Telescope for New Science*, *Astrophysical Journal, Letters* **739** (2011) L1 [[1106.0532](#)].
- [46] K.C. Gendreau, Z. Arzoumanian, P.W. Adkins, C.L. Albert, J.F. Anders, A.T. Aylward et al., *The Neutron star Interior Composition Explorer (NICER): design and development*, in *Space Telescopes and Instrumentation 2016: Ultraviolet to Gamma Ray*, vol. 9905 of *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, p. 99051H, July, 2016, [DOI](#).
- [47] D.N. Burrows, J.E. Hill, J.A. Nousek, J.A. Kennea, A. Wells, J.P. Osborne et al., *The Swift X-Ray Telescope*, *Space Sci. Rev.* **120** (2005) 165 [[astro-ph/0508071](#)].
- [48] P.W.A. Roming, T.E. Kennedy, K.O. Mason, J.A. Nousek, L. Ahr, R.E. Bingham et al., *The Swift Ultra-Violet/Optical Telescope*, *Space Sci. Rev.* **120** (2005) 95 [[astro-ph/0507413](#)].
- [49] The LIGO Scientific Collaboration, The Virgo Collaboration and The KAGRA Collaboration, *Gwtc-3: Compact binary coalescences observed by ligo and virgo during the second part of the third observing run*, 2021. [10.48550/ARXIV.2111.03606](#).
- [50] J.S. Bloom, J.W. Richards, P.E. Nugent, R.M. Quimby, M.M. Kasliwal, D.L. Starr et al., *Automating discovery and classification of transients and variable stars in the synoptic survey era*, *Publications of the Astronomical Society of the Pacific* **124** (2012) 1175.
- [51] LIGO SCIENTIFIC COLLABORATION collaboration, *Advanced ligo, Classical and Quantum Gravity* **32** (2015) 074001.
- [52] ADVANCED VIRGO: A SECOND-GENERATION INTERFEROMETRIC GRAVITATIONAL WAVE DETECTOR collaboration, *Advanced Virgo, Classical and Quantum Gravity* **32** (2015) 024001.

- [53] *KAGRA: 2.5 generation interferometric gravitational wave detector*, *Nature Astronomy* **3** (2019) 35.
- [54] E. Cuoco, J. Powell, M. Cavaglià, K. Ackley, M. Bejger, C. Chatterjee et al., *Enhancing gravitational-wave science with machine learning*, *Machine Learning: Science and Technology* **2** (2020) 011002.
- [55] N.P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa et al., *In-datacenter performance analysis of a tensor processing unit*, 2017. 10.48550/arXiv.1704.04760.
- [56] Graphcore, “IPU Products.” <https://www.graphcore.ai/products>, 2022.
- [57] Azure Machine Learning, “Deploy ML models to field-programmable gate arrays (FPGAs) with Azure Machine Learning.” <https://docs.microsoft.com/en-us/azure/machine-learning/how-to-deploy-fpga-web-service>, 2022.
- [58] NVIDIA, “CUDA C++ Programming Guide.” <https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html>, 2022.
- [59] MLCommons, “Inference: Datacenter v1.1 results.” <https://mlcommons.org/en/inference-datacenter-11/>, 2022.
- [60] J. Nickolls et al., *Scalable Parallel Programming with CUDA: Is CUDA the Parallel Programming Model That Application Developers Have Been Waiting For?*, *Queue* **6** (2008) 40–53.
- [61] Du, Peng and others, *From CUDA to OpenCL: Towards a performance-portable solution for multi-platform GPU programming*, *Parallel Computing* **38** (2012) 391.
- [62] TensorFlow Developers, *TensorFlow*, 2022. 10.5281/zenodo.5949169.
- [63] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan et al., *Pytorch: An imperative style, high-performance deep learning library*, in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox and R. Garnett, eds., pp. 8024–8035, Curran Associates, Inc. (2019), <http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf>.
- [64] J. Bai, F. Lu, K. Zhang et al., “Onnx: Open neural network exchange.” <https://github.com/onnx/onnx>, 2022.
