Applications and Techniques for Fast Machine Learning in Science

Editors: Allison McCarn Deiana [1] (coordinator), Nhan Tran [2,3] (coordinator), Joshua Agar [4], Michaela Blott [5] Giuseppe Di Guglielmo [27], Javier Duarte [23], Philip Harris [15], Scott Hauck [24], Mia Liu [25], Mark S. Neubauer [26], Jennifer Ngadiuba [2], Seda Ogrenci-Memik [3], Maurizio Pierini [6]

Report Contributors: Thea Aarrestad [6], Steffen Bähr [9], Jürgen Becker [9], Anne-Sophie Berthold [29], Richard J. Bonventre [11], Tomás E. Müller Bravo [18], Markus Diefenthaler [41], Zhen Dong [19], Nick Fritzsche [29], Amir Gholami [19], Ekaterina Govorkova [6], Kyle J Hazelwood [2], Christian Herwig [2], Babar Khan [21], Sehoon Kim [19], Thomas Klijnsma [2], Yaling Liu [4], Kin Ho Lo [8], Tri Nguyen [15], Gianantonio Pezzullo [10], Seyedramin Rasoulinezhad [22], Ryan A. Rivera [2], Kate Scholberg [13], Justin Selig [32] Sougata Sen [16], Dmitri Strukov [20], William Tang [7], Savannah Thais [7], Kai Lukas Unger [9], Ricardo Vilalta [17], Belina von Krosigk [14,9], Thomas K. Warburton [12]

Community endorsers: Maria Acosta Flechas [2], Anthony Aportela [23], Thomas Calvet [31], Leonardo Cristella 6, Daniel Diaz [23], Caterina Doglioni [37], Maria Domenica Galati 34, Elham E Khoda [24], Farah Fahim [2], Davide Giri [27], Benjamin Hawks [2], Duc Hoang [15], Burt Holzman [2], Shih-Chieh Hsu [24], Sergo Jindariani [2], Iris Johnson [2], Raghav Kansal [23], Ryan Kastner [23], Erik Katsavounidis [15], Jeffrey Krupa [15], Pan Li [25], Sandeep Madireddy [40], Ethan Marx [15], Patrick McCormack [15] Andres Meza [23], Jovan Mitrevski [2], Mohammed Attia Mohammed [36], Farouk Mokhtar [23], Eric Moreno [15], Srishti Nagu [35], Rohin Narayan [1], Noah Palladino [15], Zhiqiang Que [38], Sang Eon Park[15], Subramanian Ramamoorthy [28], Dylan Rankin [15], Simon Rothman [15], Ashish Sharma [30], Sioni Summers [6], Pietro Vischia [33], Jean-Roch Vlimant [39], Olivia Weng [23]

[1] *Southern Methodist University, Dallas, TX 75205, USA,* [2] *Fermi National Accelerator Laboratory, Batavia, IL 60510, USA,* [3] *Northwestern University, Evanston, IL 60208, USA,* [4] *Lehigh University, University, Bethlehem, PA 18015, USA,* [5] *Xilinx Research, Dublin, D24 T683, Ireland,* [6] *European Organization for Nuclear Research (CERN), Meyrin, Switzerland,* [7] *Princeton University, Princeton, NJ 08544, USA,* [8] *University of Florida, Gainesville, FL 32611, USA,* [9] *Karlsruhe Institute of Technology, 76131 Karlsruhe, Germany,* [10] *Yale University, New Haven, CT 06520, USA,* [11] *Lawrence Berkeley National Laboratory, Berkeley, CA 94720, USA,* [12] *Iowa State University, Ames, IA 50011, USA,* [13] *Duke University, Durham, NC 27708, USA,* [14] *Universität Hamburg, 22761 Hamburg, Germany,* [15] *Massachusetts Institute of Technology, Cambridge, MA 02139, USA,* [16] *Birla Institute of Technology and Science, Pilani, Goa 403726, India,* [17] *University of Houston, Houston TX 77204, USA,* [18] *University of Southampton, Southampton SO17 1BJ, United Kingdom,* [19] *University of California Berkeley, Berkeley, CA 94720, USA,* [20] *University of California Santa Barbara, Santa Barbara, CA 93106, USA,* [21] *Technical University Darmstadt, Darmstadt 64289, Germany,* [22] *University of Sydney, Camperdown NSW 2006, Australia,* [23] *University of California San Diego, La Jolla, CA 92093, USA,* [24] *University of Washington, Seattle WA 47907, USA,* [25] *Purdue University, West Lafayette IN 47907, USA,* [26] *University of Illinois Urbana-Champaign, Champaign IL 61820, USA,* [27] *Columbia University, New York, NY 10027, USA,* [28] *University of Edinburgh, Edinburgh EH8 9YL, United Kingdom ,* [29] *Technische Universität Dresden, 01062 Dresden, Germany,* [30] *Indian Institute of Technology Madras, Chennai 600 036, India,* [31] *Centre de Physique des Particules de Marseille, 13009 Marseille, France,* [32] *Cerebras Systems, Sunnyvale CA 94085, USA ,* [33] *Université Catholique de Louvain, B-1348 Louvain-la-Neuve, Belgium,* [34] *University of Groningen, 9747 AG Groningen, Netherlands,* [35] *Lucknow University, Lucknow 226007, U.P., India,* [36] *Center for High Energy Physics (CHEP-FU), Fayoum University, El-Fayoum, Egypt,* [37] *Lund University, SE-223 623 Lund, Sweden,* [38] *Imperial College London, London SW7 2BX, UK,* [39] *California Institute of Technology, Pasadena, CA 91125, USA,* [40] *Argonne National Laboratory, Lemont, IL 60439, USA,* [41] *Thomas Jefferson National Accelerator Facility, Newport News, VA 23606, USA*

Correspondence*:
Allison McCarn Deiana, Nhan Tran
adeiana@smu.edu, ntran@fnal.gov

ABSTRACT

In this community review report, we discuss applications and techniques for *fast* machine learning (ML) in science—the concept of integrating power ML methods into the real-time experimental data processing loop to accelerate scientific discovery. The material for the report builds on two workshops held by the Fast ML for Science community and covers three main areas: applications for fast ML across a number of scientific domains; techniques for training and implementing performant and resource-efficient ML algorithms; and computing architectures, platforms, and technologies for deploying these algorithms. We also present overlapping challenges across the multiple scientific domains where common solutions can be found. This community report is intended to give plenty of examples and inspiration for scientific discovery through integrated and accelerated ML solutions. This is followed by a high-level overview and organization of technical advances, including an abundance of pointers to source material, which can enable these breakthroughs.

Keywords: fast machine learning

Contents

FOREWORD

Machine learning (ML) is making a huge impact on our society and daily lives through advancements in computer vision, natural language processing, and autonomous vehicles, among others. ML is also powering scientific advances which can lead to future paradigm shifts in a broad range of domains, including particle physics, plasma physics, astronomy, neuroscience, chemistry, material science, and biomedical engineering. Scientific discoveries come from groundbreaking ideas and the capability to validate those ideas by testing nature at new scales—finer and more precise temporal and spatial resolution. This is leading to an explosion of data that must be interpreted, and ML is proving a powerful approach. The more efficiently we can test our hypotheses, the faster we can achieve discovery. To fully unleash the power of ML and accelerate discoveries, it is necessary to embed it into our scientific process, into our instruments and detectors.

It is in this spirit that the Fast Machine Learning for Science community[1] has been built. Two workshops have also been organized through this growing community and are the source for this report. The community brings together an extremely wide-ranging group of domain experts who would rarely interact as a whole. One of the underlying benefits of ML is the portability and general applicability of the techniques that can enable experts from seemingly unrelated domains to find a common language. Scientists and engineers from particle physicists to networking experts and biomedical engineers are represented and can interact with experts in fundamental ML techniques and compute systems architects.

This report aims to summarize the progress in the community to understand how our scientific challenges overlap and where there are potential commonalities in data representations, ML approaches, and technology, including hardware and software platforms. Therefore, **the content of the report includes the following: descriptions of a number of different scientific domains including existing work and applications for embedded ML; potential overlaps across scientific domains in data representation or system constraints; and an overview of state-of-the-art techniques for efficient machine learning and compute platforms, both cutting-edge and speculative technologies**.

Necessarily, such a broad scope of topics *cannot* be comprehensive. For the scientific domains, we note that the contributions are *examples* of how ML methods are currently being or planned to be deployed. We hope that giving a glimpse into specific applications will inspire readers to find more novel use-cases and potential overlaps. The summaries of state-of-the-art techniques we provide relate to rapidly developing fields and, as such, may become out of date relatively quickly. The goal is to give non-experts an overview and taxonomy of the different techniques and a starting point for further investigation. To be succinct, we rely heavily on providing references to studies and other overviews while describing most modern methods.

We hope the reader finds this report both instructive and motivational. Feedback and input to this report, and to the larger community, are welcome and appreciated.

Sincerely,
The Editors

[1] fastmachinelearning.org

1 INTRODUCTION

In pursuit of scientific advancement across many domains, experiments are becoming exceedingly sophisticated in order to probe physical systems at increasingly smaller spatial resolutions and shorter timescales. These order of magnitude advancements have lead to explosions in both data volumes and richness leaving domain scientists to develop novel methods to handle growing data processing needs.

Simultaneously, machine learning (ML), or the use of algorithms that can learn directly from data, is leading to rapid advancements across many scientific domains [1]. Recent advancements have demonstrated that deep learning (DL) architectures based on structured deep neural networks are versatile and capable of solving a broad range of complex problems. The proliferation of large datasets like ImageNet [2], computing, and DL software has led to the exploration of many different DL approaches each with their own advantages.

In this review paper, we will focus on the fusion of ML and experimental design to solve critical scientific problems by accelerating and improving data processing and real-time decision-making. We will discuss the myriad of scientific problems that require fast ML, and we will outline unifying themes across these domains that can lead to general solutions. Furthermore, we will review the current technology needed to make ML algorithms run fast, and we will present critical technological problems that, if solved, could lead to major scientific advancements. An important requirement for such advancements in science is the need for openness. It is vital for experts from domains that do not often interact to come together to develop transferable solutions and work together to develop open-source solutions.

Much of the advancements within ML over the past few years have originated from the use of heterogeneous computing hardware. In particular, the use of graphics processing units (GPUs) has enabled the development of large DL algorithms [3–5]. The ability to train large artificial intelligence (AI) algorithms on large datasets has enabled algorithms that are capable of performing sophisticated tasks. In parallel with these developments, new types of DL algorithms have emerged that aim to reduce the number of operations so as to enable fast and efficient AI algorithms.

> Within this review paper, we refer to the concept of ***Fast Machine Learning in Science*** as the integration of ML into the experimental data processing infrastructure to enable and accelerate scientific discovery. Fusing powerful ML techniques with experimental design decreases the "time to science" and can range from embedding real-time feature extraction to be as close as possible to the sensor all the way to large-scale ML acceleration across distributed grid computing datacenters. The overarching theme is to lower the barrier to advanced ML techniques and implementations to make large strides in experimental capabilities across many seemingly different scientific applications. Efficient solutions require collaboration between domain experts, machine learning researchers, and computer architecture designers.

This paper is a review of the second annual Fast Machine Learning conference [6] and will build on the materials presented at this conference. It brings together experts from multiple scientific domains ranging from particle physicists to material scientists to health monitoring researchers with machine learning experts and computer systems architects. Figure 1 illustrates the spirit of the workshop series on which this paper is inspired and the topics covered in subsequent sections.

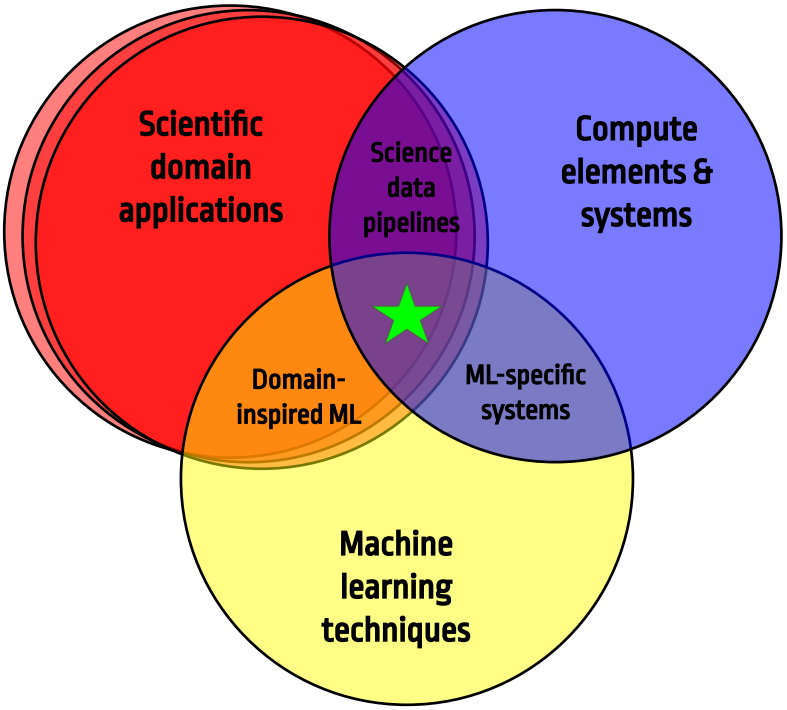

Figure 1. The concept behind this review paper is to find the confluence of domain-specific challenges, machine learning, and experiment and computer system architectures to accelerate science discovery.

As ML tools have become more sophisticated, much of the focus has turned to building very large algorithms that solve complicated problems, such as language translation and voice recognition. However, in the wake of these developments, a broad range of scientific applications have emerged that can benefit greatly from the rapid developments underway. Furthermore, these applications have diversified as people have to come to realize how to adapt their scientific approach so as to take advantage of the benefits originating from the AI revolution. This can include the capability of AI to classify events in real time, such as the identification of a collision of particles or a merger of gravitational waves. It can also include systems control, such as the response control from feedback mechanisms in plasmas and particle accelerators. The latency, bandwidth, and throughput restrictions and the reasons for such restrictions differ within each system. However, in all cases, accelerating ML is a driver in the design goal.

The design of low latency algorithms differs from other AI implementations in that we must tailor specific processing hardware to the task at hand to increase the overall algorithm performance. In particular, certain processor cores have been configured for optimized sparse matrix multiplications. Others have been optimized to maximize the total amount of compute. Processor design, and the design of algorithms around processors, often referred to as hardware AI co-design, is the focus of the work in this review. For example, in some cases, ultra-low latency inference times are needed to perform scientific measurements. One must efficiently design the algorithm to optimally utilize the hardware constraints available while preserving the algorithm performance within desired experimental requirements. This is the essence of hardware AI co-design.

The contents of this review are laid out as follows. In the Section 2, we will explore a broad range of scientific problems where Fast ML can act as a disruptive technology to the status quo and lead to a significant change in how we process data. Domain experts from seemingly different domains are examined.

In Section 3, we describe data representations and experimental platform choices are common to many types of experiments. We will connect how Fast ML solutions can be generalized to low latency, highly resource-efficient, and domain-specific deep learning inference for many scientific applications. Finally in Section 4, to achieve this requires optimized hardware-ML co-design from the algorithm design to the system architecture. We provide an overview of state-of-the-art techniques to train neural networks optimized for both performance and speed, survey various compute architectures to meet the needs of the experimental design and outline software solutions that optimize and enable the hardware deployment.

The goal of this paper is to bring together scientific opportunities, common solutions, and state-of-the-art technology into one single narrative. We hope this can contribute to accelerating the deployment of potentially transformative ML solutions to a broad range of scientific fields going forward.

2 EXEMPLARS OF DOMAIN APPLICATIONS

As scientific ecosystems grow rapidly in their speed and scale, new paradigms for data processing and reduction need to be integrated into system-level design. In this section, we explore requirements for accelerated and sophisticated data processing. Implementations of fast machine learning can appear greatly varied across domains and architectures but yet can have similar underlying data representations and needs for integrating machine learning. We enumerate here a broad sampling of scientific domains across seemingly unrelated tasks including their existing techniques and future needs. This will then lead to the next section where we discuss overlaps and common tasks.

In this section, we first have a detailed description of examples of Fast ML techniques being deployed at experiments for the Large Hadron Collider. Much rapid development has occurred for these experiments recently and gives an exemplar for how broad advancements can be made across various aspects of a specific domain. Then the following subsections will be briefer but lay out key challenges and areas of existing and potential applications of Fast ML across a number of other scientific domains.

2.1 Large Hadron Collider

The Large Hadron Collider (LHC) at CERN is the world's largest and highest-energy particle accelerator, where collisions between bunches of protons occur every 25 ns. To study the products of these collisions, several detectors are located along the ring at interaction points. The aim of these detectors is to measure the properties of the Higgs boson [7, 8] with high precision and to search for new physics phenomena beyond the standard model of particle physics. Due to the extremely high frequency of 40 MHz at which proton bunches collide, the high multiplicity of secondary particles, and the large number of sensors, the detectors have to process and store data at enormous rates. For the two multipurpose experiments, CMS [9] and ATLAS [9], comprised of tens of millions of readout channels, these rates are of the order of 100 Tb/s. Processing and storing this data presents severe challenges that are among the most critical for the execution of the LHC physics program.

The approach implemented by the detectors for data processing consists of an online processing stage, where the event is selected from a buffer and analyzed in real time, and an offline processing stage, in which data have been written to disk and are more thoroughly analyzed with sophisticated algorithms. The online processing system, called the *trigger*, reduces the data rate to a manageable level of 10 Gb/s to be recorded for offline processing. The trigger is typically divided into multiple tiers. Due to the limited size of the on-detector buffers, the first tier (Level-1 or L1) utilizes FPGAs and ASICs capable of executing the filtering process with a maximum latency of $\mathcal{O}(1)$ μs. At the second stage, the high-level trigger (HLT), data are processed on a CPU-based computing farm located at the experimental site with a latency of up to 100 ms. Finally, the complete offline event processing is performed on a globally distributed CPU-based computing grid.

Maintaining the capabilities of this system will become even more challenging in the near future. In 2027, the LHC will be upgraded to the so-called High-Luminosity LHC (HL-LHC) where each collision will produce 5–7 times more particles, ultimately resulting in a total amount of accumulated data that will be one order of magnitude higher than achieved with the present accelerator. At the same time, the particle detectors will be made larger, more granular, and capable of processing data at ever-increasing rates. Therefore, the physics that can be extracted from the experiments will be limited by the accuracy of algorithms and computational resources.

Machine learning technologies offer promising solutions and enhanced capabilities in both of these areas, thanks to their capacity for extracting the most relevant information from high-dimensional data and to their highly parallelizable implementation on suitable hardware. It is expected that a new generation of algorithms, if deployed at all stages of data-processing systems at the LHC experiments, will play a crucial part in maintaining, and hopefully improving, the physics performance. In the following sections, a few examples of the application of machine learning models to physics tasks at the LHC are reviewed, together with novel methods for their efficient deployment in both the real-time and offline data processing stages.

2.1.1 Event reconstruction

The reconstruction of proton-proton collision events in the LHC detectors involves challenging pattern recognition tasks, given the large number ($\mathcal{O}(1000)$) of secondary particles produced and the high detector granularity. Specialized detector sub-systems and algorithms are used to reconstruct the different types and properties of particles produced in collisions. For example, the trajectories of charged particles are reconstructed from space point measurements in the inner silicon detectors, and the showers arising from particles traversing the calorimeters are reconstructed from clusters of activated sensors.

Traditional algorithms are highly tuned for physics performance in the current LHC collision environment, but are inherently sequential and scale poorly to the expected HL-LHC conditions. It is thus necessary to revisit existing reconstruction algorithms and ensure that both the physics and computational performance will be sufficient. Deep learning solutions are currently being explored for pattern recognition tasks, as a significant speedup can be achieved when harnessing heterogeneous computing and parallelizable and efficient ML that exploits AI-dedicated hardware. In particular, 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. Much of the following work has been conducted using the TrackML dataset [10], which simulates a generalized detector under HL-LHC-like pileup conditions. Quantifying the performance of these GNNs in actual experimental data is an ongoing point of study.

For reconstructing showers in calorimeters, GNNs have been found to predict the properties of the original incident particle with high accuracy starting from individual energy deposits. The work in [11] proposes a graph formulation of pooling to dynamically learn the most important relationships between data via an intermediate clustering, and therefore removing the need for a predetermined graph structure. When applied to the CMS electromagnetic calorimeter, with single detector hits as inputs to predict the energy of the original incident particle, a 10% improvement is found over the traditional boosted decision tree (BDT) based approach.