- [65] NVIDIA, “TensorRT.” <https://developer.nvidia.com/tensorrt>, 2022.
- [66] C. Maxfield, *Chapter 5 - Programming (Configuring) an FPGA*, in *The Design Warrior's Guide to FPGAs*, C. Maxfield, ed., (Burlington), pp. 99–114, Newnes (2004), DOI.
- [67] CMS collaboration, *Performance of the CMS Level-1 trigger in proton-proton collisions at $\sqrt{s} = 13$ TeV*, *JINST* **15** (2020) P10017 [2006.10165].

- [68] ATLAS collaboration, *Technical Design Report for the Phase-II Upgrade of the ATLAS TDAQ System*, ATLAS Technical Design Report [CERN-LHCC-2017-020. ATLAS-TDR-029](#) (2017).
- [69] S. Han, H. Mao and W.J. Dally, *Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding*, in *4th International Conference on Learning Representations*, 2016 [[1510.00149](#)].
- [70] S. Han, J. Pool, J. Tran and W.J. Dally, *Learning both weights and connections for efficient neural networks*, in *Advances in Neural Information Processing Systems*, C. Cortes, N. Lawrence, D. Lee, M. Sugiyama and R. Garnett, eds., vol. 28, p. 1135, Curran Associates, Inc., 2015, <https://proceedings.neurips.cc/paper/2015/file/ae0eb3eed39d2bcef4622b2499a05fe6-Paper.pdf> [[1506.02626](#)].
- [71] J. Ngadiuba, V. Loncar, M. Pierini, S. Summers, G. Di Guglielmo, J. Duarte et al., *Compressing deep neural networks on FPGAs to binary and ternary precision with hls4ml*, *Mach. Learn.: Sci. Technol.* **2** (2020) 015001 [[2003.06308](#)].
- [72] C.N. Coelho, A. Kuusela, S. Li, H. Zhuang, T. Aarrestad, V. Loncar et al., *Automatic heterogeneous quantization of deep neural networks for low-latency inference on the edge for particle detectors*, *Nat. Mach. Intell.* **3** (2021) 675 [[2006.10159](#)].
- [73] B. Hawks, J. Duarte, N.J. Fraser, A. Pappalardo, N. Tran and Y. Umuroglu, *Ps and Qs: Quantization-aware pruning for efficient low latency neural network inference*, *Front. AI* **4** (2021) [[2102.11289](#)].
- [74] Y. LeCun, J.S. Denker and S.A. Solla, *Optimal brain damage*, in *Advances in Neural Information Processing Systems*, D.S. Touretzky, ed., vol. 2, p. 598, Morgan-Kaufmann, 1990, <http://papers.nips.cc/paper/250-optimal-brain-damage>.
- [75] Y. Cheng, D. Wang, P. Zhou and T. Zhang, *A survey of model compression and acceleration for deep neural networks*, *IEEE Signal Process. Mag.* **35** (2018) 126 [[1710.09282](#)].
- [76] L. Deng, G. Li, S. Han, L. Shi and Y. Xie, *Model compression and hardware acceleration for neural networks: A comprehensive survey*, *Proc. IEEE* **108** (2020) 485.
- [77] T. Choudhary, V. Mishra, A. Goswami and J. Sarangapani, *A comprehensive survey on model compression and acceleration*, *Artif. Intell. Rev.* **53** (2020) 5113.
- [78] C. Banbury, V.J. Reddi, P. Torelli, J. Holleman, N. Jeffries, C. Kiraly et al., *MLPerf Tiny benchmark*, in *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, vol. 1, 12, 2021, <https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/hash/da4fb5c6e93e74d3df8527599fa62642-Abstract-round1.html> [[2106.07597](#)].

- [79] MLCommons, *Inference: Tiny v0.5 Results*, 2022.
- [80] R. Nane, V. Sima, C. Pilato, J. Choi, B. Fort, A. Canis et al., *A survey and evaluation of FPGA high-level synthesis tools*, *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **35** (2016) 1591.
- [81] M.W. Numan, B.J. Phillips, G.S. Puddy and K. Falkner, *Towards automatic high-level code deployment on reconfigurable platforms: A survey of high-level synthesis tools and toolchains*, *IEEE Access* **8** (2020) 174692.
- [82] R. Kastner, J. Matai and S. Neuendorffer, *Parallel Programming for FPGAs*, hls.ucsd.edu (2018), [1805.03648].
- [83] H. Ye, C. Hao, J. Cheng, H. Jeong, J. Huang, S. Neuendorffer et al., *Scalehls: A new scalable high-level synthesis framework on multi-level intermediate representation*, 2021.
- [84] J. Duarte et al., *Fast inference of deep neural networks in FPGAs for particle physics*, *JINST* **13** (2018) P07027 [1804.06913].
- [85] M. Blott, T. Preusser, N. Fraser, G. Gambardella, K. O'Brien and Y. Umuroglu, *FINN-R: An end-to-end deep-learning framework for fast exploration of quantized neural networks*, *ACM Trans. Reconfigurable Technol. Syst.* **11** (2018) [1809.04570].
- [86] H. Sharma, J. Park, D. Mahajan, E. Amaro, J.K. Kim, C. Shao et al., *From high-level deep neural models to fpgas*, in *49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, p. 1, IEEE, 2016.
- [87] S.I. Venieris and C.-S. Bouganis, *fpgaConvNet: A framework for mapping convolutional neural networks on FPGAs*, in *2016 IEEE 24th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, p. 40, IEEE, 2016, DOI.
- [88] S.I. Venieris and C.-S. Bouganis, *fpgaConvNet: Automated mapping of convolutional neural networks on FPGAs*, in *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, p. 291, ACM, 2017, DOI.
- [89] S.I. Venieris and C.-S. Bouganis, *fpgaConvNet: A toolflow for mapping diverse convolutional neural networks on embedded FPGAs*, in *NIPS 2017 Workshop on Machine Learning on the Phone and other Consumer Devices*, 2017 [1711.08740].
- [90] S.I. Venieris and C.S. Bouganis, *Latency-driven design for FPGA-based convolutional neural networks*, in *2017 27th International Conference on Field Programmable Logic and Applications (FPL)*, p. 1, 2017, DOI.
- [91] Y. Guan, H. Liang, N. Xu, W. Wang, S. Shi, X. Chen et al., *FP-DNN: An automated framework for mapping deep neural networks onto FPGAs with RTL-HLS hybrid templates*, in *2017 IEEE 25th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, p. 152, 2017, DOI.

- [92] Y. Chen, J. He, X. Zhang, C. Hao and D. Chen, *Cloud-dnn: An open framework for mapping dnn models to cloud fpgas*, in *Proceedings of the 2019 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, FPGA '19, (New York, NY, USA), p. 73–82, Association for Computing Machinery, 2019, DOI.
- [93] X. Zhang, J. Wang, C. Zhu, Y. Lin, J. Xiong, W. mei W. Hwu et al., *Dnnbuilder: an automated tool for building high-performance dnn hardware accelerators for fpgas*, *2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)* (2018) 1.
- [94] S. Huang, K. Wu, H. Jeong, C. Wang, D. Chen and W. mei W. Hwu, *Pylog: An algorithm-centric python-based fpga programming and synthesis flow*, *The 2021 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays* (2021) .
- [95] S. Summers et al., *Fast inference of boosted decision trees in FPGAs for particle physics*, *JINST* **15** (2020) P05026 [2002.02534].
- [96] T. Åarrestad et al., *Fast convolutional neural networks on FPGAs with hls4ml*, *Mach. Learn.: Sci. Technol.* **2** (2021) 045015 [2101.05108].
- [97] Y. Iiyama et al., *Distance-weighted graph neural networks on FPGAs for real-time particle reconstruction in high energy physics*, *Front. Big Data* **3** (2021) 44 [2008.03601].