GNNs have been explored for a similar calorimeter reconstruction task for the high-granularity calorimeters that will replace the current design for HL-LHC. The task will become even more challenging as such detectors will feature irregular sensor structure and shape (e.g. hexagonal sensor cells for CMS [12]), high occupancy, and an unprecedented number of sensors. For this application, architectures such as EdgeConv [13] and GravNet/GarNet [14] have shown promising performance in the determination of the properties of single showers, yielding excellent energy resolution and high noise rejection [15]. While these preliminary studies were focused on scenarios with low particle multiplicities, the scalability of the clustering performance to more realistic collision scenarios is still a subject of active development.

GNNs have also been extensively studied for charged particle tracking (the task of identifying and reconstructing the trajectories of individual particles in the detector) [16–19]. The first approaches to this

problem typically utilized edge-classification GNNs in a three-step process: graphs are constructed by algorithmically constructing edges between tracker hits in a point cloud, the graphs are processed through a GNN to predict edge weights (true edges that are part of true particle trajectories should be highly weighted and false edges should be lowly rated), and finally, the selected edges are grouped together to generate high-weight sub-graphs which form full track candidates, as shown in Figure 2.

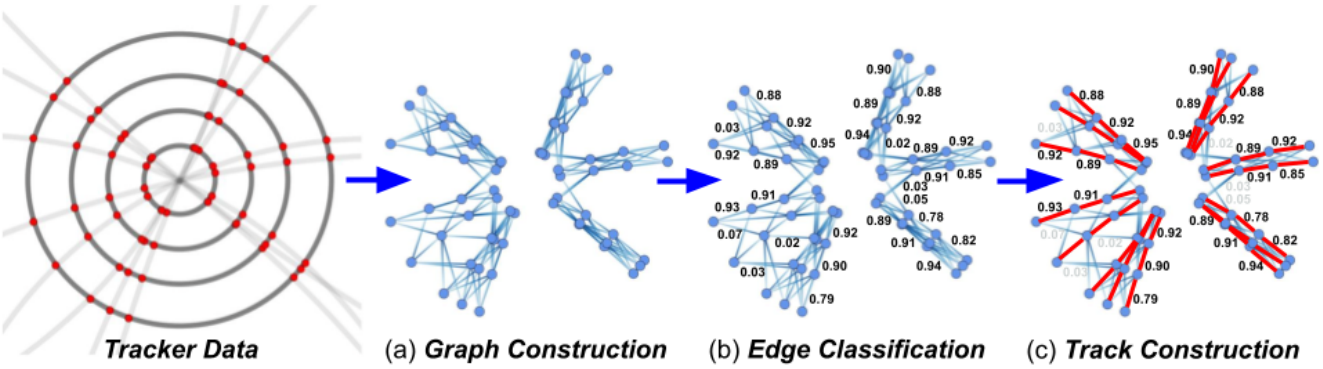

Figure 2. High-level overview of the stages in a GNN-based tracking pipeline. Only a subset of the typical edge weights are shown for illustration purposes.

There have been several studies building upon and optimizing this initial framework. The ExaTrkX collaboration has demonstrated performance improvements by incorporating a recurrent GNN structure [16] and re-embedding graphs prior to training the GNNs [20]. Other work has shown that using an Interaction Network architecture [21] can substantially reduce the number of learnable parameters in the GNN [22]; the authors also provide comprehensive comparisons between different graph construction and track building algorithms. Recent work has also explored alternate approaches that combine graph building, GNN inference, and track construction into a single algorithm that is trainable end-to-end; in particular, instance segmentation architectures have generated promising results [23].

Finally, a novel approach based on GNNs [24] has been proposed as an alternative solution to the so-called particle-flow algorithm that is used by LHC experiments to optimally reconstruct each individual particle produced in a collision by combining information from the calorimeters and the tracking detectors [25]. The new GNN algorithm is found to offer comparable performance for charged and neutral hadrons to the existing reconstruction algorithm. At the same time, the inference time is found to scale approximately linearly with the particle multiplicity, which is promising for its ability to maintain computing costs within budget for the HL-LHC. Further improvements to this original approach are currently under study, including an event-based loss, such as the object condensation approach. Second, a complete assessment of the physics performance remains to be evaluated, including reconstruction of rare particles and other corners of the phase space. Finally, it remains to be understood how to optimize and coherently interface this with the ML-based approach proposed for tasks downstream and upstream in the particle-level reconstruction.

2.1.2 Event simulation

The extraction of results from LHC data relies on a detailed and precise simulation of the physics of proton-proton collisions and of the response of the detector. In fact, the collected data are typically compared to a reference model, representing the current knowledge, in order to either confirm or disprove it. Numerical models, based on Monte Carlo (MC) methods, are used to simulate the interaction between

elementary particles and matter, while the Geant4 toolkit is employed to simulate the detectors. These simulations are generally very CPU intensive and require roughly half of the experiment's computing resources, with this fraction expected to increase significantly for the HL-LHC.

Novel computational methods based on ML are being explored so as to perform precise modeling from particle interactions to detector readouts and response while maintaining feasible computing budgets for HL-LHC. In particular, numerous works have focused on the usage of generative adversarial networks or other state-of-the-art generative models to replace computationally intensive fragments of MC simulation, such as modeling of electromagnetic showers [26–28], reconstruction of jet images [29] or matrix element calculations [30]. In addition, the usage of ML generative models on end-to-end analysis-specific fast simulations have also been investigated in the context of Drell-Yan [31], dijet [32] and W+jets [33] production. These case-by-case proposals serve as proof-of-principle examples for complementary data augmentation strategy for LHC experiments.

2.1.3 Heterogeneous computing

State-of-the-art deep learning models are being explored for the compute-intensive reconstruction of each collision event at the LHC. However, their efficient deployment within the experiments' computing paradigms is still a challenge, despite the potential speed-up when the inference is executed on suitable AI-dedicated hardware. In order to gain from a parallelizable ML-based translation of traditional and mostly sequential algorithms, a heterogeneous computing architecture needs to be implemented in the experiment infrastructure. For this reason, comprehensive exploration of the use of CPU+GPU [34] and CPU+FPGA [35, 36] heterogeneous architectures was made to achieve the desired acceleration of deep learning inference within the data processing workflow of LHC experiments. These works demonstrated that the acceleration of machine learning inference "as a service" represents a heterogeneous computing solution for LHC experiments that potentially requires minimal modification to the current computing model.

In this approach, the ML algorithms are transferred to a co-processor on an independent (local or remote) server by reconfiguring the CPU node to communicate with it through asynchronous and non-blocking inference requests. With the inference task offloaded on demand to the server, the CPU can be dedicated to performing other necessary tasks within the event. As one server can serve many CPUs, this approach has the advantage of increasing the hardware cost-effectiveness to achieve the same throughput when comparing it to a direct-connection paradigm. It also facilitates the integration and scalability of different types of co-processor devices, where the best one is chosen for each task.

Finally, existing open-source frameworks that have been optimized for fast DL on several different types of hardware can be exploited for a quick adaptation to LHC computing. In particular, one could use the Nvidia Triton Inference Server within a custom framework, so-called Services for Optimized Network Inference on Co-processors (SONIC), to enable remote gRPC calls to either GPUs or FPGAs within the experimental software, which then only has to handle the input and output conversion between event data format and inference server format. The integration of this approach within the CMS reconstruction software has been shown to lead to a significant overall reduction in the computing demands both at the HLT and offline.

2.1.4 Real-time analysis at 40 MHz

Bringing deep learning algorithms to the Level-1 hardware trigger is an extremely challenging task due to the strict latency requirement and the resource constraints imposed by the system. Depending

on which part of the system an algorithm is designed to run on, a latency down to $\mathcal{O}(10)$ ns might be required. With $\mathcal{O}(100)$ processors running large-capacity FPGAs, processing thousands of algorithms in parallel, dedicated FPGA-implementations are needed to make ML algorithms as resource-efficient and fast as possible. To facilitate the design process and subsequent deployment of highly parallel, highly compressed ML algorithms on FPGAs, dedicated open-source libraries have been developed: `hls4ml` and `Conifer`. The former, `hls4ml`, provides conversion tools for deep neural networks, while `Conifer` aids the deployment of Boosted Decision Trees (BDTs) on FPGAs. Both libraries, as well as example LHC appli

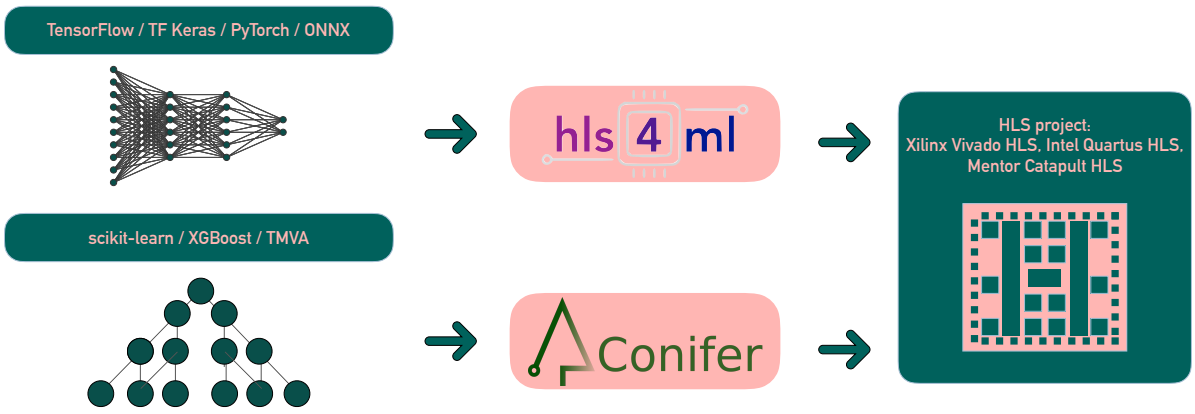

Figure 3. Two dedicated libraries for the conversion of Machine Learning algorithms into FPGA or ASIC firmware: `hls4ml` for deep neural network architectures and `Conifer` for Boosted Decision Tree architectures. Models from a wide range of open-source ML libraries are supported and may be converted using three different high-level synthesis backends.

The `hls4ml` library [37–40] converts pre-trained ML models into ultra low-latency FPGA or ASIC firmware with little overhead required. Integration with the Google QKeras library [41] allows users to design aggressively quantized deep neural networks and train them quantization-aware [40] down to 1 or 2 bits for weights and activations [39]. This step results in highly resource-efficient equivalents of the original model, sacrificing little to no accuracy in the process. The goal of this joint package is to provide a simple two-step approach going from a pre-trained floating point model to FPGA firmware. The `hls4ml` library currently provides support for several commonly used neural network layers like fully connected, convolutional, batch normalization, pooling, as well as several activation functions. These implementations are already sufficient to provide support for the most common architectures envisioned for deployment at L1.

Some first examples of machine learning models designed for the L1 trigger are based on fully connected layers, and they are proposed for tasks such as the reconstruction and calibration of final objects or lower-level inputs like trajectories, vertices, and calorimeter clusters [42]. One example of a convolutional NN (CNN) architecture targeting the L1 trigger is a dedicated algorithm for the identification of long-lived particles [43]. Here, an attempt is made to efficiently identify showers from displaced particles in a high-granularity forward calorimeter. The algorithm is demonstrated to be highly efficient down to low energies while operating at a low trigger rate. Traditionally, cut-based selection algorithms have been used for these purposes, in order to meet the limited latency- and resource budget. However, with the advent

of tools like `hls4ml` and QKeras, ML alternatives are being explored to improve the sensitivity to such physics processes while maintaining latency and resources in the available budget.

More recently, (variational) auto-encoders (VAEs or AEs) are being considered for the detection of "anomalous" collision events, i.e. events that are not produced by standard physics processes but that could be due instead to unexpected processes not yet explored at colliders. Such algorithms have been proposed for both the incoming LHC run starting in 2022 as well as for the future high-luminosity runs where more granular information will be available. The common approach uses global information about the event, including a subset of individual produced particles or final objects such as jets as well as energy sums. The algorithm trained on these inputs is then used to classify the event as anomalous if surpassing a threshold on the degree of anomaly (typically the loss function), ultimately decided upon the available bandwidth. Deploying a typical variational autoencoder is impossible in the L1-trigger since the bottleneck layer involves Gaussian random sampling. The explored solution is therefore to only deploy the encoder part of the network and do inference directly from the latent dimension. Another possibility is to deploy a simple auto-encoder with the same architecture and do inference computing the difference between output and input. However, this would require buffering a copy of the input for the duration it takes the auto-encoder to process the input. For this reason, the two methods are being considered and compared in terms of accuracy over a range of new physics processes, as well as latency and resources.

Finally, another interesting aspect of the `hls4ml` tool is the capability for users to easily add custom layers that might serve a specific task not captured by the most common layers supported in the library. One example of this is compressed distance-weighted graph networks [44], where a graph network block called a *GarNet layer* takes as input a set of V vertices, each of which has F_{in} features, and returns the same set of vertices with F_{out} features. To keep the dimensionality of the problem at a manageable level, the input features of each vertex are encoded and aggregated at S aggregators. Message-passing is only performed between vertices and a limited set of aggregators, and not between all vertices, significantly reducing the network size. In Ref. [44], an example task of pion and electron identification and energy regression in a 3D calorimeter is studied. A total inference latency of $\mathcal{O}(100)$ ns is reported, satisfying the L1 requirement of $\mathcal{O}(1)$ μs latency. The critical resource is digital signal processing (DSP) units, where 29% of the DSPs are in use by the algorithm. This can be further reduced by taking advantage of quantization-aware training with QKeras. Another example of a GNN architecture implemented on FPGA hardware using `hls4ml` is presented in Ref. [45]. This work shows that a compressed GNN can be deployed on FPGA hardware within the latency and resources required by L1 trigger system for the challenging task of reconstructing the trajectory of charged particles.

In many cases, the task to be performed is simple enough that a boosted decision tree (BDT) architecture suffices to solve the problem. As of today, BDTs are still the most commonly used ML algorithm for LHC experiments. To simplify the deployment of these, the library `Conifer` [46] has been developed. In `Conifer`, the BDT implementation targets extreme low latency inference by executing all trees, and all decisions within each tree, in parallel. BDTs and random forests can be converted from scikit-learn [47], XGBoost [48], and TMVA [49], with support for more BDT training libraries planned.

There are several ongoing projects at LHC which plan to deploy BDTs in the Level-1 trigger using `Conifer`. One example is a BDT designed to provide an estimate of the *track quality*, by learning to identify tracks that are reconstructed in error, and do not originate from a real particle [50]. While the accuracy and resource usage are similar between a BDT and a DNN, the latency is significantly reduced

for a BDT architecture. The algorithm is planned to be implemented in the CMS Experiment for the data-taking period beginning in 2022.

Rather than relying on open source libraries such as `hls4ml` or `Conifer`, which are based on high-level synthesis tools from FPGA vendors, other approaches are being considered based directly on hardware description languages, such as VHDL [51, 52]. One example is the application of ML for the real-time signal processing of the ATLAS Liquid Argon calorimeter [53]. It has been shown that with upgraded capabilities for the HL-LHC collision environment the conventional signal processing, which applies an optimal filtering algorithm [54], will lose its performance due to the increase of overlapping signals. More sophisticated DL methods have been found to be more suitable to cope with these challenges being able to maintain high signal detection efficiency and energy reconstruction. More specifically, studies based on simulation [55] of dilated convolutional neural networks showed promising results. An implementation of this architecture for FPGA is designed using VHDL [52] to meet the strict requirements on latency and resources required by the L1 trigger system. The firmware runs with a multiple of the bunch crossing frequency to reuse hardware resources by implementing time-division multiplexing while using pipeline stages, the maximum frequency can be increased. Furthermore, DSPs are chained up to perform the MAC operation in between two layers efficiently. In this way, a core frequency of more than 480 MHz could be reached, corresponding to twelve times the bunch crossing frequency.

2.1.5 Bringing ML to detector front-end

While LHC detectors grow in complexity to meet the challenging conditions of higher-luminosity environments, growing data rates prohibit transmission of full event images off-detector for analysis by conventional FPGA-based trigger systems. As a consequence, event data must be compressed on-detector in low-power, radiation-hard ASICs while sacrificing minimal physics information.

Traditionally this has been accomplished by simple algorithms, such as grouping nearby sensors together so that only these summed "super-cells" are transmitted, sacrificing the fine segmentation of the detector. Recently, an autoencoder-based approach has been proposed, relying instead on a set of machine-learned radiation patterns to more efficiently encode the complete calorimeter image via a CNN. Targeting the CMS high-granularity endcap calorimeter (HGCal) [12] at the HL-LHC, the algorithm aims to achieve higher-fidelity electromagnetic and hadronic showers, critical for accurate particle identification.

The on-detector environment (the ECON-T concentrator ASIC [12]) demands a highly-efficient CNN implementation; a compact design should be thoroughly optimized for limited-precision calculations via quantization-aware training tools [56]. Further, to automate the design, optimization, and validation of the complex NN circuit, HLS-based tool flows [37] may be adapted to target the ASIC form factor. Finally, as the front-end ASIC cannot be completely reprogrammed in the manner of an FPGA, a mature NN design is required from the time of initial fabrication. However, adaptability to changing run conditions and experimental priorities over the lifetime of the experiment motivate the implementation of all NN weights as configurable registers accessible via the chip's slow-control interface.

2.2 High intensity accelerator experiments

2.2.1 ML-based Trigger System at the Belle II Experiment

Context:

The Belle II experiment in Japan is engaged in the search for physics phenomena that cannot be explained by the Standard Model. Electrons and positrons are accelerated at the SuperKEKB particle accelerator to collide at the interaction point located inside of the Belle II detector. The resulting decay products are

continually measured by the detector's heterogeneous sensor composition. The resulting data is then stored offline for detailed analysis.

Challenges:

Due to the increasing luminosity (target luminosity is $8 \times 10^{35}\mathrm{cm}^{-2}\mathrm{s}^{-1}$) most of the recorded data is from unwanted but unavoidable background reactions, rather than electron-positron annihilation at the interaction point. Not only is storing all the data inefficient due to the high background rates, but it is also not feasible to build an infrastructure that stores all the generated data. A multilevel trigger system is used as a solution to decide online which recorded events are to be stored.

Existing and Planned Work:

The Neural Network z-Vertex Trigger (NNT) described used at Belle II is a deadtime-free level 1 (L1) trigger that identifies particles by estimating their origin along the beampipe. For the whole L1 trigger process, from data readout to the decision, a real-time 5 μs time budget is given to avoid dead-time [57]. Due to the time cost of data pre-processing and transmission, the NNT needs to provide a decision within 300 ns processing time.

The task of the NNT is to estimate the origin of a particle track so that it can be decided whether it originates from the interaction point or not. For this purpose, a multilayer perceptron (MLP) implemented on a Xilinx Virtex 6 XC6VHX380T FPGA is used. The MLP consists of three layers with 27 input neurons, 81 hidden layer neurons and two output neurons. Data from the Belle II's central drift chamber (CDC) is used for this task, since it is dedicated to the detection of particle tracks. Before being processed by the network, the raw detector data is first combined into a 2D track based on so-called track segments, which are groupings of adjacent active sense wires. The output of the NNT delivers the origin of the track in z, along the beampipe, as well as the polar angle θ. With the help of the z-vertex, the downstream global decision logic (GDL) can decide whether a track is from the interaction point or not. In addition, the particle momentum can be detected using the polar angle θ [58].

The networks used in the NNT are trained offline. The first networks were trained with plain simulated data because no experimental data were available. For more recent networks, reconstructed tracks from the experimental data are used. For the training the iRPROP algorithm is used which is an extension of the RPROP backpropagation algorithm. Current results show a good correlation between the NNT tracks and reconstructed tracks. Since the event rate and the background noise are currently still tolerable, the z-cut, i.e., the allowed estimated origin of a track origin in order to be kept, is chosen at ± 40 cm. With increasing luminosity and the associated increasing background, this z-cut can be tightened. Since the new Virtex Ultrascale based universal trigger board (UT4) is available for the NNT this year, an extension of the data preprocessing is planned. This will be done by a 3D Hough transformation for further efficiency increases. It has already been shown in simulation that a more accurate resolution and larger solid angle coverage can be achieved [59].

2.2.2 Mu2e

Context:

The Mu2e experiment at Fermilab will search for the charged lepton flavor violating process of neutrinoless $\mu \rightarrow e$ coherent conversion in the field of an aluminum nucleus. About $7 \cdot 10^{17}$ muons, provided by a dedicated muon beamline in construction at Fermilab, will be stopped in 3 years in the aluminum target. The corresponding single event sensitivity will be $2.5 \cdot 10^{-17}$. To detect the signal e^- ($p = 105$ MeV), Mu2e uses a detector system made of a straw-tube tracker and a crystal electromagnetic calorimeter [60].

Challenges:

The trigger system is based on detector Read Out Controllers (ROCs) which stream out continuously the data, zero-suppressed, to the Data Transfer Controller units (DTCs). The proton pulses are delivered at a rate of about 600 kHz and a duty cycle of about 30% (0.4 s out of 1.4 s of the booster-ring delivery period). Each proton pulse is considered a single event, with the data from each event then grouped at a single server using a 10 Gbps Ethernet switch. Then, the online reconstruction of the events starts and makes a trigger decision. The trigger system needs to satisfy the following requirements: (1) provide efficiency better than 90% for the signals; (2) keep the trigger rate below a few kHz – equivalent to 7 Pb/year; (3) achieve a processing time < 5 ms/event. Our main physics triggers use the information of the reconstructed tracks to make the final decision.

Existing and Planned Work:

The current strategy is to perform the helix pattern recognition and the track reconstruction with the CPUs of the DAQ servers, but so far this design showed limitations in matching the required timing performance [61]. Another idea that the collaboration started exploring is to perform the early stage of the track reconstruction on the ROC and DTC FPGA using the High Level Synthesis tool (HLS) and the `hls4ml` package. The Mu2e helix pattern-recognition algorithms [61] are a natural fit for these tools for several reasons: they use neural-networks to clean up the recorded straw-hits from hits by low-momentum electrons ($p < 10\,\text{MeV}$) and they perform large combinatorics calculations when reconstructing the helicoidal electron trajectory. This R&D is particularly important for the design of the trigger system of the planned upgrade of Mu2e [62], where we expect to: (i) increase the beam intensity by at least a factor of 10, (ii) increase the duty cycle to at least 90%, and (iii) increase the number of detector's channels to cope with the increased occupancy.

2.3 Materials Discovery

2.3.1 Materials Synthesis

Context:

Advances in electronics, transportation, healthcare, and buildings require the synthesis of materials with controlled synthesis-structure-property relationships. To achieve application-specific performance metrics, it is common to design and engineer materials with highly ordered structures. This directive has led to a boom in non-equilibrium materials synthesis techniques. Most exciting are additive synthesis and manufacturing techniques, for example, 3d-printing[63–67] and thin film deposition[68–74], where complex nanoscale architectures of materials can be fabricated. To glean insight into synthesis dynamics, there has been a trend to include in situ diagnostics to observe synthesis dynamics[75–78]. There is less emphasis on automating the downstream analysis to turn data into actionable information that can detect anomalies in synthesis, guide experimentation, or enable closed-loop control. Part of the challenge with automating analysis pipelines for in situ diagnostics is the highly variable nature and multimodality of the measurements and the sensors. A system might measure many time-resolved state variables (time-series) at various locations (e.g., temperature, pressure, energy, flow rate, etc.)[79]. Additionally, it is common to measure time-resolved spectroscopic signals (spectrograms) that provide, for instance, information about the dynamics of the chemistry and energetic distributions of the materials being synthesized[80–83]. Furthermore, there are a growing number of techniques that leverage high-speed temporally-resolved imaging to observe synthesis dynamics[84, 85].

Challenges:

Experimental synthesis tools and in situ diagnostic instrumentation are generally semi-custom instruments provided by commercial vendors. Many of these vendors rely on proprietary software to differentiate their products from their competition. In turn, the closed-nature of these tools and even data schemas makes it hard to utilize these tools fully. The varied nature and suppliers for sensors compounds this challenge. Integration and synchronization of multiple sensing modalities require a custom software solution. However, there is a catch-22 because the software does not yet exist. Researchers cannot be ensured that the development of analysis pipelines will contribute to their ultimate goal to discover new materials or synthesize materials with increased fecundity. Furthermore, there are significant workforce challenges as most curriculums emphasize Edisonian rather than computational methods in the design of synthesis. There is an urgent need for multilingual trainees fluent in typically disparate fields.

Existing and Planned Work:

Recently, the materials science community has started to embrace machine learning to accelerate scientific discovery[86–88]. However, there have been growing pains. The ability to create highly overparameterized models to solve problems with limited data provides a false sense of efficacy without the generalization required for science. Machine learning model architectures designed for natural time-series and images are ill-posed for physical processes governed by equations. In this regard, there is a growing body of work to embed physics in machine learning models, which serve as the ultimate regularizers. For instance, rotational [89, 90] and Euclidean equivariance [91, 92] has been built into the model architectures, and methods to learn sparse representations of underlying governing equations have been developed[93–95].

Another challenge is that real systems have system-specific discrepancies that need to be compensated[96]. For example, a precursor from a different batch might have a slightly different viscosity that needs to be considered. There is an urgent need to develop these foundational methods for materials synthesis. Complementing these foundational studies, there has been a growing body of literature emphasizing post-mortem machine-learning-based analysis of in situ spectroscopies[97, 98]. As these concepts become more mature, there will be an increasing emphasis on codesign of synthesis systems, machine learning methods, and hardware for on-the-fly analysis and control. This effort towards self-driving laboratories is already underway in wet-chemical synthesis where there are minimal dynamics, and thus, latencies are not a factor[99, 100]. Future efforts will undoubtedly focus on controlling dynamic synthesis processes where millisecond-to-nanosecond latencies are required.

2.3.2 Scanning Probe Microscopy

Context:

Touch is the first sense humans develop. Since the atomic force microscope's (AFM) invention in 1985[101], humans have been able to "feel" surfaces with atomic level resolution with pN sensitivity. AFMs rely on bringing an atomically sharp tip mounted on a cantilever into contact with a surface. By scanning this tip nanometer-to-atomically resolved images can be constructed by measuring the angular deflection of a laser bounced off the cantilever. This detection mechanism provides high-precision sub-angstrom measures of displacement.

By adding functionality to the probe (e.g., electrical conductivity[102], resistive heaters[103], single-molecule probes[104], and N-V centers[105]), scanning probe microscopy (SPM) can measure nanoscale functional properties, including electrical conductivity[106, 107], piezoresponse[108], electrochemical response[109], magnetic force[110], magnetometry[111], and much more. These techniques have been

expanded to include dynamics measurements during a tip-induced perturbation that drives a structural transformation. These methods have led to a boom in new AFM techniques, including fast-force microscopy[112], current-voltage spectroscopies[113], band-excitation-based spectroscopies[114], and full-acquisition mode spectroscopies[115]. What has emerged is a data deluge where these techniques are either underutilized or under-analyzed.

Challenges:

The key practical challenge is that it takes on days-to-weeks to analyze data from a single measurement properly. As a result, experimentalists have little information on how to design their experiments. There is even minimal feedback on whether the experiments have artifacts (e.g., tip damage) that would render the results unusable. The number of costly failed experiments is a strong deterrent to conducting advanced scanning probe spectroscopies and developing even more sophisticated imaging techniques. There is a significant challenge in both the acceleration and automation of analysis pipelines.

Existing and Planned Work:

In materials science, scanning probe microscopy has quickly adopted machine learning. Techniques for linear and nonlinear spectral unmixing provide rapid visualization and extraction of information from these datasets to discover and unravel physical mechanisms [116–119]. The ease of applying these techniques has led to justified concerns about the overinterpretation of results and overextension of linear models [120] to highly nonlinear systems. More recently, long-short term memory autoencoders were controlled to have non-negative and sparse latent spaces for spectral unmixing. By traversing the learned latent space, it has been possible to draw complex structure-property relationships [121, 113]. There are significant opportunities to accelerate the computational pipeline such that information can be extracted on practically relevant time scales by the experimentalist on the microscope.

Due to the high velocity of data, up to GB/s, with sample rates of 100,000 spectra, extracting even cursory information will require the confluence of data-driven models, physics-informed machine learning, and AI hardware. As a tangible example, in band-excitation piezoresponse force microscopy, the frequency-dependent cantilever response is measured at rates up to 2,000 spectra-per-second. Extracting the parameters from these measurements requires fitting the response to an empirical model. Using least-squares fitting throughput is limited to ~ 50-fits/core-minute, but neural networks provide an opportunity to accelerate analysis and better handle noisy data [122]. There is an opportunity to deploy neural networks on GPU or FPGA hardware accelerators to approximate and accelerate this pipeline by orders of magnitude.

2.4 Fermilab Accelerator Controls

Context:

The Fermi National Accelerator Laboratory (Fermilab) is dedicated to investigating matter, energy, space, and time [123]. For over 50 years, Fermilab's primary tool for probing the most elementary nature of matter has been its vast accelerator complex. Spanning a number of miles of tunnels, the accelerator complex is actually multiple accelerators and beam transport lines each representing different accelerator techniques and eras of accelerator technologies. In its long history, Fermilab's accelerator complex has had to adapt to the mission, asking more of the accelerators than they were designed for and often for purposes they were never intended. This often resulted in layering new controls on top of existing antiquated hardware. Until recently, accelerator controls focused mainly on providing tools and data to the machine operators and experts for tuning and optimization. Having recognized the future inadequacies of the current control system and the promise of new technologies such as ML, the Fermilab accelerator control system will be

largely overhauled in the coming years as part of the Accelerator Controls Operations Research Network (ACORN) project [124].

Challenges:

The accelerator complex brings unique challenges for machine learning. Particle accelerators are immensely complicated machines, each consisting of many thousands of variable components and even larger data sources. Their large size and differing types, resolution, and frequency of data mean collecting and synchronizing data is difficult. Also, as one might imagine, control and regulation of beams that travel at near light speeds is always a challenge. Maintaining and upgrading the accelerator complex controls is costly. For this reason, much of the accelerator complex is a mixture of obsolete, new and cutting edge hardware.

Existing and Planned Work:

Traditional accelerator controls have focused on grouping like elements so that particular aspects of the beam can be tuned independently. However, many elements are not always completely separable. Magnets, for example, often have higher-order fields that affect the beam in different ways than is the primary intent. Machine learning has made it finally possible to combine previously believed to be unrelated readings and beam control elements into new novel control and regulation schemes.

One such novel regulation project is underway for the Booster Gradient Magnet Power Supply (GMPS). GMPS controls the primary trajectory of the beam in the Booster [125]. The project hopes to increase the regulation precision of GMPS ten-fold. When complete, GMPS would be the first FPGA online ML-model-based regulation system in the Fermilab accelerator complex [126]. The promise of ML for accelerator controls is so apparent to the Department of Energy that a call for accelerator controls using ML was made to the national labs [127]. Of the two proposals submitted by Fermilab and approved by the DOE is the Real-time Edge AI for Distributed Systems (READS) project. READS is actually two projects. The first READS project will create a complimentary ML regulation system for slow extraction from the Delivery Ring to the future Mu2e experiment [128]. The second READS project will tackle a long-standing problem with de-blending beam losses in the Main Injector (MI) enclosure. The MI enclosure houses two accelerators, the MI and the Recycler. During normal operation, high intensity beams exist in both machines. One to use ML to help regulate slow spill in the Delivery ring to Mu2e, and another to develop a real-time online model to de-blend losses coming from the Recycler and Main Injector accelerators which share an enclosure. Both READS projects will make use of FPGA online ML models for inference and will collect data at low latencies from distributed systems around the accelerator complex [129].

2.5 Neutrino and direct dark matter experiments

2.5.1 Accelerator Neutrino Experiments

Context:

Accelerator neutrino experiments detect neutrinos with energies ranging from a few tens of MeV up to about 20 GeV. The detectors can be anywhere from tens of meters away from the neutrino production source, to as far as away as 1500 km. For experiments with longer baselines it is common for experiments to consist of both a near ($\sim$1 km baseline) and a more distant far detector (100's km baseline). Accelerator neutrino experiments focused on long-baseline oscillations use highly pure muon neutrino beams, produced by pion decays in flight. By using a system of magnetic horns it is possible to produce either a neutrino, or antineutrino beam. This ability is particularly useful for CP-violation measurements. Other experiments use pions decaying at rest, which produce both muon and electron flavors.

The primary research goal of many accelerator neutrino experiments is to perform neutrino oscillation measurements; the process by which neutrinos created in one flavor state are observed interacting as different flavor states after traveling a given distance. Often this takes the form of measuring electron neutrino appearance and muon neutrino disappearance. The rate of oscillation is energy-dependent, and so highly accurate energy estimation is essential. Another key research goal for accelerator neutrinos is to measure neutrino cross-sections, which in addition to accurate energy estimation requires the identification of the particles produced by the neutrino interaction.

Challenges:

Accelerator neutrino experiments employ a variety of detector technologies. These range from scintillator detectors such as NOvA (liquid), MINOS (solid), and MINERvA (solid), to water Cherenkov detectors such as T2K, and finally liquid argon time projection chambers such as MicroBooNE, ICARUS, and DUNE. Pion decay-at-rest experiments (COHERENT, JSNS^2) use yet different technologies (liquid and solid scintillators, as well as solid-state detectors). The individual challenges and solutions are unique to each experiment, though common themes do emerge.

Neutrino interactions are fairly uncommon due to their low cross-section. Some experiments can see as few as one neutrino interaction per day. This, combined with many detectors being close to the surface, means that analyses have to be highly efficient whilst achieving excellent background rejection. This is true both in online data taking and offline data analysis.

As experiments typically have very good temporal and/or spatial resolution it is often fairly trivial to isolate entire neutrino interactions. This means that it is then possible to use image recognition tools such as CNNs to perform classification tasks. As a result, many experiments initially utilized variants of GoogLeNet, though many are now transitioning to use GNNs and networks better able to identify sparse images.

Existing and Planned Work:

As discussed in Section 2.5.2, DUNE will use machine learning in its triggering framework to handle its immense data rates and to identify candidate interactions, for both traditional neutrino oscillation measurements and for candidate solar and supernova events. Accelerator neutrino experiments have successfully implemented machine learning techniques for a number of years, the first such example being in 2017 [130], where the network increased the effective exposure of the analysis by 30%. Networks aimed at performing event classification are common across many experiments, with DUNE having recently published a network capable of exceeding its design sensitivity on simulated data and which includes outputs that count the numbers of final state particles from the interaction [131].

Experiments are becoming increasingly cognizant of the dangers of networks learning features of the training data beyond what is intended. For this reason, it is essential to carefully construct training datasets such that this risk is reduced. However, it is not possible to correct or quantify bias which is not yet known; therefore the MINERvA experiment has explored the use of a domain adversarial neural network [132] to reduce unknown biases from differences in simulated and real data. The network features a gradient reversal layer in the domain network (trained on data), thus discouraging the classification network (trained on simulation) to learn from any features that behave differently between the two domains. A more robust exploration of the machine learning applied to accelerator neutrino experiments can be found here in Ref. [133].

2.5.2 Neutrino Astrophysics

Context:

Neutrino astrophysics spans a wide range of energies, with neutrinos emitted from both steady-state and transient sources with energies from less than MeV to EeV scale. Observations of astrophysical neutrinos are valuable both for the understanding of neutrino sources and for probing fundamental physics. Neutrino detectors designed for observing these tend to be huge scale (kilotons to megatons). Existing detectors involve a diverse range of materials and technologies for particle detection; they include Cherenkov radiation detectors in water and ice, liquid scintillator detectors and, liquid argon time projection chambers.

Astrophysical neutrinos are one kind of messenger contributing to the thriving field of *multimessenger astronomy*, in which signals from neutrinos, charged particles, gravitational waves, and photons spanning the electromagnetic spectrum are observed in coincidence. This field has had some recent spectacular successes [134–136]. For multimessenger transient astronomy, time is of the essence for sharing data and locating sources. *Directional information* from the neutrinos is critically valuable, to allow prompt location of the source by other messengers.

Potential interesting transient astrophysical sources include sources of ultra-high energy neutrinos, as well as nearby stellar core collapses. Neutrinos in the multi-GeV and higher range are emitted from distant cosmic sources, including kilonovae and blazars, and cubic-km-scale water-based Cherenkov detectors such as IceCube at the South Pole can produce fast alerts from single neutrino observations.

Core-collapse supernovae are another promising use case for fast machine learning. These are copious sources of few tens of MeV-scale neutrinos, which are emitted in a burst lasting a few tens of seconds [137, 138]. The neutrinos are prompt after core collapse (as will be gravitational waves) but observable electromagnetic radiation will not emerge for anywhere from tens to 10^6 s, depending on the nature of the progenitor and its envelope [139]. Low-latency information is therefore immensely valuable. Core-collapse supernovae are rare events within the distance range observable by current and near-future neutrino detectors. They occur only every several decades, which makes prompt and robust detection especially important. The SuperNova Early Warning System [140, 141] aims to provide a prompt alert from a coincidence of burst detections. However, pointing information from neutrinos is relatively difficult to extract promptly. Detectors with the capability for prompt pointing thanks to the anisotropy of neutrino interactions (i.e. the interaction products that remember where the neutrino came from) offer the best prospects, but these need to be able to select neutrino events from background and reconstruct their directions with very low latency.

Presupernova neutrinos are another interesting possibility. In the final stages of stellar burning, one expects a characteristic uptick in neutrino luminosity and average energy, producing observable events in detectors for nearby progenitors. This could give a warning of hours or perhaps days before core collapse for the nearest progenitors. For this case, fast selection of neutrino-like events and reconstruction of their directional information for background reduction is needed.

Challenges:

The challenges, in general, are fast selection and reconstruction of neutrino event (interaction) information. The specifics of the problem depend on the particular detector technology, but in general, the charged particle products of a neutrino interaction will have a distinctive topology or other signature and must be selected from a background of cosmic rays, radiologicals, or detector noise. Taking as an example a liquid argon time projection chamber like the Deep Underground Neutrino Experiment (DUNE), neutrino-induced

charged particles produce charge and light signals in liquid argon. Supernova neutrino interactions appear as small (tens of cm spatial scale) stubs and blips [142, 143]. The recorded neutrino event information from the burst can be used to reconstruct the supernova direction to ∼5–10° for core collapse at 10 kpc distance [144, 143]. The neutrino events need to be selected from a background of radioactivity and cosmogenics, as well as detector noise, requiring background reduction of many orders of magnitude. Total data rate amounts to ∼40 Tb/s. The detector must take data for a decade or more at this rate, with near-continuous uptime.

For steady-state signals such as solar neutrinos, triggering on individual events in the presence of large backgrounds is a challenge that can be addressed with machine learning. For burst signals, the triggering is a different problem: the general strategy is to read out all information on every channel within a tens-of-seconds time window, for the case of a triggered burst. This leads to the subsequent problem of sifting the signal events and reconstructing sufficient information on a very short timescale to point back to the supernova. The required timescale is minutes, or preferably seconds. Both the event-by-event triggering and fast directional reconstruction can be addressed with fast machine learning.

Existing and Planned Work:

There are a number of existing efforts towards the use of machine learning for particle reconstruction in neutrino detectors including water Cherenkov, scintillator, and liquid argon detectors. These overlap to some extent with the efforts described in Sec. 2.5.1. Efforts directed specifically towards real-time event selection and reconstruction are ramping up. Some examples of ongoing efforts can be found in Refs. [131, 145–149, 133].

2.5.3 Direct Detection Dark Matter Experiments

Context:

Direct dark matter (DM) search experiments take advantage of the vastly abundant DM in the universe and are searching for direct interactions of DM particles with the detector target material. The various target materials can be separated into two main categories, crystals and liquid noble gases, though other material types are subject to ongoing detector R&D efforts [150, 151].

One of the most prominent particle DM candidates is the WIMP (weakly interacting massive particle), a thermal, cold DM candidate with an expected mass and coupling to Standard Model particles at the weak scale [152]. However, decades of intensive searches both at direct DM and at collider experiments have not yet been able to discover[2] the vanilla WIMP while excluding most of the parameter space of the simplest WIMP hypothesis [151]. This instance has lead to a shift in paradigm for thermal DM towards increasingly lower masses well below 1 GeV (and thus the weak scale) [154] and as low as a few keV, i.e. the warm DM limit [155]. Thermal sub-GeV DM is also referred to as light dark matter (LDM). Other DM candidates that are being considered include non-thermal, bosonic candidates like dark photons, axions and axion-light particles (ALPs) [156–158].

The most common interactions direct DM experiments are trying to observe are thermal DM scattering off either a nucleus or an electron and the absorption of dark bosons under the emission of an electron. The corresponding signatures are either nuclear recoil or electron recoil signatures.

[2] The DAMA/NaI and subsequent DAMA/LIBRA experiment, claim the direct observation of DM particles in the galactic halo [153], but the results are in tension with negative results from similar experiments [151].

Challenges:

In all mentioned interactions, and independent of the target material, a lower DM mass means a smaller energy deposition in the detector and thus a signal amplitude closer to the baseline noise. Typically, the baseline noise has non-Gaussian contributions that can fire a simple amplitude-over-threshold trigger even if the duration of the amplitude above threshold is taken into account. The closer the trigger threshold is to the baseline, the higher the rate of these spurious events. In experiments which cannot read out raw data continuously and which have constraints on the data throughput, the hardware-level trigger threshold has thus to be high enough to significantly suppress accidental noise triggers.