- [98] A. Elabd et al., *Graph Neural Networks for Charged Particle Tracking on FPGAs*, *Front. Big Data* **5** (2022) 828666 [2112.02048].
- [99] E. Govorkova et al., *Autoencoders on FPGAs for real-time, unsupervised new physics detection at 40 MHz at the Large Hadron Collider*, *Nat. Mach. Intell.* (2022) 154 [2108.03986].
- [100] 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].
- [101] D.S. Rankin et al., *FPGAs-as-a-Service Toolkit (FaaSST)*, in *2020 IEEE/ACM International Workshop on Heterogeneous High-performance Reconfigurable Computing (H2RC)*, 10, 2020, DOI [2010.08556].
- [102] Mentor/Siemens, “Catapult High-Level Synthesis.” <https://www.mentor.com/hls-lp/catapult-high-level-synthesis>, 2020.
- [103] G. Di Guglielmo et al., *A Reconfigurable Neural Network ASIC for Detector Front-End Data Compression at the HL-LHC*, *IEEE Trans. Nucl. Sci.* **68** (2021) 2179 [2105.01683].
- [104] S. Habinc, *Functional triple modular redundancy (ftmr). vhdl design methodology for redundancy in combinatorial and sequential logic*, Gaisler Research, *Design and Assessment Report (Version 0.2)* (2002) .

- [105] R.E. Lyons and W. Vanderkulk, *The use of triple-modular redundancy to improve computer reliability*, *IBM J. Res. Dev* **6** (1962) 200.
- [106] Z. Jia, B. Tillman, M. Maggioni and D.P. Scarpazza, *Dissecting the Graphcore IPU Architecture via Microbenchmarking*, 2019. 10.48550/ARXIV.1912.03413.
- [107] GraphCore, “Poplar Graph Framework Software.” <https://www.graphcore.ai/products/poplar>, 2022.
- [108] M. Abadi et al., “TensorFlow: Large-scale machine learning on heterogeneous systems.” 2015.
- [109] GraphCore, “IPU Performance Results.” <https://www.graphcore.ai/performance-results>, 2022.
- [110] Y. Shen, N.C. Harris, S. Skirlo, M. Prabhu, T. Baehr-Jones, M. Hochberg et al., *Deep learning with coherent nanophotonic circuits*, *Nature Photonics* **11** (2017) 441.
- [111] A. Sludds, S. Bandyopadhyay, Z. Chen, Z. Zhong, L. Bernstein, D. Bunandar et al., *Wavelength multiplexed ultralow-power photonic edge computing*, 2022. 10.48550/ARXIV.2203.05466.
- [112] R. Hamerly, L. Bernstein, A. Sludds, M. Soljačić and D. Englund, *Large-scale optical neural networks based on photoelectric multiplication*, *Physical Review X* **9** (2019) .
- [113] Kaz Sato, Cliff Young, David Patterson, “An-in-depth-look-at-googles-first-tensor-processing-unit-tpu.” <https://cloud.google.com/blog/products/ai-machine-learning/an-in-depth-look-at-googles-first-tensor-processing-unit-tpu>, 2017.
- [114] P. Banerjee, R. Friedrich, C. Bash, P. Goldsack, B. Huberman, J. Manley et al., *Everything as a service: Powering the new information economy*, *Computer* **44** (2011) 36.
- [115] F.M. Aymerich, G. Fenu and S. Surcis, *An approach to a cloud computing network*, in *2008 First International Conference on the Applications of Digital Information and Web Technologies (ICADIWT)*, p. 113, 2008.
- [116] K. Bennett, P. Layzell, D. Budgen, P. Brereton, L. Macaulay and M. Munro, *Service-based software: the future for flexible software*, in *Proceedings Seventh Asia-Pacific Software Engineering Conference. APSEC 2000*, p. 214, 2000.
- [117] A. Bocci, D. Dagenhart, V. Innocente, C. Jones, M. Kortelainen, F. Pantaleo et al., *Bringing heterogeneity to the CMS software framework*, *EPJ Web Conf.* **245** (2020) 05009 [2004.04334].