In the hunt for increasingly lower DM masses, however, an as-low-as-possible trigger threshold is highly desirable, calling for a more sophisticated and extremely efficient event classification at the hardware trigger level. Particle-induced events have a known, and generally constant, pulse-shape while non-physical noise "events" (e.g. induced by the electronics) generally have a varying pulse-shape which is not necessarily predictable. A promising approach in such a scenario is the use of machine learning techniques for most efficient noise event rejection in real-time allowing to lower the hardware-level trigger threshold, and thus the low mass reach in most to all direct DM searches, while remaining within the raw data read-out limitations imposed by the experimental set-up.

Existing and Planned Work:

Machine learning is already applied by various direct DM search experiments [159–161], especially in the context of offline data analyses. However, it is not yet used to its full potential within the direct DM search community. Activities in this regard are still ramping up but with increasing interest, efforts, and commitment. Typical offline applications to date are the reconstruction of the energy or position of an event and the classification of events (e.g. signal against noise or single-scattering against multiple-scattering). In parallel R&D has started on real-time event classification within the FPGA-level trigger architecture of the SuperCDMS experiment [162] with the long-term goal of lowering the trigger threshold notably closer to the baseline noise without triggering on spurious events. While these efforts are being conducted within the context of SuperCDMS the goal is a modular trigger solution for easier adaption to other experiments.

2.6 Electron-Ion Collider

Context:

The Electron-Ion Collider (EIC) will support the exploration of nuclear physics over a wide range of center-of-mass energies and ion species, using highly-polarized electrons to probe highly-polarized light ions and unpolarized heavy ions. The frontier accelerator facility will be designed and constructed in the U.S. over the next ten years. The requirements of the EIC are detailed in a white paper [163], the 2015 Nuclear Physics Long Range Plan [164], and an assessment of the science by the National Academies of Science [165]. The EIC's high luminosity and highly polarized beams will push the frontiers of particle accelerator science and technology and will enable us to embark on a precision study of the nucleon and the nucleus at the scale of sea quarks and gluons, over all of the kinematic range that is relevant as described in the EIC Yellow Report [166].

Challenges:

While the event reconstruction at the EIC is likely easier than the same task at present LHC or RHIC hadron machines, and much easier than for the High-Luminosity LHC, which will start operating two years earlier than the EIC, possible contributions from machine backgrounds form a challenge. The expected gain in CPU performance in the next ten years as well as the possible improvement in the reconstruction software from the use of AI and ML techniques give a considerable margin to cope with higher event

complexity that may come by higher background rates. Software design and development will constitute an important ingredient for the future success of the experimental program at the EIC. Moreover, the cost of the IT related components, from software development to storage systems and to distributed complex e-Infrastructures can be raised considerably if a proper understanding and planning is not taken into account from the beginning in the design of the EIC. The planning must include AI and ML techniques, in particular for the compute-detector integration at the EIC, and training in these techniques.

Existing and Planned Work:

Accessing the EIC physics of interest requires an unprecedented integration of the interaction region (IR) and detector designs. The triggerless DAQ scheme that is foreseen for the EIC will extend the highly integrated IR-detector designs to analysis. A seamless data processing from DAQ to analysis at the EIC would allow to streamline workflows, e.g., in a combined software effort for the DAQ, online, and offline analysis, as well as to utilize emerging software technologies, in particular fast ML algorithms, at all levels of data processing. This will provide an opportunity to further optimize the physics reach of the EIC. The status and prospects for "AI for Nuclear Physics" have been discussed in a workshop in 2020 [167]. Topics related to fast ML are intelligent decisions about data storage and (near) real-time analysis. Intelligent decisions about data storage are required to ensure the relevant physics is captured. Fast ML algorithms can improve the data taken through data compactification, sophisticated triggers, and fast online analysis. At the EIC, this could include automated alignment and calibration of the detectors as well as automated data-quality monitoring. A (near) real-time analysis and feedback enables quick diagnostics and optimization of experimental setups as well as significantly faster access to physics results.

2.7 Gravitational Waves

Context:

As predicted by Einstein in 1916, gravitational waves are fluctuations in the gravitational field which within the theory of general relativity manifest as a change in the spacetime metric. These ripples in the fabric of spacetime travel at the speed of light and are generated by changes in the mass quadruple moment, as, for example, in the case of two merging black holes [168]. To detect gravitational waves, the LIGO/Virgo/KAGRA collaborations employ a network of kilometer-scale laser interferometers [169–172]. An interferometer consists of two perpendicular arms; as the gravitational wave passes through the instrument, it stretches one arm while compressing the other in an alternating pattern dictated by the gravitational wave itself. Such length difference is then measured from the laser interference pattern.

Gravitational waves are providing a unique way to study fundamental physics, including testing the theory of general relativity at the strong field regime, the speed of propagation and polarization of gravitational waves, the state of matter at nuclear densities, formation of black holes, effects of quantum gravity and more. They have also opened up a completely new window for observing the Universe and in a complementary way to one enabled by electromagnetic and neutrino astronomy. This includes the study of populations, including their formation and evolution, of compact objects such as binary black holes and neutron stars, establish the origin of gamma-ray bursts (GRBs), measure the expansion of the Universe independently of electromagnetic observations, and more [173].

Challenges:

In the next observing run in 2022, LIGO, Virgo, and KAGRA will detect an increasing number of gravitational-wave candidates. This poses a computational challenge to the current detection framework, which relies on matched-filtering techniques that match parameterized waveforms (templates) from simulations into the gravitational-wave time series data [168, 174, 175]. Matched filtering scales poorly as

the low-frequency sensitivity of the instrument improves and the search parameter space of the gravitational wave expands to cover spin effects and low mass compact objects. To estimate the physical properties of the gravitational wave, stochastic Bayesian posterior samplers, such as Markov-chain Monte Carlo and Nested Sampling, have been used until now. Such analysis approaches can take up hours to days to complete [176]. The latency introduced by the current search and parameter estimation pipeline is non-negligible and can hinder electromagnetic follow-ups of time-sensitive sources like binary neutron stars, supernovae, and other, yet unknown, systems.

Observations of gravitational-wave transients are also susceptible to environmental and instrumental noise. Transient noise artifacts can be misidentified as a potential source, especially when the gravitational-wave transients have an unknown morphology (e.g. supernovae, neutron star glitches). Line noise in the noise spectrum of the instruments can affect the search for continuous gravitational waves (e.g. spinning neutron stars) and stochastic gravitational waves (e.g astrophysical background of gravitational waves from unresolved compact binary systems). These noise sources are difficult to simulate, and current noise subtraction techniques are insufficient to remove the more complex noise sources, such as non-linear and non-stationary ones.

Existing and Planned Work:

In recent years, machine learning algorithms have been explored in different areas of gravitational-wave physics [177]. CNNs have been applied to detect and categorize compact binary coalescence gravitational waves [178–182], burst gravitational waves from core-collapse supernovae [183–185], and continuous gravitational waves [186, 187]. Besides, recurrent neural networks (RNNs) based autoencoders have been explored to detect gravitational wave using an unsupervised strategy [188]. FPGA-based RNNs are also explored to show the potential in low-latency detection of gravitational wave [189]. Applications of ML in searches of other types of gravitational waves, such as generic burst and stochastic background, are currently being explored. Moreover, probabilistic and generative ML models can be used for posterior sampling in gravitational-wave parameter estimation and achieve comparable performance to Bayesian sampler on mock data while taking significantly less time to complete [190–192]. ML algorithms are also being used to improve the gravitational-wave data quality and subtract noise. Transient noise artifacts can be identified and categorized from their time-frequency transforms and constant-Q transforms [193, 194] or through examining hundreds of thousands of LIGO's auxiliary channels [195]. These auxiliary channels can also be used to subtract quasi-periodic noise sources (e.g. spectral lines) [196, 197]. Although ML algorithms have shown a lot of promise in gravitational-wave data analysis, many of these algorithms are still at the proof-of-concept stage and have not yet been successfully applied in real-time analysis. Current efforts seek to create a computational infrastructure for low-latency analysis, improve the quality of the training data (e.g. expanding the parameter space, using a more realistic noise model), and better quantify the performance of these algorithms on longer stretches of data.

2.8 Biomedical engineering

Context:

We have seen an explosion of biomedical data, such as biomedical images, genomic sequences, and protein structures, due to the advances in high-resolution and high-throughput biomedical devices. AI-augmented reality-based microscopy [198] enables automatic analysis of cellular images and real-time characterization of cells. Machine learning is used *in-silico* prediction of fluorescent labels, label-free rare cell classification, morphology characterization, and RNA sequencing [199–203]. For in-situ cell sorting, real-time therapy response prediction, and augmented reality microscope-assisted diagnosis [198, 204, 205],

it is important to standardize and optimize data structure in deep learning models to increase speed and efficiency. Various machine-learning-based algorithms for detecting hemorrhage and lesions, accelerating diagnosis, and enhancing medical video and image quality have also been proposed in biopsy analysis and surgery assistance.

Challenges:

A major challenge for clinical application of ML is inadequate training and testing data. The medical data annotation process is both time-consuming and expensive for large image and video datasets which require expert knowledge. The latency of trained models' inference also introduces computational difficulties in performing real-time diagnosis and surgical operation. The quality of services for time-critical healthcare requires less than 300 milliseconds as real-time video communication [206]. For reaching 60 frames per second (FPS) high-quality medical video, the efficiency and performance of a deep learning model become crucial.

Existing and Planned Work:

Many changes in ML algorithms have involved improvements to performance both in accuracy and inference speed. Some state-of-art machine learning models can reach a high speed for inference. For example, *YOLOv3-tiny* [207], an object detection model commonly used for medical imaging, can process images at over 200 FPS on a standard dataset with producing reasonable accuracy. Currently both GPU- and FPGA-based [208–210], distributed networks of wireless sensors connected to cloud ML (edge computing), and 5G-high-speed-WiFi-based ML models are deployed in medical AI applications [211–213]. ML models for fast diagnosis of stroke, thrombosis, colon polyps, cancer, and epilepsy have significantly reduced the time in lesion detection and clinical decision [214–218]. Real-time AI-assisted surgery can improve perioperative workflow, perform video segmentation [219], detection of surgical instruments [220], and visualization of tissue deformation [221]. High-speed ML is playing a critical role in digital health, *i.e.*, remote diagnosis, surgery, and monitoring [212].

2.9 Health Monitoring

Context:

Our habits and behaviors affect our health and wellness. Unhealthy behaviors such as smoking, consuming excessive alcohol, or medication non-adherence often has an adverse effect on our health [222–225]. Traditional behavior monitoring approaches relied on self-reports, which were often biased and required intense manual labor [226]. With the advent of mobile and wearable devices, it is gradually becoming possible to monitor various human behaviors automatically and unobtrusively. Over the years, researchers have either developed custom wearable hardware or have used off-the-shelf commercial devices for mobile and wearable health (mHealth) monitoring [227–233]. The automatic and unobtrusive monitoring capability of these devices makes it possible to detect, identify and monitor behaviors, including unhealthy behaviors in a free-living setting.

Challenges:

There are various challenges associated with monitoring habits and behaviors using wearable devices. Firstly, these devices should be capable of monitoring unhealthy behaviors accurately, and in real-time. The occurrence of these unhealthy behaviors in a free-living setting is often sparse as compared to other behaviors and thus it is important to spot them accurately, whenever they occur. Most existing systems take an offline ML approach of detecting these unhealthy behaviors, where the ML algorithm identifies these behaviors well after they have occurred. An offline approach prevents providing interventions that can minimize unhealthy behaviors. Thus, it is necessary to develop ML approaches that can detect these

behaviors online, and in real-time, so that interventions such as just-in-time adaptive interventions (JITAIs) can be delivered. Secondly, since these devices capture sensitive information, it is necessary to ensure that an individual's privacy is preserved. Privacy-preserving approaches such as locally processing the data on-device can be taken so that critical information does not leave the device. Finally, these behaviors can occur in various heterogeneous environments and thus the health monitoring system should be agnostic to where the behavior occurs. Such monitoring requires developing multiple machine learning models for diverse environments.

Existing and Planned Work:

While existing work has ventured in various directions, there is a growing need for sensing health biomarkers correctly and developing ML approaches that are fast and can accurately identify these biomarkers. Researchers have focused on developing novel sensing systems that can sense various health behaviors and biomarkers [234–240]. Historically, most of these novel sensing techniques were tested in controlled settings, but more recently researchers are ensuring that these systems can work seamlessly in free-living settings as well. This often requires developing multiple ML models, each catering to a specific context and environment. A new trend in this field has started relying on implementing models that can be implemented on-device and are both quick and accurate in detecting these behaviors. In addition to providing real-time interventions [241, 242], on-device monitoring of these behaviors can reduce privacy concerns [243]. However, since wearable devices themselves might not be capable of processing the data, federated machine learning approaches are also being explored recently by several researchers [244].

2.10 Cosmology

Context:

Cosmology is the study of the Universe's origin (big bang), evolution, and future (ultimate fate). The large-scale dynamics of the universe are governed by gravity, where dark matter plays an important role, and the accelerating expansion rate of the universe itself, caused by the so-called dark energy. A non-exhaustive list of cosmological probes includes type Ia supernovae [245–249], cosmic microwave background [250–254], large-scale structures (including baryon acoustic oscillation) [255–258], gravitational lensing [259–263] and 21 cm cosmology [264–267].

Challenges:

As astronomy is approaching the big data era with next-generation facilities, such as the Nancy Grace Roman Space telescope, Vera C. Rubin Observatory, and Euclid telescope, the uncertainty budget in the estimation of cosmological parameters is no longer expected to be dominated by statistical uncertainties, but rather by systematic ones; understanding such uncertainties can lead to attaining sub-percent precision. On the other hand, the immense stream of astronomical images will be impossible to analyze in a standard fashion (by human interaction); new automated methods are needed to extract valuable pieces of cosmological data.

Existing and future work:

Current efforts are focused on applying ML techniques to study the influence of systematic biases on available analysis methods (e.g., for purposes of fitting or modeling) or on developing new methods to overcome present limitations; for example CNNs can be adapted to spherical surfaces to generate more accurate models when producing weak lensing maps [268], or to remove noise from cosmic microwave background maps [269]. In addition, discovery and classification engines are being developed to extract useful cosmological data from next-generation facilities [270–273]. Furthermore, ML is also being used in cosmological simulations to test new analyses and methods and to set the foundations for the first operation

of such new facilities [274–276]. An extensive list of published ML applications in cosmology can be found in https://github.com/georgestein/ml-in-cosmology.

2.11 Plasma Physics

Context:

The focus of this description is on the Plasma Physics/Fusion Energy Science domain with regard to the major system constraints encountered for existing and expected algorithms and data representations when dealing with the challenge of delivering accelerated progress in AI—enabled deep machine learning prediction and control of magnetically-confined thermonuclear plasmas. Associated techniques have enabled new avenues of data-driven discovery in the quest to deliver fusion energy—identified by the 2015 CNN "Moonshots for the 21st Century" televised series as one of 5 prominent grand challenges for the world today.

Challenges:

An especially time-urgent and challenging problem is the need to reliably predict and avoid large-scale major disruptions in "tokamak systems" such as the EUROFUSION Joint European Torus (JET) today and the burning plasma ITER device in the near future—a ground-breaking $25B international burning plasma experiment with the potential capability to exceed "breakeven" fusion power by a factor of 10 or more with "first plasma" targeted for 2026 in France. The associated requirement is for real-time plasma forecasting with control capabilities operative during the temporal evolution of the plasma state well before the arrival of damaging disruptive events. High-level supervisory control of many lower-level control loops via actuators (analogous to advanced robotics operations) will be essential for ITER and future burning plasmas to protect the facility and to avoid operational limits (for magnets, walls, plasma position, stability, etc.) while optimizing performance.

Existing and Planned Work:

In short, an overarching goal here involves developing realistic *predictive plasma models of disruptions integrated with a modern plasma control system to deliver the capability to design experiments before they are performed*. The associated novel AI-enabled integrated modeling tool would clearly be of great value for the most efficient and safe planning of the expensive discharges in ITER and future burning plasmas. Verification, validation, and uncertainty quantification of associated components would include: (1) development of predictive neural net models of the plasma and actuators that can be extrapolated to burning plasma scales via advanced Bayesian reinforcement learning methods that incorporate prior information into efficient inference algorithms; (2) systematic well-diagnosed experimental validation studies of components in the integrated plasma forecasting models involving massive amounts of data from major tokamak experiments worldwide (e.g., DIII-D in the US, KSTAR & EAST in Asia, JET in Europe, followed by JT60 SA—the large superconducting device in Japan that will precede ITER). This would ideally lead to a mature AI-enabled comprehensive control system for ITER and future reactors that feature integration with full pilot-plant system models.

At present, a key challenge is to deliver significantly improved methods of prediction with better than 95% predictive accuracy to provide advanced warning for disruption avoidance/mitigation strategies to be effectively applied before critical damage can be done to ITER. Significant advances in the deployment of deep learning recurrent and CNNs are well illustrated in Princeton's Deep Learning Code—"FRNN"—that have enabled the rapid analysis of large complex datasets on supercomputing systems. Associated acceleration of progress in predicting tokamak disruptions with unprecedented accuracy and speed is described in [277]. Included in this paper (and extensive references cited therein) are descriptions of FES

data representation for physics features (density, temperature, current, radiation, fluctuations, etc.) and the nature of key plasma experiments featuring detectors/diagnostics with frame (event-based) level of accuracy accounting for required "zero-D" (scalar) and higher-dimension signals and real-time resolution recorded at manageable data rates. Rough future estimates indicate that ITER will likely require dealing with the challenge of processing and interpreting exabytes of complex spatial and temporal data.

Since simulation is another vital aspect of ITER data analysis, dealing with the associated major computational expenses will demand the introduction of advanced compressional methods. More generally, real-time predictions based on actual first-principles simulations are important for providing insights into instability properties and particle-phase space dynamics. This motivates the development of an AI-based "surrogate model"—for example, of the well-established HPC "gyrokinetic" particle-in-cell simulation code GTC [278] that would be capable of accurately simulating plasma instabilities in real-time. Data preparation and training a surrogate model – e.g., "SGTC"—provides a clear example of the modern task of integration/connection between modern High Performance Computing (HPC) predictive simulations with AI-enabled Deep Learning/Machine Learning campaigns. These considerations also serve to further illustrate/motivate the need to integrate HPC & Big Data ML approaches to expedite the delivery of scientific discovery.

As a final note, the cited paper [277] represents the first adaptable predictive DL software trained on leadership class supercomputing systems to deliver accurate predictions for disruptions across different tokamak devices (DIII-D in the US and JET in the UK). It features the unique statistical capability to carry out efficient "transfer learning" via training on a large database from one experiment (i.e., DIII-D) and be able to accurately predict disruption onset on an unseen device (i.e., JET). In more recent advances, the FRNN inference engine has been deployed in a real-time plasma control system on the DIII-D tokamak facility in San Diego, CA. As illustrated in slides 18 through 20 of the attached invited presentation slide deck, this opens up exciting avenues for moving from passive disruption prediction to active real-time control with subsequent optimization for reactor scenarios.

2.12 ML for Wireless Networking and Edge Computing

Context:

Wireless devices and services have become a crucial tool for collecting and relaying big data in many scientific studies. Moreover, mobility information has proven to be extremely useful in understanding human activities and their impact on the environment and public health. The exponential growth of data traffic is placing significant pressure on the wireless infrastructure. In particular, inter-cell interference causes large variability in reliability and latency. To meet user demands for data communication and value-added AI/ML services, wireless providers must 1) develop more intelligent learning algorithms for radio resource management that adapt to complicated and ever-changing traffic and interference conditions; and 2) realize many ML/AI computations and functionalities in edge devices to achieve lower latency and higher communication efficiency.

Challenges:

Conventional implementations of ML models, especially deep learning algorithms, lag far behind the packet-level dynamics for utility. Moreover, existing ML/AI services are often performed in the cloud for efficiency at the expense of communication overhead and higher latency. A major challenge in the wireless networking and edge computing context is to build a computing platform that can execute complex ML models at relevant timescales (< 10 ms) within small cell access points.

Existing and planned work:

Researchers have proposed a variety of learning algorithms to perform specific radio resource management tasks using artificial neural networks [279–282]. Some of the first proposals to train a NN to perform transmit power control adopts supervised learning [283, 284]. More recent proposals adopt deep reinforcement learning approaches that work better with channel and network uncertainties and require little training data *a priori* [285–288]. A number of works are focused on the convergence of edge computing and deep learning [289–291]. A specific set of work is on federated learning where participants jointly train their models in lieu of sending all their data to a central controller for training purposes [292–295]. All of the preceding work basically ends at the simulation stage for the lack of practical ML/AI solutions that are fast and computationally efficient at the same time. More specifically, the research challenge is to develop a computing platform that can execute complex ML models at a very fast timescale (< 10 ms) and can also be equipped in small cell access points. One project with a potentially very high impact is to map intelligent radio resource management algorithms (such as that of [285]) onto an FPGA device suitable for deployment in a large network of connected and interfering access points. Another interesting project is to build a federated learning system to conduct time-sensitive ML for Internet-of-Things (IoT) devices where transferring data to centralized computing facilities is latency-prohibitive. This opens up entirely new possibilities for low-cost closed-loop IoT devices in healthcare, smart buildings, agriculture, and transportation.