- [118] Nvidia, *Triton Inference Server*, 2022.
- [119] Google, “gRPC.” <https://grpc.io/>, 2022.

- [120] A.M. Caulfield, E.S. Chung, A. Putnam, H. Angepat, J. Fowers, M. Haselman et al., *A cloud-scale acceleration architecture*, in *2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pp. 1–13, 2016, DOI.
- [121] Apache, “Apache Arrow.” <https://arrow.apache.org/docs/index.html>, 2022.
- [122] Graphcore, “Graphcloud.” <https://www.graphcore.ai/graphcloud>, 2022.
- [123] Kubernetes, “Kubernetes.” <https://kubernetes.io/>, 2022.
- [124] A.M. Deiana, N. Tran, J. Agar, M. Blott, G. Di Guglielmo, J. Duarte et al., *Applications and techniques for fast machine learning in science*, *arXiv preprint arXiv:2110.13041* (2021) .
- [125] K. Biery, E. Flumerfelt, A. Lyon, R. Rechenmacher, R. Rivera, M. Rominsky et al., *The fermilab test beam facility data acquisition system based on otdaq*, *arXiv preprint arXiv:1806.07240* (2018) .
- [126] M. Turqueti, R.A. Rivera, A. Prosser, J. Andresen and J. Chramowicz, *Captan: A hardware architecture for integrated data acquisition, control, and analysis for detector development*, in *2008 IEEE Nuclear Science Symposium Conference Record*, pp. 3546–3552, 2008, DOI.
- [127] R. Herbst, R. Claus, M. Freytag, G. Haller, M. Huffer, S. Maldonado et al., *Design of the slac rce platform: A general purpose atca based data acquisition system*, in *2014 IEEE Nuclear Science Symposium and Medical Imaging Conference (NSS/MIC)*, pp. 1–4, 2014, DOI.
- [128] W. Wu, *Felix: the new detector interface for the atlas experiment*, *IEEE Transactions on Nuclear Science* **66** (2019) 986.
- [129] Reflexces, “Achilles Arria 10 SoC SoM board.” <https://www.reflexces.com/modules/arria-10-soc/arria-10-soc-som>, 2022.
- [130] AMD/Xilinx, “Xilinx Alveo.” <https://www.xilinx.com/products/boards-and-kits/alveo.html>, 2022.
- [131] Amazon.com, Inc., “Amazon EC2 F1 Instances.” <https://aws.amazon.com/ec2/instance-types/f1/>, 2022.
- [132] AMD/Xilinx, “Vivado.” <https://www.xilinx.com/support/university/vivado.html>, 2022.
- [133] Intel, “Quartus.” <https://www.intel.com/content/www/us/en/software/programmable/quartus-prime/overview.html>, 2022.
- [134] Cadence, “Cadence.” https://www.cadence.com/en_US/home.html, 2022.
- [135] Synopsys, “Synopsys.” <https://www.synopsys.com/>, 2022.
- [136] Siemens EDA, “Siemens.” <https://eda.sw.siemens.com/en-US/1>, 2022.

- [137] SymbiFlow, “SymbiFlow.” <https://symbiflow.github.io/>, 2022.
- [138] Yosys, “Yosys: Open SYnthesis Suite.” <https://yosyshq.net/yosys/documentation.html>, 2022.
- [139] SkyWater Technology, “SkyWater SKY130 PDK.” <https://skywater-pdk.readthedocs.io/en/main/#>, 2022.
- [140] Google LLC, “Google Cloud Platform.” <https://cloud.google.com/>, 2022.
- [141] Amazon.com, Inc., “Amazon Web Services.” <https://aws.amazon.com/>, 2022.
- [142] Google LLC, “Google Cloud Platform: CPU Platforms.” <https://cloud.google.com/compute/docs/cpu-platforms>, 2022.
- [143] Google LLC, “Google Cloud Platform: GPU Platforms.” <https://cloud.google.com/compute/docs/gpus>, 2022.
- [144] Amazon.com, Inc., “Amazon EC2 Instance Types.” <https://aws.amazon.com/ec2/instance-types/>, 2022.