3 KEY AREAS OF OVERLAP

Real-time, accelerated AI inference show promises in improving the discovery potential at current and planned scientific instruments across the domains as detailed in Sec. 2. Design of high performant specialty systems for real-time/accelerated AI applications requires particular attention to the figure-of-merit of the target domain's ML algorithm. It might be dominated by its latency per inference, computational cost (e.g., power consumption), reliability, security, and ability to operate in extreme environments (e.g., radiation). For instance, ML might need to: trigger acquisition systems for rare events with $\sim$100 ns latency on the Large Hadron Collider [37]; analyze multi-channel ambulatory health monitors at kilohertz frequencies where wireless transfer of data is not possible due to power limitations ($\sim$50 iPhone batteries/day for data transfer) or security requirements; or to keep pace with materials spectroscopy data streams on the order of terabits per second [296]. Furthermore, real-time analysis of advanced scientific instrumentation must have an uninterrupted allocation of computing resources and patient sensitive information processed by wireless health devices must be secured. Such features and characteristics create quantifiable guidelines for understanding distinctions and commonalities among domains and applications. Thereby, we can coordinate efforts towards creating fundamental design principles and tools, which may address needs across seemingly disparate domains. Appropriate data representation is an essential first step of the design process as it determines the choice of NN architecture to be implemented in real-time systems that need to meet the performance targets outlined above. Prominent data representations of different scientific instruments are summarized below. Other areas of overlap across domains such as NN and hardware co-design tools and workflows, NN complexity reduction with quantization and pruning are also recent technology advancements in real-time/accelerated AI and therefore are outlined in Section 4.

3.1 Data representations

Data representation used in a particular domain influences both the computation system and data storage. One global classification for data representations across domains can be considered as being into raw versus reconstructed data. The data representation often varies depending on the stage of the reconstruction and the upstream steps in the data processing pipeline. Existing applications include fully connected NNs that often take pre-processed expert feature variables as inputs or CNNs when the data is of image nature. On-going development of domain knowledge-inspired NN algorithms could further take advantage of the expert features in the accuracy and efficiency as detailed below. To fully exploit the power of advanced NNs and bring it closer to data creation for minimum information loss, a more suitable representation of the raw data, e.g as point clouds, needs to be employed. Prominent representations for raw data from different experimental and measurement systems are:

- **Spatial Data**: Used for describing physical objects in geometric space. There are two main types, called vector and raster data. Vector data, in turn, can be comprised of points, lines, or polygons. Raster data refers to a grid of pixels, such as images, but pixels can also represent other measurements such as intensity, charge, field strength, etc.
- **Point Clouds**: Can be considered a type of spatial data. This data representation is created by collating a set of spatial data, i.e., points in a 3D space, that usually form an object in space collectively.
- **Temporal Data**: Used to represent the state of a system/experiment at a particular time. Data collected across time, in a specific order, is classified in this manner. Time-series data is a subset of this representation, where data is sampled at regular time intervals. An example of time-series data can be seen in Fig. 4, for the specific case of supernova classification.

- **Spatio-Temporal Data**: Measurements and observations of a system can be collected across both the space and time dimensions. In that case, the data can be considered spatio-temporal.
- **Multispectral Data**: Used to represent outputs of multiple sensors that capture measurements from multiple bands of the electromagnetic spectrum. Multispectral representation is commonly used in the context of imaging, involving sensors that are sensitive to different wavelengths of light. This usually involves in the order of a few to 10s of spectra.
- **Hyperspectral Data**: Used to represent measurements from a high number of spectra, e.g., in the order of 100s. These images collected from different narrow-band spectra are combined into a so-called hyperspectral cube with three main dimensions. The first two reference the 2D spatial placement (e.g., earth's surface) while the third dimension represents the complete spectrum content at each "pixel" location.

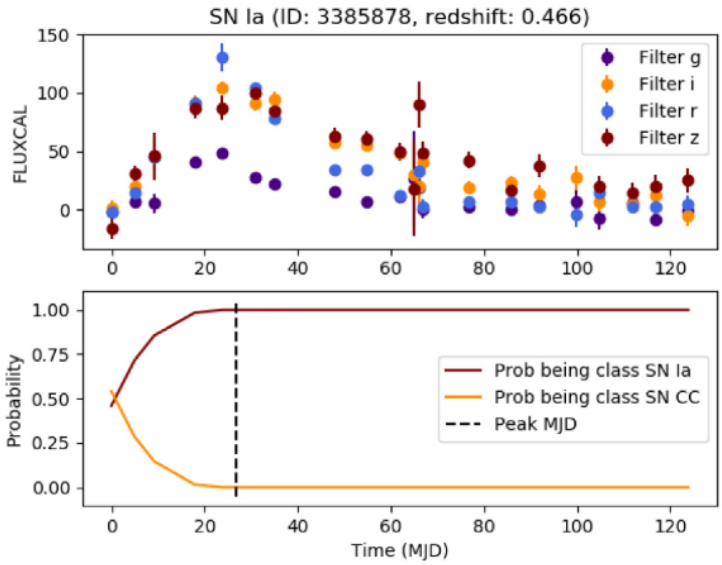

Figure 4. Simulated type Ia supernova light-curve and classification. Top: calibrated flux evolution in different DES band-passes as a function of normalized time (the first photometric measurement is set to time equals zero). Bottom: Baseline RNN classification probability evolution with respect of time, no host-galaxy redshift information was provided. At each photometric measurement, classification probability is obtained. The maximum light of the simulated supernova is shown in a gray dashed line and the simulated redshift of the supernovae is shown on the top $z = 0.466$. We highlight that redshift is not used for this classification but can improve results. Our baseline RNN classifies this light-curve as type Ia SN with great accuracy before maximum light, it only requires a handful of photometric epochs. [297].

In Table 1, we match these data representations to scientific application domains and give a brief description. We highlight the data representations which are particularly important for a specific domain. We will give more detailed examples below.

Cost of data communication (in terms of latency) and data storage (in terms of the cost of acquiring and managing the physical storage resources) present important challenges. Particularly, application domains, which require real-time analysis and/or real-time feedback demand highly optimized data analytics solutions. Applications that rely on hyper-spectral data are faced with an ever-increasing rate of data input across the electromagnetic spectrum. High-speed data reduction is required in these domains. Applications that generate large-scale point clouds similarly demand efficient compression on their spatial data. Application domains that handle multi-spectral data with limited spatial resolution require ultra-fast reconstruction in order to enable real-time control feedback. Another challenge is posed by applications that rely on accurate analysis of streaming time-series data, yet they are forced to perform under highly limited storage and communication resources, either due to privacy and security concerns or limitations of the associated edge devices.

Some current efforts in developing ML solutions to data processing front-ends focus on developing autoencoder based compression engines [298, 39]. ML-based dimensionality reduction for hyper-spectral data is another direction which has drawn attention [299]. Deep learning-based approaches are investigated for image reconstruction; the field of material sciences being one of the most active fields in that regards [300].

Domain	Spatial	Point Cloud	Temporal	Spatio-Temporal	Multi/Hyper-spectral	Examples
LHC	✓✓	✓✓	✓	✓	–	detector reconstruction
Belle-II/Mu2e	✓✓	✓✓	–	–	–	track reconstruction
Material Synthesis	✓	–	✓	✓✓	✓✓	high-speed plasma imaging
Accelerator Controls	✓	–	✓✓	–	–	beam sensors
Accelerator neutrino	✓✓	✓✓	✓	✓	–	detector reconstruction
Direct detection DM	✓✓	✓✓	✓	✓	–	energy signatures
EIC	✓✓	✓✓	✓	✓	–	detector reconstruction
Gravitational Waves	✓	–	✓✓	–	–	laser inference patterns
Biomedical engineering	✓✓	–	–	✓✓	–	cell and tissue images
Health Monitoring	✓	–	✓✓	✓	✓	physiological sensor data
Cosmology	✓✓	✓✓	✓✓	✓	✓✓	lensing/radiation maps
Plasma Physics	✓	–	✓✓	✓	–	detector actuator signals
Wireless networking	–	–	✓✓	–	–	electromagnetic spectrum

Table 1. Types of data representations and their relevance for the scientific domains discussed in this paper; ✓✓= Particularly important for domain, ✓= Relevant for domain

3.1.1 Expert Feature DNNs

One straightforward approach to building powerful domain-specific ML algorithms is to start with expert domain features and combine them in a neural network or other multivariate analysis technique. This embedded expertise has inherent advantages because the input features are interpretable, and correlations between features can yield insight into a particular task while optimizing performance. Furthermore, depending on the computational complexity of the domain features, the computation efficiency of such a machine learning approach can be greater than the direct use of raw features. However, the downside is that, by using expert features, we rely entirely on the informativeness of such new features.

Therefore, there is a lot of interest in automating the process of building informative new features from raw features. In image classification tasks, for example, a lot of progress has been made in extracting high-level data representations through deep neural networks DNNs [301]. In DNNs, layers of neurons above the original input signal are built to ensure that each new layer captures a more abstract representation of the data. Each layer constructs new features by forming nonlinear combinations of the features in the layer below. This hierarchical approach to feature construction has been effective in disentangling factors of variation in the data [302, 303, 301], and has been useful to construct informative and meaningful representations. In astronomical images, for example, a DNN starts with low-level pixel information, gradually capturing at upper layers edges, motifs, and eventually entire objects (e.g., galaxies) to provide a broad view of the Universe [304, 305]. The same applies to other fields of science. For example, detecting particles in large accelerators requires transforming low-level signals into dynamic patterns that can be ascribed to specific particles [306]. In medical imaging, there is a need to quickly identify abnormal tissue from low-level pixel information by gradually capturing global tissue patterns [307]. The importance of transforming the initial input data into meaningful abstract representations cannot be overstated: it remains one of the most powerful properties of modern neural network architectures.

Several challenges exist in the construction of increasingly abstract representations using DNNs. One challenge is to incorporate domain knowledge (e.g., physical constraints) into the neural network model. This is important to address the need for excessive amounts of data when training a DNN and narrow the gap in representational bias between the model and target concept. Under scarce data but abundant domain expertise, adding domain knowledge can expedite the training process [308], as well as improving the model generalization performance. Another challenge is to develop tools for model interpretability by

explaining the semantics of the representations embedded at each layer [309]. This is challenging due to the distributed representation of information in the network architecture.

Despite the lack of a formal mechanism to attain a seamless integration between a statistical model and domain knowledge, current approaches point to interesting directions, e.g., using knowledge to add training data or to change the loss function [310]. Model interpretability in DNNs has seen an upsurge in research over the past years [309]. Commonly, studies look at individual units and their activation patterns to elucidate what is learned across layers of neurons.

3.1.2 Frame-based images

Frame-based images are a suitable representation of the experimental data in multiple domains such as neutrino detection with time projection chambers in particle physics. An example of this data representation can be seen in Fig. 5 for an electron deposition in the ProtoDUNE neutrino detector. A spatial frame is shown by plotting the time coordinate "Tick" and wire position in space. Recent developments in neural network architectures exploit the sparsity of the images to reduce the computation complexity for real-time/accelerated ML applications. Other types of experimental data in HEP and many other domains can also be processed to be represented as frame-based images, although often not without information loss.

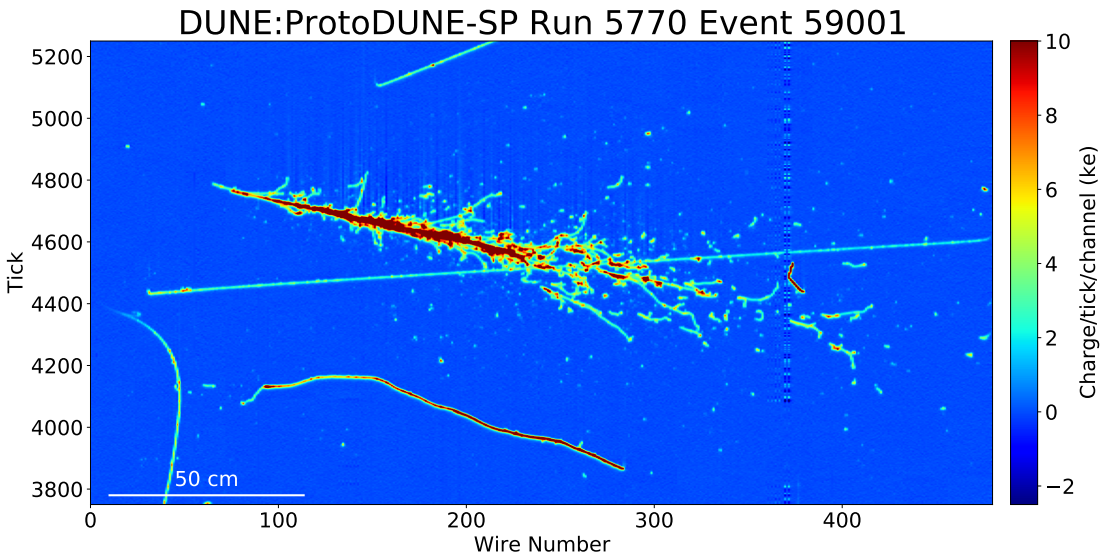

Figure 5. A 6 GeV/c electron event in the ProtoDUNE detector. The x-axis shows the wire number. The y-axis shows the time tick in the unit of $0.5\mu s$. The color scale represents the charge deposition.[]

3.1.3 Point clouds

Point cloud data representation is often used in HEP, where multiple frames of event-based measurements collected by a large number of detectors are combined into a data set. Across many HEP applications point clouds commonly help to represent particle jets with data sizes exceeding Pb/s. More broadly, point clouds can be used to capture any 3D space event and interactions of moving parts in space. A point cloud visualization of the CMS detector at the LHC is shown in Fig. 6. Remnants of proton-proton collisions create sensors signals in a customized and optimized detector geometry and points are illustrated in space. Various types of scan-based imaging data can be represented as point clouds. Other domains such as CT

and PET scanning in biomedical engineering and virtual reality also utilize this representation for imaging. 3D scanners used for product design, solid object modeling, architecture, and infrastructure design leverage point clouds as well. Many of these imaging tasks generate point clouds of sizes in the order of several GB to TB. Domains sharing point cloud representation (e.g., HEP and biomedical imaging) also commonly involve spatial characteristics.

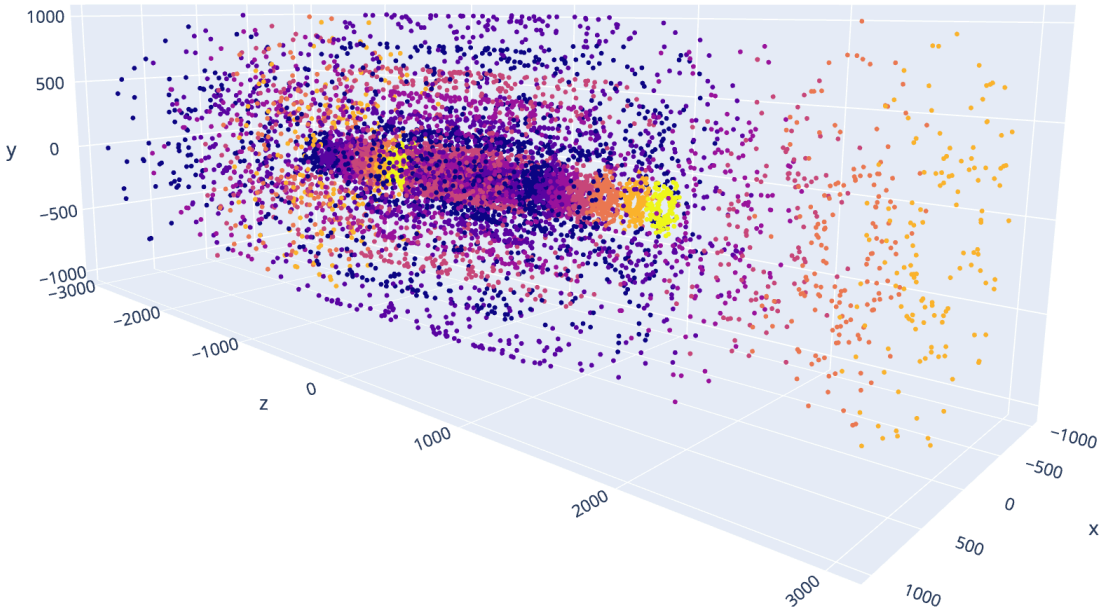

Figure 6. Visualization of particle tracking hits in 3D space from the TrackML Kaggle dataset [311]

3.1.4 Multi-/Hyperspectral Data

Multispectral data is common between wireless health monitoring and wireless communication systems. A set of physiological sensors, often representing different modalities, are combined into a multispectral data set for health monitoring and intervention systems. For wireless communication, signal interference and network traffic conditions are captured via multispectral data. Both domains capture this data across the time domain, so also exhibit temporal features. Furthermore, in both domains generated data size can be considered relatively smaller (ranging from 100s of Mb/s to 10s of Gb/s), compared to the rest of the domains discussed in this article. Hyperspectral data is used across many astronomy applications, medical imaging, and electron microscopy, which is used to drive many materials science design and discovery applications. An example of hyperspectral data in electron microscopy is shown in Fig. 7. An electron probe is rastered over a sample under study and diffraction patterns are captured on a pixelated detector. The pixelated detector captures many images as the electron probe is scanned across the sample. Emerging multimessenger astronomy applications further emphasize the utility of hyperspectral data representations combining observations from a wide array of detectors and telescopes.

3.1.5 Time-series data

Time-series data is common in experiments that observe dynamically evolving systems in processes such as synthesis for material discoveries or the temporal evolution of the plasma state in nuclear fusion experiments. It can be a measurement of high-speed temporally resolved imaging in material science or physics features (density, temperature, current, radiation, fluctuations, etc.) or spatial features of evolving

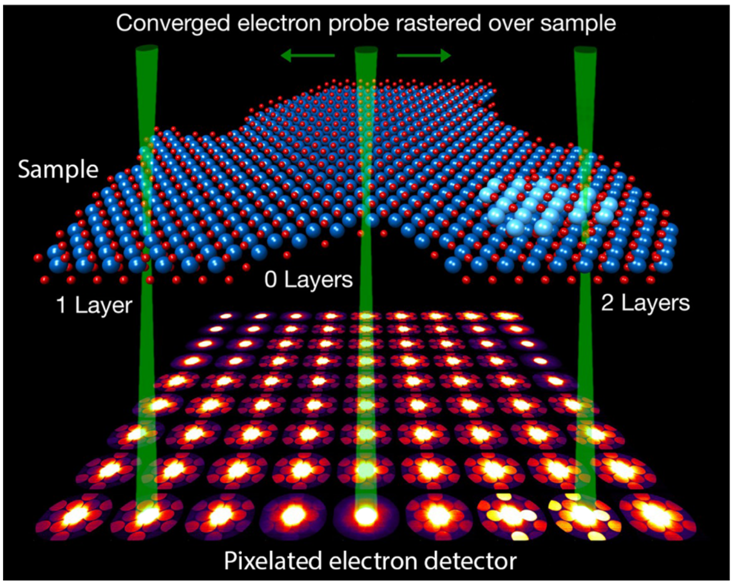

Figure 7. Experimental 4D-STEM measurement of a dichalcogenide 2D material. Atomic map is inferred from the data, each diffraction pattern represents an average of 7 × 7 experimental images, green STEM probes are labeled for regions of the sample with one layer, vacuum, and two layers [312].

Fig. 1. Experimental 4D-STEM measurement of a dichalcogenide 2D material. Atomic map is inferred from the data, each diffraction pattern represents an average of 7 × 7 experimental images, green STEM probes are labeled for regions of the sample with one layer, vacuum, and two layers.

plasma state, as a function of time. In-situ diagnostics of the time-series data can either provide alerts to terminate an experiment early that indicates undesired outcome in material science without performing the entire experiment and offline analysis that is time-consuming and computationally expensive, thus improves the experiment operation efficiency and accelerates discoveries of material of desired properties. This is illustrated in Fig. 8 for accelerator controls at the Fermilab Booster accelerator. In this application, magnet voltages that steer proton beams around a synchrotron are recorded at 15Hz time samples. This study builds a digital twin which is used to simulate the Booster data. Furthermore, to reliably predict and avoid large-scale major disruptions in nuclear fusion experiments, real-time analysis of the time-series data is crucial in guiding the action needed in experimental prediction and control.

3.2 System constraints

In this section, we present an overview of desired system properties and constraints that are prevalent across a number of application domains. Unique challenges are arising from each scientific application based on sensing technology, the physical processes, and the timescales and data rates, and bandwidth. These system constraints result in specific choices of data processing platforms, often with multiple compute architectures across the data continuum, such as the choice of FPGA-based systems versus embedded processors, GPUs, or custom ASICs. Table 2 summarizes several scientific application domains along with their event rates, system latency constraints and performance requirements, and deployment characteristics. We broadly define platforms for integration fast machine learning-techniques into "soft", software programmable coprocessors, and "custom", custom embedded computing devices. Software-programmable systems are often preferred because they are less complex to implement while custom embedded solutions are required when software programmable systems cannot satisfy experimental throughput, bandwidth, or latency constraints. We will describe in further detail this distinction below. Examples of these system design choices are the trigger systems for HEP include LHC reconstruction of collision events, the Belle-II experiment, the Mu2e experiment which deploy custom embedded systems. Meanwhile, experiments like the Electron-Ion Collider have data rates that may not require custom hardware

example in Jarausch et al. (2009), Florea et al. (2012), Gatis et al. (2014), and Midgley & Thomas (2014). Related terms for images of STEM diffraction patterns in common use from the literature include "convergent beam electron diffraction" (CBED), "microdiffraction," "nanodiffraction," "diffraction imaging," and "diffractogram," all of which refer to diffraction images of a converged electron probe. The term "ronchigram" is named for the "Ronchi test" for measuring aberrations of telescope mirrors and other optical elements, developed by Ronchi (1964). STEM probe diffraction measurements of aberrations using periodic objects were introduced by Cowley & Spence (1979) and were referred to as ronchigrams by Cowley (1986). Today the term usually refers to a diffraction image that is nearly in focus, typically recorded from an amorphous material.

Some of the earliest experiments that could be classified as 4D-STEM in the sense of this paper were those performed by Zaluzec (2002) to measure the Lorentz deflection. Zaluzec referred to this method as "position resolved diffraction" (PRD) in accordance with earlier work where 2D diffraction patterns were recorded over a line scan. The term PRD is more often found in the X-ray diffraction literature, but can still be found in the TEM literature, for example in Cheng et al. (2016). The similar term "spatially resolved diffractometry" was also used by Kimoto & Ishizuka (2011), which they used to refer to virtual imaging in 4D-STEM. The term "momentum-resolved STEM" is also used by some authors, for example Müller-Caspary et al. (2018a).

Perhaps the most common alternative name for a 4D-STEM measurement in diffraction space is "scanning electron nanodiffraction" used by Tao et al. (2009), Liu et al. (2013), Gallagher-Jones et al. (2019), and many others. A similar descriptor used in many studies is "nanobeam electron diffraction" (NBED), used for example by Clément et al. (2004), Hirata et al. (2011), and Ozdol et al. (2015). The term "pixelated STEM" can also be found in the literature, for example in MacArthur et al. (2013). In addition to referring to pixelated STEM, Hachtel et al. (2018) also introduced the term "universal detector" to refer to virtual imaging in 4D-STEM.

One 4D-STEM application discussed extensively below is crys-

method is called "automate
(ACOM), for example in Schw
et al. (2011), Kobler et al. (201
Conventional STEM detecto
usually have an annular (ring
imaging modes include bright
aligned with all or part of the
field (ABF) where a circle is rem
tor, and annular dark field (AD
of electrons scattered outside o
common STEM imaging mod
which records only the incohe
scattering (TDS) electrons,
(Pennycook & Nellist, 2011).
In this manuscript, we have
4D-STEM in order to include
probe is recorded in real spac
2006; Zaluzec, 2007; Etheridge

Detector Development

The rise of popularity for 4D
linked to the availability of high
nology. Conventional STEM d
HAADF record only a single va
segmented detectors with 4–16
measurements (Haider et al., 19
detector configuration recording
coupled device (CCD) with digi
lator, such as in Fan & Ellism
These detectors have good elec
readout speeds limited to video
dynamic range. This makes CCD
imaging, which requires readout
probe scanning rate (μs to ms tim
high-intensity signals such as th
such as the high-angle scattered
There are two primary ro
suitable for 4D-STEM applica
monolithic active pixel sensors
metal–oxide–semiconductor (C
epitaxial layer. When high en
layer, many low energy electr
toward sensor diodes where t
using CMOS electronics, as de
Dierickx et al. (1997), and M
electron detectors have seen w
commercialized by several com
(2016). See McMullan et al. (2
son. APS detectors have very h
speed, but relatively poor dyn
imaging, single "electron count
recorded with APS detectors (L
pixels and relatively low electr
electron density recorded in ea
electrons per pixel per frame, s
zation of individual electron st
electron counting can maxim
experiments, see Gallagher-Jon
that because the design of APS

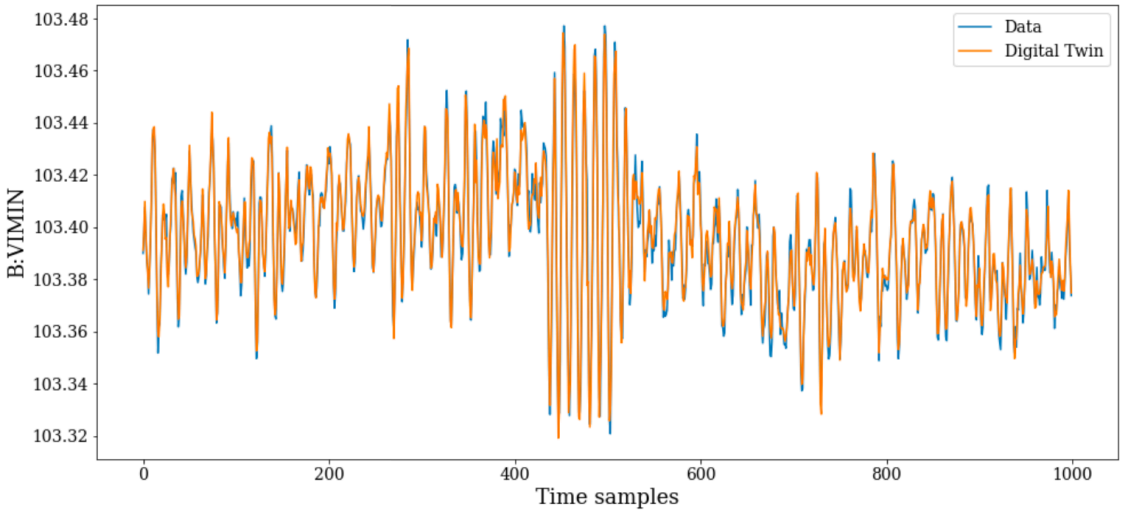

Figure 8. Selected test data (blue) versus prediction values (orange) from the Booster LSTM surrogate model for the Booster proton synchrotron complex [313].

Table 2. Domains and practical constraints: systems are broadly classified as soft (software-programmable computing devices: CPUs, GPUs, and TPUs) and custom (custom embedded computing devices: FPGAs and ASICs)

Domain	Event Rate	Latency	Systems	Energy-constrained
Detection and Event Reconstruction				**No**
LHC & intensity frontier HEP	10s Mhz	ns-ms	Soft/custom	
Nuclear physics	10s kHz	ms	soft	
Dark matter & neutrino physics	10s MHz	μs	Soft/custom	
Image Processing				
Material synthesis	10s kHz	ms	Soft/custom	
Scanning probe microscopy	kHz	ms	Soft/custom	
Electron microscopy	MHz	μs	Soft/custom	
Biomedical engineering	kHz	ms	Soft/custom	Yes (mobile settings)
Cosmology	Hz	s	soft	
Astrophysics	kHz–MHz	ms-us	Soft	Yes (remote locations)
Signal Processing				
Gravitational waves	kHz	ms	Soft	
Health monitoring	kHz	ms	Custom	Yes
Communications	kHz	ms	Soft	Yes (mobile settings)
Control Systems				
Accelerator controls	kHz	ms–μs	Soft/custom	
Plasma physics	kHz	ms	Soft	

solutions and could deploy only software programmable solutions for event reconstruction and real-time processing experiments. One final distinction worth discussing concerns the nature of real-time processing and the in-situ versus post-mortem nature of the inference and analysis tasks. Examples that we consider in classifying tasks that have different requirements are: data reduction which primarily focuses on limiting data collection rates of experiments for offline analysis; real-time processing and data analysis which is required to extract real-time domain features of the data for tasks like filtering/triggering; and closed-loop controls where data processing provides direct feedback to the operation and continuous control of an experiment. These distinctions and their consequences on the computing systems is illustrated in Table 3

Table 3. Classification of domains and their system requirements with respect to real-time needs.

Domain	Real-time data reduction	Real-time analysis	Closed-loop Control
Detection/Event Reconstruction			
LHC	Yes	Yes	No
Nuclear Physics	Yes	No	No
Dark Matter - Neutrino	Yes	No	No
Image Processing			
Material Synthesis	Yes	Yes	Yes
Scanning Probe Microscopy	Yes		
Electron Microscopy	Yes		
Biomedical Engineering	Yes		
Cosmology	Yes	No	No
Astrophysics	Yes	No	No
Signal Processing			
Gravitational Waves	Yes	No	No
Health Monitoring	Yes	Yes	Yes
Communications	Yes	Yes	Yes
Control Systems			
Accelerator Controls	Yes	Yes	Yes
Plasma Physics	Yes	Yes	Yes

3.2.1 Software programmable coprocessors

Historically, the first attempts at addressing the computational needs of the problems reviewed in this article have been through software-programmable systems. CPU-based local clusters or cloud services as well as cloud computing resources utilizing GPU or TPU-based hardware accelerators are utilized in different applications. One particular concept explored by the HEP community is the GPU as a Service (GPUaaS) model [314]. This can further be expanded into the Machine Learning as a Service concept, similarly explored within HEP [315]. These paradigms involve the implementation of machine learning modules to solve a set of physics problems, which are then transferred to GPU or TPU accelerators and accessed by the local CPU "client" of the native experimental system.

One of the major system constraints is the computational capacity, which can be defined in terms of a number of floating point operations as far as neural network implementations are concerned. Real-time machine learning methods require an ever-increasing rate of computational capacity as it directly impacts the *latency per task*. The *task* could be a trigger for LHC, reconstruction of an event in accelerator experiments or astrophysics, material synthesis, reconstruction of an image captured by an electron microscope, etc. Extreme parallelism would be desired to provide the highest capacity possible to minimize latency and maximize throughput. In a processor-based system, this can be addressed by increasing the size of the compute cluster. Naturally, facility costs impose a limit on the scale of these clusters. Another constraint is the available amount of storage coupled with the cost of data movement across the memory hierarchy. In the majority of the use cases, the latency involved with moving data from the front-end (detectors, microscopes, sensors, etc.) dominates the total latency. One of the prominent performance constraints is related to the utilization and subsequent latency of the network that links the front-end with the back-end. Current limitations on the speed of data movement renders the CPU/GPU cluster-based systems unable to meet the real-time requirements.

3.2.2 Custom embedded computing devices

As the latency and throughput constraints are coupled with challenging practical energy constraints, efforts have been directed towards specialized computing systems to address the hard real-time needs. An increasingly attractive paradigm is to design components that are finely optimized for specific steps in the data capture workflow. These components can be mapped onto FPGA devices or they can be designed and manufactured as an application-specific integrated circuit (ASIC). In the LHC and accelerator domains, there is a rich set of FPGA-based demonstrations of front-end data processing systems, which meet microsecond latencies. These systems are in charge of tasks such as triggering, event reconstruction, and anomaly detection. Direct and naive implementations of neural networks to perform inference for these tasks can fail to meet the latency requirements since they often incur significant resource utilization. The highest achievable FPGA clock frequency and inference latency is correlated with the resource utilization and percentage occupancy of the device. Co-design techniques developed for these applications particularly specialize in extreme quantization and pruning (with an awareness of accuracy) so that resource requirements can be controlled aggressively to ensure inference latency targets. These optimizations push the resource usage envelope as far as down as 10s of percent of the FPGA device in order to meet the system constraints and yet demonstrate implementations with high inference accuracy.

Some other applications (e.g., accelerator controls, biomedical and health applications) impose less stringent latency expectations, in the order of ms, where the urgency for resource minimization is alleviated. Hence, the focus of the system design can shift from extreme resource economy to enhanced sophistication in the algorithms that are being mapped to the device. Inference models can now include deep(er) learning models coupled with advanced video and signal processing engines, as well as local privacy-preserving processing tasks (applicable particularly to mobile health and networking and communication applications).

For mobile and IoT-based deployment of the edge devices, resource efficiency emerges as an important factor as it impacts energy consumption. However, in these applications, energy efficiency can also be achieved by alternative means. One option would be selective powering, i.e., creating a resource-rich full-featured baseline implementation, which still comfortably meets latency constraints if energy was not an issue, and introducing power gating or standby features to modulate energy consumption during periods of low/no activity.

There are system constraints, which point the designers to a custom ASIC solution in addition to or in place of FPGA devices. ASICs can address extreme form factor considerations, integration of computation with sensing (e.g., smart photon detectors) into compact front-end devices, tight integration with other mixed-signal or analog functionalities, radiation hardening requirements, and ultra-low energy budgets.

4 TECHNOLOGY STATE-OF-THE-ART

In this section, we aim to give an overview of technologies and techniques for building fast ML algorithms. This requires *codesign*: building algorithms with hardware in mind and providing efficient platforms for programming the hardware. Section 4.1 and Section 4.2 focus on neural network design and training for efficient implementation in hardware. In Section 4.3 and Section 4.5, we classify our discussion of ML hardware compute platforms into two categories: "Conventional CMOS Hardware" and "Emerging Beyond CMOS Hardware." The former will address nearer-term hardware solutions, while the latter will focus on the speculative end of the spectrum. Meanwhile, because the area of programming new hardware is rapidly moving, we lay out an example of the options and challenges for one device family: FPGAs. This is presented in Sec. 4.4, and from the details for FPGAs we hope the reader also gets a sense of the fundamental approaches for designing software for emerging hardware.

4.1 Systematic Methods for the Efficient Deployment of ML Models

As discussed in Section 2, many ML problems in science require low latency, often with constrained resources. However, most of the current state-of-the-art NN models have prohibitively high latency with a large memory footprint and energy consumption. For this reason, practitioners have been forced to use sub-optimal models (e.g. shallow NNs) with non-ideal accuracy to avoid this latency problem. There is a large body of literature that has focused on solving this problem by making NN models more efficient (in terms of latency, memory footprint, and energy consumption). These efforts could be broadly categorized as follows: (i) Designing new efficient NN architectures; (ii) NN and hardware co-design; (iii) Quantization (low precision inference); (iv) Pruning and sparse inference; and (v) Knowledge distillation. Here we briefly discuss each of these approaches.

Designing new efficient NN architectures

One line of research has been focused on finding new NN models that are efficient by design. A notable early work is SqueezeNet [316], a new NN model without any expensive Fully Connected layers, along with a new lightweight *Fire module*, that resulted in a $50\times$ smaller model as compared to AlexNet, but with the same accuracy. Later on, several new innovations were made in efficient NN architecture design. One focus has been to find efficient layers/operators. Notable works are group convolutions [317], depthwise convolutions [318], spatial separable convolutions [319], shuffle layers [320], and shift convolutions [321], to name a few.

Another focus has been to find similar substitutes to *Fire module* that are more efficient and result in better accuracy/generalization. Notable works include residual networks [322] (originally designed to solve issues with vanishing gradients, but these structures are generally more efficient than non-residual architectures), densely connected networks [323], squeeze-and-excite modules [324], and inverted residual blocks [325].

These classical techniques mostly found new architecture modules through a manual design search. This is not scalable, and as such recent approaches have proposed automated methods that use neural architecture search (NAS). NAS methods automatically find the right NN architecture for a given constraint of model size, depth/width, and/or latency. The high-level approach here is to train a probabilistic *SuperNet* that includes all possible combinations of NN architectures within the prescribed constraints, but with learnable probabilities. After this SuperNet is trained, one can sample an architecture from its learned probability distribution. Notable works include RL based methods [326], efficient NAS [327], MNasNet [328], DARTS [329], and Differentiable NAS [330].

NN and hardware co-design

Another promising line of work has been to tailor the NN architecture for a specific hardware platform, and/or co-design them together. This is quite promising for configurable hardware such as FPGAs. The importance of hardware-aware NN design is that the cost of performing different types of operations varies for different hardware. For example, hardware that has a dedicated cache hierarchy can execute bandwidth bound operations much more efficiently than hardware without a cache hierarchy. Notable works in this area include SqueezeNext [331], where both the NN and the hardware accelerator were co-designed with a manual tuning approach. More recent works have proposed to automate hardware-aware design through NAS. Notable works include ProxylessNAS [332], OnceForAll [333], FBNet [330], and MobileNetV3 [334].

Quantization (low precision inference)

A common solution is to compress NN models with quantization [335–348], where low bit-precision is used for weights/activations. A notable work here is Deep Compression [349], which used quantization to compress the model footprint of the SqueezeNet model discussed above, bringing its size to 500x smaller than AlexNet. In quantization, the model size is reduced without changing the original network architecture, and it could potentially permit the use of low-precision matrix multiplication or convolution. Therefore, both the memory footprint and the latency could be improved.

The quantization methods can be broadly classified into two categories of *Post-Training Quantization* (PTQ), and *Quantization-Aware Training* (QAT). In PTQ, a pre-trained model in single precision is quantized to low precision without any fine-tuning or re-training [350–357, 347, 358]. As such, these quantization methods are typically very fast, and, in some cases, do not even require any training data [347, 359, 357]. However, PTQ often leads to high accuracy degradation, especially for low precision quantization. To address this, some quantization methods adopt QAT to re-train the model after the quantization, so that the parameters can get adjusted. This approach often results in higher accuracy, but at the cost of longer time associated with re-training the model [360, 361, 336, 337, 339, 362, 340, 363–366].

Another differentiator is the use of *simulated quantization* (aka fake quantization), versus *integer-only* quantization [341, 367–369]. In the former, the weights/activations are stored in low precision, but they are cast to higher precision during inference. In the latter, there is no casting involved, and the multiplication and accumulation also happen in low precision. Using integer-only quantization has the advantage that one can speed up inference by using low-precision logic for multiplication and addition, besides reducing the memory footprint of the model.

Another distinction is *hardware-aware quantization*. Similar to NN architecture design, quantization can also be tailored for specific hardware platforms. This becomes important for mixed-precision quantization [370–376, 367]. The reason is that certain operations in the NN model may benefit more from low precision quantization than others, based on whether they are bandwidth bound or compute-bound. As such, as schematically illustrated in Figure 9, one must determine the best precision setting based on the tradeoff between the potential footprint/latency gain and the sensitivity to accuracy degradation.

Pruning and sparse inference

Another approach reducing the memory footprint and computational cost of NNs is to apply pruning, which could be thought of as quantization to 0-bits. In pruning, neurons with small *saliency* (sensitivity) are removed, which results in a sparse computational graph [377]. Here, neurons with small saliency are those whose removal should minimally affect the model output/loss function. Pruning methods can be broadly

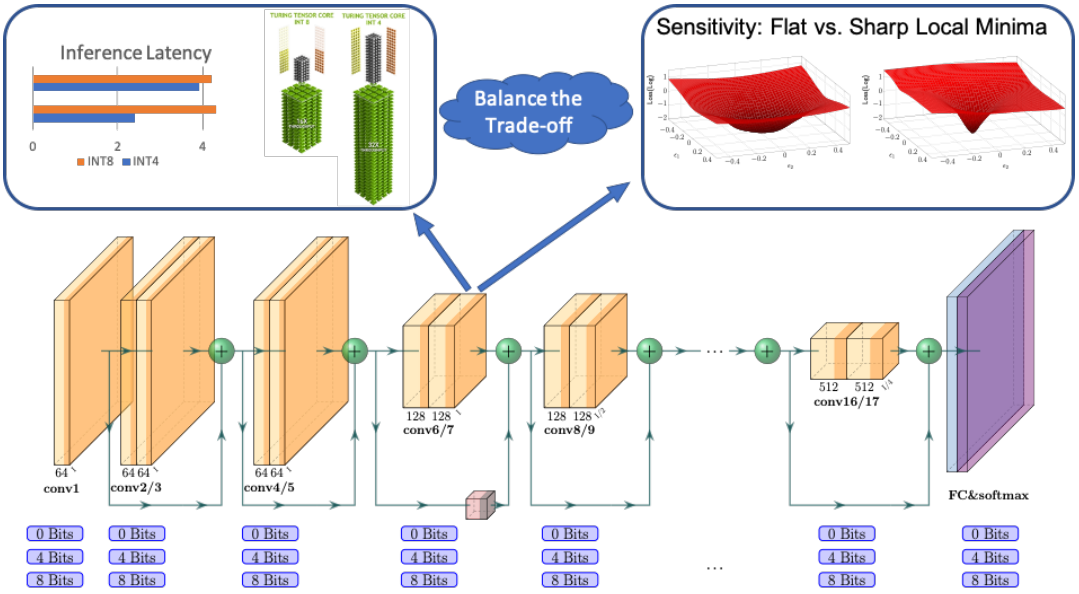

Figure 9. The illustration of hardware-aware quantization and pruning. A given NN model can be compressed by using low precision quantization instead of single precision. The extreme case is to use 0-bit quantization which is equivalent to removing/pruning the corresponding neurons. The goal of compression is to find the best bit-precision setting for quantization/pruning to reduce model footprint/latency on a target hardware with minimal generalization loss.

categorized into unstructured pruning [377–382], and structured pruning [383–388]. Unstructured pruning removes neurons without any structure. With this approach, one can remove most of the NN parameters with little impact on the generalization performance of the model. However, this approach leads to sparse matrix operations which are hard to accelerate and are typically memory-bounded [389–392]. This can be addressed with structured pruning, where a group of parameters (e.g., an output channel) is removed. However, the challenge here is that high degrees of structured pruning often lead to significant accuracy degradation.