- [145] B. Holzman, L.A.T. Bauerdick, B. Bockelman, D. Dykstra, I. Fisk, S. Fuess et al., *Hepcloud, a new paradigm for hep facilities: Cms amazon web services investigation*, *Computing and Software for Big Science* **1** (2017) .
- [146] A. Gunny, D. Rankin, J. Krupa, M. Saleem, T. Nguyen, M. Coughlin et al., *Hardware-accelerated Inference for Real-Time Gravitational-Wave Astronomy*, **2108.12430**.
- [147] V. Loncar et al., *Compressing deep neural networks on FPGAs to binary and ternary precision with HLS4ML*, *Mach. Learn. Sci. Tech.* **2** (2021) 015001 [2003.06308].
- [148] T.M. Hong, B. Carlson, B. Eubanks, S. Racz, S. Roche, J. Stelzer et al., *Nanosecond machine learning event classification with boosted decision trees in FPGA for high energy physics*, *JINST* **16** (2021) P08016 [2104.03408].
- [149] R. Rao, *Implementation of Long Short-Term Memory Neural Networks in High-Level Synthesis Targeting FPGAs*, Ph.D. thesis, University of Washington, 2020.
- [150] J. Alimena, Y. Iiyama and J. Kieseler, *Fast convolutional neural networks for identifying long-lived particles in a high-granularity calorimeter*, *JINST* **15** (2020) P12006 [2004.10744].
- [151] N. Nottbeck, C. Schmitt and V. Büscher, *High-performance, deep neural networks with sub-microsecond latency on FPGAs for trigger applications*, *J. Phys. Conf. Ser.* **1525** (2020) 012046.
- [152] ATLAS collaboration, *Machine Learning for Real-Time Processing of ATLAS Liquid Argon Calorimeter Signals with FPGAs*, in *European Physics Society Conference on High Energy Physics*, no. ATL-LARG-PROC-2021-008, 2021, <https://cds.cern.ch/record/2789991>.

- [153] N. Chiedde, *Machine Learning for Real-Time Processing of ATLAS Liquid Argon Calorimeter Signals with FPGAs*, in *Topical Workshop on Electronics for Particle Physics*, 2021 [[2111.08590](#)].
- [154] V. Mikuni, B. Nachman and D. Shih, *Online-compatible unsupervised nonresonant anomaly detection*, *Phys. Rev. D* **105** (2022) 055006 [[2111.06417](#)].
- [155] P. Jawahar, T. Aarrestad, N. Chernyavskaya, M. Pierini, K.A. Wozniak, J. Ngadiuba et al., *Improving Variational Autoencoders for New Physics Detection at the LHC With Normalizing Flows*, *Front. Big Data* **5** (2022) 803685 [[2110.08508](#)].
- [156] A. Butter, S. Diefenbacher, G. Kasieczka, B. Nachman, T. Plehn, D. Shih et al., *Ephemeral Learning - Augmenting Triggers with Online-Trained Normalizing Flows*, [2202.09375](#).
- [157] Y.-j. Jwa, G. Di Guglielmo, L. Arnold, L. Carloni and G. Karagiorgi, *Real-time Inference with 2D Convolutional Neural Networks on Field Programmable Gate Arrays for High-rate Particle Imaging Detectors*, [2201.05638](#).
- [158] R. Ormiston, T. Nguyen, M. Coughlin, R.X. Adhikari and E. Katsavounidis, *Noise reduction in gravitational-wave data via deep learning*, *Physical Review Research* **2** (2020) .
- [159] H. Gabbard, M. Williams, F. Hayes and C. Messenger, *Matching matched filtering with deep networks for gravitational-wave astronomy*, *Physical Review Letters* **120** (2018) .
- [160] H. Gabbard, C. Messenger, I.S. Heng, F. Tonolini and R. Murray-Smith, *Bayesian parameter estimation using conditional variational autoencoders for gravitational-wave astronomy*, *Nature Physics* **18** (2021) 112.