In both approaches, the key question is to find which parameters to prune. A simple and popular approach is magnitude-based pruning [393–400]. In this approach, the magnitude of parameters is used as the pruning metric. The assumption here is that small parameters are not important and can be removed.

An important problem with magnitude-based pruning methods is that parameters with small magnitudes can actually be quite sensitive. It is easy to see this through a second-order Taylor series expansion, where the perturbation is dependent on not just the weight magnitude but also the Hessian [377]. As such there are several works that use second-order based pruning [377, 401, 378, 402, 403].

Finally, we should mention that it is possible to combine pruning and quantization together to compress the NN model. In fact, pruning could be viewed as quantization to 0-bits. The recent work of [358] proposes a quantization-aware pruning method and applies to high energy physics problems; It reports better results than pruning or quantization alone.

Knowledge distillation

Model distillation [404–411] trains a large model and then uses it as a teacher to train a compact model. Instead of using class labels during the training of the student model, the key idea of model distillation is to leverage the soft probabilities produced by the teacher, which can guide/help the student training.

Previous methods of knowledge distillation focus on exploring different knowledge sources. Refs. [405, 407, 412] use logits (the soft probabilities) as the source of knowledge, while Refs. [404, 408, 410] try to leverage the knowledge from intermediate layers. The choices of teacher models are also well studied, where Refs. [413, 414] use multiple teacher models to jointly supervise the student model, while Refs. [415, 416] apply self-distillation without an extra teacher model. Other previous efforts apply knowledge distillation with different settings on different applications. Refs. [417, 418, 411] study data-free knowledge distillation, and Refs. [419, 420] combine knowledge distillation with GANs.

A major challenge of knowledge distillation methods is to achieve a high compression ratio. Compared to quantization and pruning which can usually maintain accuracy at $4\times$ compression, knowledge distillation methods tend to have non-negligible accuracy degradation at those compression levels. But these two approaches are orthogonal, and recent works have shown that their combination can result in high accuracy/compression [409, 406, 367, 421]. It should be mentioned that current distillation methods are mostly applied to classical ML problems, and few works have looked into their application in Science AI problems.

4.2 Systematic Neural Network Design and Training

There is currently no analytical approach to find the right NN architecture for a given task and training dataset. Originally, designing the NN architecture was mostly a manual task with intuitions that were often ad-hoc. However, in recent years there has been a lot of innovations in automating the NN architecture design process, which is referred to as Neural Architecture Search [326–330, 332, 333].

NAS could be viewed as a hyperparameter tuning problem, where the hyperparameters are the design choices for a NN architecture. This could include width, depth, types of operations, etc. The main challenge is that the search space for the operation types scales exponentially with the number of layers. As such, one has to still include some high-level intuition about the NN architecture to limit the search space.

After limiting the search space, the general NAS process is as follows: A candidate architecture is sampled from the set of all possible architectures and is then trained for a number of epochs on the training dataset. The accuracy is then used as the metric to evaluate how good that candidate architecture is. Then based on this reward, the probability distribution of sampling architectures is updated. This process needs to be repeated for many different candidate architectures (sometimes exceeding hundreds of thousands). Inherently, this leads to another problem related to tuning the optimization hyper-parameters for each candidate architecture. For example, if a good architecture is sampled from the NAS but is trained with sub-optimal hyperparamters, then the error will be high and the NAS algorithm will reduce the likelihood of sampling that architecture which is not the desired property.

As a result, *scalability* has become an integral concern for any procedure in the presence of "big data." One main class of procedures for which scalability has become indispensable is in numerical optimization algorithms, which are the core of training methods. There is a large body of literature on designing efficient numerical optimization/training methods [422–424, 382, 425–430] as well as efficient NAS algorithms to search for the right NN architecture [326–330].

For the optimization, the goal is to design new methods that require fewer iterations to converge and are more robust to hyper-parameter tuning. One notable advancement here is the ability to apply second-order methods without the need for forming the second-order operator [428, 431, 430, 422]. It has been shown that the performance and robustness of these methods are higher than first-order optimization methods on classical ML problems (e.g. in computer vision or natural language processing). Interestingly, some

recent results for Physics Informed Neural Networks (PINN) [432] have found that first-order methods work significantly sub-par to (quasi) second-order methods. This could potentially provide opportunities to adapt or redesign some of the second-order algorithms for Science problems.

For the NAS algorithms, the goal is similar, which is to find methods that require evaluating fewer candidate architectures, with less manual restriction or tuning of the search space. Another goal is to design transferable NAS algorithms that can be trained on a small problem and then transferred to larger problems that are more expensive [332, 333].

In summary, the core of designing NN architecture is to have a fast method of sampling architectures (through NAS), and the fast training of the sampled architectures (through fast and robust optimization algorithms).

4.3 Hardware Architectures: Conventional CMOS

As the prevalence and demands for machine learning rapidly continue to grow, it is increasingly important that we design machine learning algorithms efficiently and simultaneously deploy them on complementary and powerful hardware platforms. The compute and memory demands of NN deployments are huge and growing beyond the limits to where standard silicon-based semiconductors can scale. The reasons behind the scalability challenges in the semiconductor industry are as follows: Firstly, as we approach the End of Moore's Law, transistor cost has been exponentially rising due to rising chip design costs with shrinking technology nodes (as published by Xilinx and Gartner in 2011 already [433]). Furthermore, with the end of Dennard scaling, we've encountered considerable thermal challenges as power density no longer remains constant between node generations. To mitigate the challenges of increasing thermal density, chips are now designed to conditionally deliver power to groups of transistors, effectively throttling or "turning off" parts of a chip. This technique has come to be known as creating dark silicon [434].

To overcome these challenges and provide sufficient compute capabilities, many disruptive approaches have been proposed. For example, Cerebras Systems [435] has brought to market the first computer system which employs **wafer scale integration**. where chips are built from complete wafers rather than individual dies. Such a technique brought with it substantial engineering challenges in regards to power delivery, packaging, and cooling. Exploring the other dimension, foundries are investigating true **3D chip stacking** as was presented at HotChips'2019 by TSMC [436]. Even **analog computing** [437, 438], **quantum computing** [439] and **in-memory computing** [440, 441] are investigated as well.

Less risky approaches focus on moving away from traditional von Neumann architectures, using specialization of compute architectures to provide the necessary performance scaling and energy efficiency. Due to the specialization, the devices become increasingly heterogeneous. A huge range of devices has emerged that all try to address this problem in different ways, whereby the key challenge is: How do we loop transform and unfold the algorithms best to maximize data reuse and compute efficiency, minimize memory bottlenecks, and limit power consumption while meeting real-time requirements?

The choice of hardware type and quantity often boils down to a set of constraints imposed by compute environment (datacenter, cloud, on-premise, edge, mobile), workload type (inference, training), data type (Language, Time Series, Vision, Graph, etc), ML model, usage model (online inference, batch jobs), and user-centric Service-Level Agreements (encryption level, request latency, etc). For large datacenter deployments handling various types of workloads, it is often the case that several platforms must be combined to reduce Total Cost of Ownership (ToC) across all their hardware platforms. It has therefore

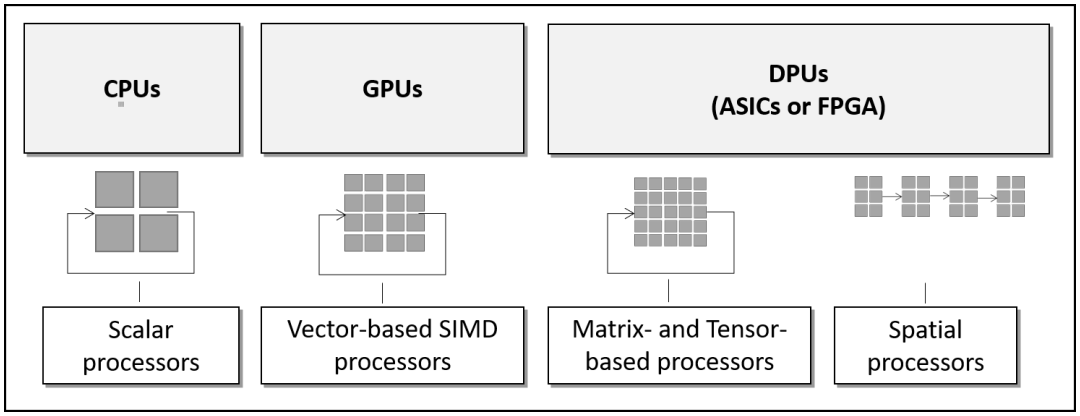

Figure 10. Taxonomy of compute architectures, differentiating CPUs, GPUs and DPUs

become increasingly necessary for owners of heterogeneous platforms to think of their systems as large-scale multi-processor computers, a trend sometimes termed Warehouse Scale Computing [442]. For Deep Learning hardware accelerators, these new computers generally take the form of CPU co-processors: a host CPU communicates with other entities in the datacenter, interfaces with disk memory, and formats input data which is then offloaded to the accelerator responsible for executing a user-defined compute graph, or Neural Network.

We begin with a taxonomy of these hardware architectures and discuss their relevant characteristics when it comes to the acceleration of machine learning workloads. This is essential to understand how they will differ in their execution behavior, what it takes to leverage their unique features and how they can potentially benefit from previously introduced optimization techniques.

Taxonomy of Compute Architectures for Deep Learning

A broad range of hardware architectures to deploy machine learning algorithms exists today. We can broadly classify them by the following criteria:

1. Basic type of compute operation
2. Inherent support for specific numerical representations
3. External memory capacity (which is mostly relevant for training workloads) [3]
4. External memory access bandwidth
5. Power consumption in the form of thermal design power (TDP)
6. Level of parallelism in the architecture and the degree of specialization

As is shown in Figure 10, we classify the compute architectures into scalar processors (**CPUs**), vector-based processors (**GPUs**), and so-called deep learning processing units (**DPUs**), although realistically these categories blend to some degree. DPUs are specialized for this application domain whereby we distinguish the more generic matrix- or tensor-based processor and a spatial processing approach. DPUs can be implemented with either ASICs or FPGAs. All of these architectures will be discussed individually below.

[3] In these comparisons, we treat HBM and HBM2 as external memory as it is used in the same way as DDR4 or GDDR memory.

CPUs

CPUs are widely used for ML applications and are viewed as largely serial or scalar compute engines (even though high-end variants for cloud deployment may have up to 10s of cores). They are optimized for single-thread performance, with implicitly managed memory hierarchies (with multiple levels of caches), and support floating point operations (FP64 and FP32) as well as 8bit and 16bit integer formats with dedicated vector units in most recent variants. Theoretical peak performance tops at 6.8TOPs for FP64 assuming boost clock speed (Cascade lake, 56 cores, 3.8GHz). External memory is currently primarily leveraging DDR4 memory banks with large capacities: Intel's Cascade Lake offers up to 4.5 TebiByte (2^{40} Bytes) which is beyond what any of the other device categories can offer. Access is at maximum speed through high-end hardened memory controllers, offering 282 Gbps bandwidth (for example Cascade Lake with 12 DDR4 channels). Compared to GPUs and other HBM-enabled devices, the memory bandwidth of CPUs is lower. However, for many use cases, this can be compensated through their sophisticated cache hierarchies, combined with mature compiler tools. Regarding power consumption, CPUs are at the upper end of the spectrum with high-end devices range up to 400 W [443]. In the embedded space, ARM processors provide generally popular solutions, in particular when performance requirements are very low and when functionality is required that is not supported by the specialized device variants. In particular, the Ethos [444] family of processing cores is specialized for CNN workloads and as such is considered under the DPU category below. The advantages of CPUs are the generality of the hardware, as well as the ease of programming where design environments have matured over decades. As expected this comes at the cost of lower peak performance and less efficiency compared to the more specialized device families. In regards to quantization, CPUs can only leverage this optimization technique for INT8 and INT16 if supported.

GPUs

GPUs are SIMD-based (Single Instruction, Multiple Data) vector processors that support smaller floating point formats (FP16) natively, as well as fixed point 8-bit and 4-bit integer formats more recently, and have a mix of implicitly and explicitly managed memory. NVIDIA GPUs are some of the most popular hardware targets for machine learning, and newer families of chips have been introduced to specifically accelerate this workload, with AMD not far behind. The latest devices in NVIDIA's Volta and Turing architecture families, introduced in 2018 and 2019 respectively, offer up 130TOPs in FP16, which is beyond the capabilities of the latest CPU generations. As such they are amongst the highest performant devices in the market for the acceleration of DNNs as they can exploit the high degree of parallelism inherent in this application via increasingly specialized architectural features. For example, NVIDIA's Volta is the first generation to incorporate tensor cores as a new feature, as well as improved FP32 and FP64 support for training in a data center setting [445], and also introduced a deep learning accelerator (DLA) in their embedded devices to further reduce power consumption. This specialization brings additional challenges for their usage; there are up to 3 distinct execution units now, namely CUDA cores, tensor cores, and the DLA, which don't operate on the workload simultaneously (at least not easily or by default). We, therefore, don't sum up the peak performance of different execution units, but use only the maximum. AMD announced the Vega GPU [446] with new deep learning instruction set operations, with the goal of obtaining parity with NVIDIA's high-end Tesla V100 datacenter GPUs. Also, AMD's most recent EPYC family supports customized instructions for deep learning [447]. Both companies offer also low power GPUs for the embedded space, namely the AMD Vega mobile GPU [448] and NVIDIA Jetson TX2 [449] and AGX family [450].

In regards to memory, GPUs leverage specialized and highly pipelined GDDR memory, which reduces capacity, but offers much higher bandwidth (up to 732GBps). With NVIDIA's Turing family the latest devices include HBM2 DDR memory stacks [451], which scales the memory access bandwidth to 1TBps and beyond. Again this is particularly important to address the needs of training workloads. For the same reason, some of the DPUs introduce HBM2 as well, as discussed below. In regards to power consumption, GPUs are high, up to 345 W.

One general challenge for GPUs is that they need to leverage input parallelism to achieve high utilization of their large compute arrays. Therefore before execution inputs need to be grouped into batches, which has adverse effects on end latency. Further, GPUs are relatively high in power consumption. Regarding quantization, support is limited to the inherent datatypes, which are INT4 at smallest in the context of NVIDIA's Turing family, and INT8 for many of the others. Finally, the corresponding software environments for GPUs, while not on the same level as CPUs, have matured significantly and provide increased ease of use.

FPGAs and ASICs

FPGA and ASIC customize hardware architectures to the specifics of a given application. They can be adapted in all aspects to suit a use case's specific requirements. This includes their IO capability, their functionality, or even to suit specific performance or efficiency targets. FPGAs can be reprogrammed whereas ASICs are fully hardened. This flexibility allows for amortizing the design costs of the circuit across many applications but comes at the expense of hardware resource cost and performance.

FPGAs are a popular choice for the acceleration of CNNs. Traditionally, an FPGA compute fabric consist of a sea of lookup tables (LUTs) which are interconnected through a programmable interconnect. The latest generations host millions of LUTs. Furthermore, the fabric is interspersed with specialized hardened compute blocks (DSPs) which accelerate n-bit multiply accumulate operations (MACs), as well as SRAM blocks. The latter are referred to as block RAMs (BRAMs), which hold 36 kbits, and Ultra RAMs (URAMs) which store 288 kbits. More recent FPGA generations combine multiple FPGA dies, referred to as super logic regions (SLRs), and leverage a silicon interposer to provide connectivity between SLRs. This technology is referred to as stacked silicon interconnect (SSIT) and helps scale device capacity.

DPUs

As mentioned at the beginning, the term DPU (short for deep learning processing unit) refers to a new type of compute architecture, specialized for the acceleration of CNNs. DPUs are customized for these types of applications in a number of ways: types of operations supported, direct support of tensors or matrices, inherent data types and supported numerical representations, macro-architecture, explicitly managed and specialized memory hierarchies, and which levels of parallelism they exploit (input, output pixel, IFM, OFM, bit, and layer and branch parallelism) as was introduced in the first part of this chapter. We differentiate two types of DPUs, which can be implemented with both ASIC technology and FPGAs.

Matrix of Processing Elements (MPE)

The first type, as shown on the left side of Figure 11, consists of an MPE that operates on matrices or higher dimensional tensors. The processing engines can be simple MACs, vector processors, or more complex VLIW (Very Long Instruction Word) cores that can support concurrent execution of different instructions. A popular example in this category is Google's Tensor Processing Unit (TPU). Introduced in 2016 [452], it was originally designed to accelerate Google's TensorFlow framework. The first generation supported integer arithmetic with a massively parallel INT8 matrix-multiply engine. The second generation

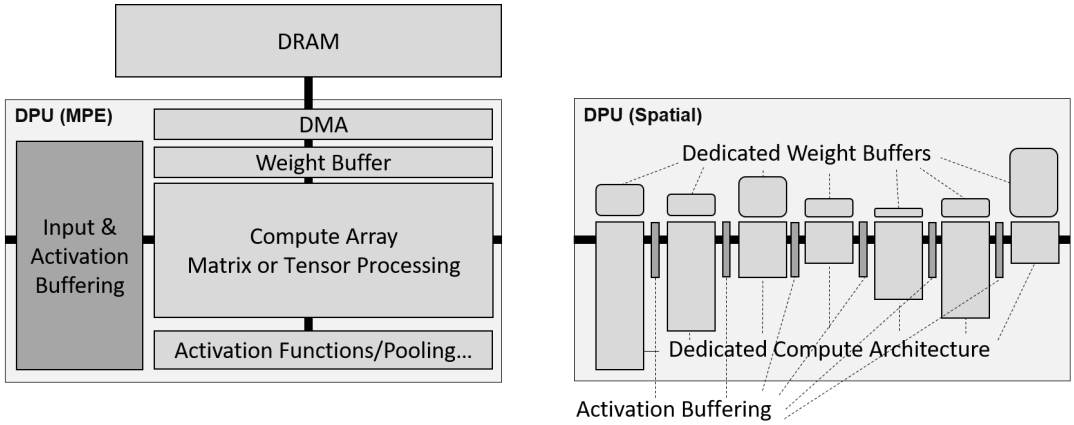

Figure 11. DPU architectures: Matrix of Processing Engines (MPE) on the left, and spatial architecture on the right

TPU was announced in May 2017 [453], and the third generation in May 2018 [454]. These newer chips boast improved memory performance as well as support for floating point specifically aimed at training. There are a number of startups introducing custom hardware that fall into this category. Within the cloud , there are Graphcore, Groq, and Wave Computing. Within the embedded space, where the design constraints are even more stringent, we find even more solutions. Most are secretive about the details of their designs. Intel is investigating several custom accelerators and has for that purpose acquired a number of startups, namely Nervana, Habana, and Movidius. Fathom [455] is Movidius' ultra low power Neural Compute Stick (NCS) which operates at about 1 W. Also, ARM offers specialized CNN processors in the form of their Ethos family, boosting performance up to 4TOPs with support for INT8 and INT16 datatypes.

As mentioned above, DPUs provide specialized datatypes to execute heavily quantized, reduced precision CNN implementations. At the extreme, binarized neural networks (which are very high throughput at extremely low power) are exploited in the following ASICs: BinarEye [456], BNN Custom Fabric [457], and IBM AI Accelerator [458]. Also, Lattice has announced binarized neural network libraries targeting low power FPGA and achieving 1 TOPs/W [459]. Custom floating point representations are also considered. For example, Microsoft's Brainwave project [460] uses this approach with the aim of applying FPGAs to CNNs at datacenter scale. However, typically the hardened versions in ASICs only support INT8, as lower precisions could potentially limit their application scope. FPGA-based MPE implementations such as Xilinx's xDNN are less constrained and in principle can be customized as needed.

Similar to the GPU, but perhaps to a lesser degree, DPUs leverage input, IFM (input feature map) and OFM (output feature map) parallelism, which requires buffering of inputs and may have adverse effects on latency as well. A particular challenge arises in the context of software environments, which differ for all vendors and are less mature than what we have observed for CPUs and GPUs. Typically, they are limited to support execution of very specific layer types (sometimes even restricted in regards to parameter ranges) and neural networks, whereby the range of layer types and neural network models is continuously expanding.

In summary, through their specialization, these implementations minimize hardware cost, maximize performance and optimize efficiency by exploiting specific precision arithmetic with a specialized

instruction set and customized memory system. However, in order to gain a performance advantage, the algorithms need to be adapted to leverage these features.

Spatial DPUs.

The second type of DPU leverages spatial acceleration and exploits layer and branch parallelism. Popular examples are hls4ml [461] and FINN [462, 463]. To that extent, the hardware architecture is even further specialized to the specifics of a given deep learning topology. This is visualized on the right side of Figure 11. The hardware architecture actually mimics the given deep learning topology and the inputs are streamed through the architecture. Every layer is instantiated with a dedicated compute datapath. Each layer has a dedicated weight buffer, and activation buffers in-between layers are FIFOs of minimal size. They buffer just enough data to feed the next set of convolutions in the next layer. This is substantially more efficient compared to the first type of DPUs or GPUs and yields reduced latency.

DPUs and GPUs generally perform a layer-by-layer compute, where a sequence of images has to be buffered in order to extract maximum compute out of the platform (input, IFM and OFM parallelism). For this, the device buffers a batch of images before computing the first layer of all images. Then all intermediate results are buffered, and then the next layer is computed, and so on. Hence the latency is heavily dependent on the size of the input batch.

As a result, spatial DPUs have an advantage in regard to latency. This level of customization is only possible with programmable hardware architectures such as FPGAs, as they can adapt the hardware architecture for different use cases. This generally wouldn't make sense in the context of an ASIC accelerator, as that would yield an ASIC only capable of accelerating one specific topology, which would be far too restrictive in scope. The limitation in spatial architectures is the scalability in the numbers of layers. Each layer comes at a resource cost overhead and there is a maximum number of layers that can be created within a single device. As a result, some extremely deep CNNs might not be able to fit into a single device. Microsoft's Brainwave project leverages spatial computing and overcomes this limitation with a distributed approach [460].

Once a spatial DPU has been leveraged and the architecture is specialized for a very specific CNN, the architecture can be further customized in regards to minimum precision. By supporting only the bits as needed per layer of the CNN they can achieve even higher performance and efficiency, while in an MPE, the hardware will support the maximum precision that is required over the whole network. In regards to customized precisions and spatial architectures, FINN has pioneered the first binarized neural network accelerators [462, 464] and provided many proof points for customized reduced precision implementations [463]. This flexibility comes at a cost, in the form of programming complexity, and they are extremely difficult to characterize in general, as the performance characteristics depend on the specifics of the hardware architecture that has been implemented.

Further Variants of DPUs

Beyond the previously discussed spatial DPUs and MPEs, there are many more variants. Some exploit sparse computing engines for example, such as EIE and its successor ESE [465], SCNN [466], Cnvlutin [467], Cambricon-S and Cambricon-X [468]. These are the only architectures that can benefit from irregular sparsity. Finally, another dimension for customization of precision is to optimize over the execution- or run-time of a CNN. In other words, beyond using statically fixed reduced precision, where the hardware operates with a fixed precision for all variables, some approaches explore run-time configurable bit precision which allows for the exploitation of bit-parallelism in the arithmetic. On the

hardware implementation side, this can be exploited with run-time programmable precision and is effective with **bit-serial** implementations. For example Umuroglu et al. [469] demonstrate with BISMO that bit-serial can provide highly attractive performance with minimal overhead on FPGAs, while Judd et al. show the same is true for ASICs with their prototype ASIC called Stripes [470]. While this concept can be applied to both MPE and spatial architectures, it makes the most sense for MPEs.

Server-class	Throughput	Latency	Power	Ext. Mem. Bandwidth	HW specialization	Ease of Use	Training/Inference
Conventional							
CPU	Medium	High	High	Medium	Low	High	Both
DPU-MPE	High	Medium-High	Medium	High	Medium	Low-Medium	Inference
DPU-Spatial	High	Low	Medium	High	High	Low	Inference
GPU (NVIDIA A100)	High	High	High	High	Medium	High	Both
Speculative							
Cerebras CS-1	Very High	Medium	High	Very High	Medium	Medium	Both

Table 4. Characterization of types of hardware based on important metrics.

Summary of Conventional CMOS Hardware Architectures

We analyzed three categories of hardware architectures that are leveraged for CNN inference, namely common CPUs, SIMD-based vector processors such as GPUs, and DPUs which are specialized architectures for the acceleration of deep learning workloads. An overview of the architectures is visualized in Table 4. Please note, "Ease of Use" includes compute kernel programmability as well as general ease of use. The degree of specialization includes operators, precision support, and customization towards topologies. In summary, for DPUs, we distinguish between tensor processors which leverage a matrix of processing engines and spatial architectures which can be further specialized for specific topologies using FPGAs. CPUs are the most general solution but high in power. GPUs and DPUs offer the highest performance, though GPU are more expensive in energy cost. Spatial DPU architectures excel at low latency and provide the highest compute efficiency through maximized customization. CPUs, GPUs, and DPUs (MPE) use a sequential layer-by-layer compute model whereas spatial DPUs execute all layers of the network concurrently. Hardened topologies in form of ASICs, CPU and GPU offer a fixed set of native dataypes, whereas FPGAs can adopt any precision and numerical representation, which provides the utmost flexibility and leverages optimization with quantization to the maximum, whereas hardened approaches need to default to the next higher supported precision into which the reduced precision variable can be embedded. However, the programmability in the FPGA fabric also comes at a speed and energy cost. All architectures can benefit from coarse-grained pruning optimization techniques. Only sparse execution engines can benefit from irregular pruning, such as synaptic pruning. We also discussed the various deployment options. Many devices offer different power and operating modes as different compromises between throughput and power consumption to adapt to the potentially very different optimization targets of different application settings. Similarly, batch sizes, thread counts and stream sizes offer another compromise in regards to throughput versus latency. Again this is to facilitate a spectrum of different use cases. Finally, the table shows that speculative approaches such as Cerebras can bring fundamental performance scalability. Overall, each approach comes with its own advantages and disadvantages and the best solution greatly depends on the specifics of a given use case.

4.4 Hardware/Software Codesign Example: FPGA-based Systems

In the last decade, we have observed the rise of two significant paradigms that have come to scientific applications: heterogeneous-computing systems and machine learning. Heterogeneous computing can overcome the decline of Moore's Law and Dennard Scaling and achieve the desired computational cost and performance by executing portions of the applications on the best-matched hardware, e.g., CPU, GPU,

ASIC, and FPGA. On the other hand, machine learning is an automatic process that creates programs that can solve classes of problems. As with traditional programming, machine learning can significantly benefit from heterogeneous computing; in addition, designers can tailor specialized but reprogrammable hardware to fit ever-changing machine learning requirements. This section examines tools and methodologies that can automatically deploy and orchestrate machine learning on FPGA systems in larger scientific applications. FPGAs are a particularly compelling example to explore because the efficiency of the hardware coupled with their programmability makes for an interesting case study in hardware/software codesign.

Traditional software programming is complicated, and parallel high-performance programming is even more challenging. Programming heterogeneous systems that integrate FPGAs bring the challenge to the next level: the programmer must deal with a multi-objective optimization problem that involves performance and costs, i.e., hardware resources. For machine learning applications, a common practice is to profile the application on CPU (or GPU) to identify the bottlenecks to be offloaded onto the reprogrammable logic to improve latency, throughput, or energy efficiency of the application as a whole. Then, part of the application can remain on the CPUs to control the execution and interact with the rest of the scientific setup.

FPGA Programming

FPGA are configurable integrated circuits that provide a good trade-off in terms of performance, power consumption, and flexibility with respect to other hardware paradigms. However, it is a challenging and lengthy task to program FPGAs. FPGA programming has traditionally been a job for hardware designers familiar with digital design and computer architecture. These requirements lead to a steep learning curve for software developers and other domain experts. In order to lower the entry barrier, there has been a growing focus on designing FPGA hardware at a higher level of abstraction. As a result, various approaches have brought FPGA development into the mainstream by allowing developers to design for FPGAs at a higher level using familiar languages such as C, C++, OpenCL, and in some cases, even C# [471]. Here an important question arises: what are the additional advantages of designing the hardware at a higher level of abstraction? High-level languages (HLLs) include various constructs and design patterns that are more functionally expressive. Furthermore, the amount of time spent in the verification of the design is also a crucial factor. Hardware-description languages such as Verilog or VHDL focus on the final implementation details and, because of that, are more verbose. Bigger code repositories are not easy to verify for functional correctness. On the other hand, HLLs are more compact and simulate faster. Thus, a designer can do more verification in the same span of time. Despite these advances, FPGA programming remains complex. This has compelled academia and industry to develop new compilers, frameworks, and libraries to facilitate hardware design.

High-Level Synthesis and Languages

High-level synthesis (HLS), also known as behavioral or algorithmic synthesis, is an automated design process that takes as input a functional description of a design and outputs an RTL implementation. It transforms an untimed (or partially timed) high-level specification into a fully timed implementation. The process of HLS starts by analyzing the data dependencies between the various operations in the functional description. This analysis leads to a Data Flow Graph (DFG) representation. After the DFG generation, during the allocation phase, HLS maps each operation onto a hardware resource with latency and area characteristics. Then, HLS adds the notion of time to the design during the scheduling phase. Scheduling takes the operations and resources of the DFG and decides in which clock cycle to execute them, given

their latency information. This step infers sequential logic by adding registers between operations and creating finite state machines [472].

Over the past three decades, many HLS tools have been proposed. The work in [473] presents an evaluation of different academic and commercial HLS tools tested on the same set of benchmarks. These tools have different input languages, perform different internal optimizations, and produce different quality results, even for the same input languages. The results show that each HLS tool can significantly improve performance once the designer has mastered benchmark-specific optimizations and constraints. However, academic HLS tools have a higher learning curve because of a minor focus on usability. Commercial HLS tools have an advantage because of their better documentation, robustness, and design verification integration.

In terms of input languages for HLS, most of the HLLs are variants of the C language. However, there are a few limitations to generate hardware from a pure C specification. First, C lacks the notion of timing and concurrency. The designer must rely on the HLS tool to create clock-based timing. Similarly, the designer must specify the concurrency model or rely on HLS to extract the parallelism among operations or processes. Second, C lacks bit-accurate data types. It only provides "native" data types such as `char`, `int`, and `long`, whose size is a multiple of a byte. Third, it lacks the concepts of hardware interfaces and communication channels. SystemC was adopted as HLS language to address all of these limitations [474]. However, SystemC still has not entirely made inroads in the FPGA community. Another common problem with all C-based languages, including SystemC, is memory access and modeling. These languages have a flat memory model, and memory access is done through pointers. Either HLS has to decide how to implement the memories in hardware, or the designer must leverage additional HLS directives or libraries to model the memory sub-system properly. Finally, in the family of the C-based specification languages for HLS, the SYCL language is emerging. SYCL (pronounced sickle) is an industry-driven standard that adds parallelism to C++ to design heterogeneous systems. SYCL programs perform best when paired with SYCL-aware C++ compilers such as the open-source data-parallel C++ (DPC++) compiler [475].

Apart from the variations of C, Bluespec is an open-source language for the description and synthesis of hardware based on SystemVerilog. It provides levels of abstraction with a clean semantic that highlights aspects of the architecture. It can be considered a high-level functional HDL, where modules are implemented as rules using SystemVerilog syntax. Those rules are called guarded atomic actions and express behaviors as concurrently cooperating finite state machines (FSMs). Another recent language among FPGA designers is Chisel. It is based on Scala and supports hardware definition using highly parameterized generators, object-oriented and functional programming. Similar to an HLS flow, it compiles into an RTL Verilog implementation.

Although all these languages have helped create efficient hardware and significantly shorten the development time, specific coding techniques are still necessary. Also, the growth and diversification of the application domains have shown the limitations of these programming languages. This has further pushed the level of abstraction to domain-specific languages (DSLs). In recent years, we are observing the growth of a considerable corpus of DSLs and frameworks for FPGA designs [476, 517]. In a DSL-based approach, the users and the tools can use domain knowledge to apply static and dynamic optimizations. However, a domain-specific HLS tool requires an appropriate compiler and a development environment that caters to the target domain. Table 5 shows some of the DSLs and frameworks developed over the years for FPGA computing organized by domains of application. Although all the approaches in the table are diverse in

Table 5. A brief taxonomy of domain-specific languages and frameworks for FPGA applications

Domain and Interfaces	DSLs and Frameworks
Signal-Processing	HDLCoder [476], LabView [476], Spiral [477], VSIPL [478]
Networking	SNORT [479], Click [480], P4 [481], Floem [482]
Databases	Glacier [483]
Machine Learning	OptiML [484]
Numerics	Verilog AMS [485]
Streaming	Maxeler [486], SCORE [487], Lime [488], Aetherling [489]
Dataflow	OpenDF [490], OpenSpatial [476]
Graphs	GraphStep [491], GraphGen [492]
Data Parallel	MapReduce [476], Accelerator [493], FCUDA [494], SuSy [495]
Circuit Generators	Flopoco [496], JHDL [497], PAMDC [498]
Image processing	HIPACC [499], FROST [500], Darkroom [501], RIPL [502], PolyMage [503]
Static	JBits [504], TVM [505]
Task based	TAPAS [506]
Dynamic	PyRTL [507], APARAPI [508], TornadoVM[509], Caldeira et al. [510], LINQits [511], DHDL [512], Spatial [513]
Type Systems	DAHLIA [514]
Verification	Kami [515]
Virtualization	Cascade [516]

terms of applications, the interesting question is, what are the common denominators? To the best of our knowledge, most of the approaches are broadly based on two approaches: either the DSL specification gets directly compiled into the RTL implementation, or the approach leverages source-to-source compilers. In the latter case, the DSL compiler produces an equivalent source code in a different programming language, for example, C++, for a more standard HLS flow. As a final concluding remark for this paragraph, the efforts for designing better HLS compilers and languages are a significant part of present FPGA research. Furthermore, the work in Table 5 by no means is an exhaustive list. The area of DSLs for FPGA easily outnumbers the work presented in the table.

Software and Hardware Integration

Running an application as software on a microprocessor is more accessible than designing and running specialized hardware, but it may result in poor performance and higher power costs. On the other hand, partitioning an application into software and hardware components is challenging. This process, also known as hardware/software codesign, divides an application between software running on the microprocessor and one or more custom hardware or co-processors components to achieve desired performance goals. Understandably there exists a plethora of research work in this area. The authors in [518] have provided background information on notable aspects of older FPGA technologies and simultaneously explained the fundamental architectures and design methods for codesign. Furthermore, the work in [519] is another comprehensive study that aims to evaluate and analyze the microarchitectural characteristics of state-of-the-art CPU-FPGA platforms in depth. That paper covers most of the shared-memory platforms with detailed benchmarks.

The two leading FPGA vendors, Xilinx and Intel, have their own solutions. The Xilinx Runtime Library (XRT) [520] is implemented as a combination of userspace and kernel driver components. It supports both PCIe-based boards and MPSoC based embedded platforms. Similarly, Xilinx SDSoc [521] and SDAccel [522] became publicly available later in late 2015; the former works only on select boards of the Zynq family of FPGAs, the latter only on selected PCIe-based boards for OpenCL computing. Since 2020

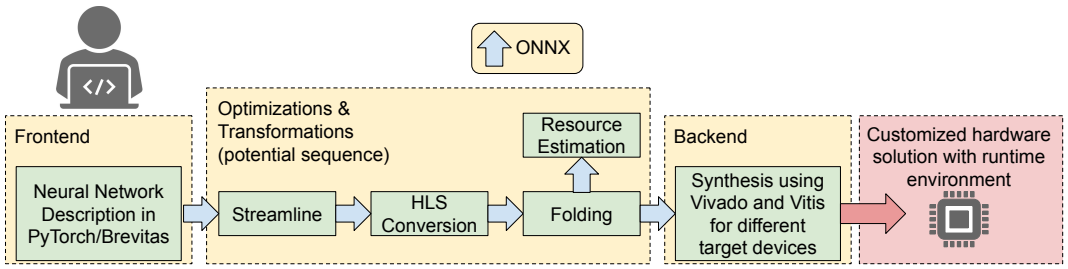

Figure 12. FINN Compiler Flow

Xilinx has introduced Vitis [523] as a unified platform. Vitis Unified Software Platform is a comprehensive development environment to build and seamlessly deploy accelerated applications on Xilinx platforms, including on-premises Alveo cards, FPGA-instances in the cloud, and embedded platforms. In addition, the recent efforts of Xilinx under the flagship Versal [524] is also a step towards codesign applications. Intel has the Open Programmable Acceleration Engine (OPAE) [525] which is the API library for programmers writing host applications that will leverage the FPGA acceleration. Likewise, Intel oneAPI [526] is an open, unified programming model built on standards to simplify the development and deployment of data-centric workloads across CPUs, GPUs, FPGAs, and other accelerators.

Apart from vendor solutions, academia and the open-source community have also attempted to simplify the integration of applications, operating systems, and hardware acceleration. For a comprehensive analysis, the reader is referred to the works in [527, 528], which give a historical review and summary on ideas and key concepts to include reconfigurable computing aspects in operating systems. They also present an overview of published and available operating systems of the last 30 years targeting reconfigurable computing. Similarly, the design exploration and engineering of FPGA drivers that are portable across multiple physical interfaces (PCIe, Ethernet, optical links) have remained a significant part of HW/SW codesign research. The challenges come from the variety of FPGA boards, the plethora of interfaces, and the diverse user requirements. Fundamentally, the FPGA drivers should allow the designer to load or reconfigure an application bitstream and support data transfers between the FPGA and host.

A significant engineering challenge is to consider how to partition driver functionality between the hardware and software components. One growing research focus is to exploit the spatial parallelism of FPGA technology through implementing multiple queues on FPGA drivers. A thorough analysis of system-level drivers for FPGA is out of the scope of our white paper. Readers interested in FPGA system-level drivers are referred to the work in [529, 530]. The authors of those papers have provided benchmarks of various mainstream academic and vendor solutions regarding system-level drivers in the FPGA domain.

Despite various existing OS and driver solutions, an open problem that remains is standardization. An industry-wide standardization would allow for faster development and better portability, and (re)usability of FPGA applications. There is already ongoing work in this area. Standards like the CCIX consortium [531] and the Heterogeneous System Architecture (HSA) foundation [532] have already made good progress.

The Case for ML Frameworks for FPGA Design

Machine learning is one of the fastest growing application domains and over the years there has been an increasing demand for FPGA-based implementations, as the FPGA can achieve latency and throughput and efficiency requirements through extreme customization of the hardware design leveraging reduced precision arithmetic, streaming dataflow implementations (as were introduced as spatial architectures), and

fine-granular sparsity. In order to enable a broad spectrum of users with these customizations and to reduce the significant engineering effort, compilers and tools are needed that cater to the needs of ML researchers and domain experts working with FPGAs. Two main ML frameworks are making the effort to fill this vacuum: hls4ml and FINN. Considering the aforementioned tools, compilers, programming languages, and codesign solutions, both hls4ml and FINN have the potential to reach a broader scientific community. To get a better understanding of how such a tool flow works, we consider the FINN compiler in more detail in the following paragraphs.

The **FINN compiler** [462] is an open-source framework to generate spatial DPU or streaming dataflow accelerators on FPGAs. The FINN compiler has a highly modular structure as shown in Figure 12, which allows the user to interactively generate a specialized architecture for a specific DNN. The framework provides a frontend, transformation and analysis passes, and multiple backends to explore the design space in terms of resource and throughput constraints. Brevitas [533], a PyTorch library for quantization-aware training, is the **frontend** used in this work. It enables training DNNs with weights and activations quantized down to a few bits, then exports the trained network into the intermediate representation (IR) used by the FINN compiler. The **transformation and analysis passes** help to generate an efficient representation of the DNN. Finally, the **backend** contains a code generator that creates synthesizable accelerator descriptions, which can be implemented as either a standalone Vivado IPI component or integrated into various shells, including Xilinx Alveo boards and PYNQ embedded platforms.

For further processing, the DNN model must be converted into the IR of the FINN compiler first. The frontend stage takes care of this by converting the PyTorch description into the IR, called FINN-ONNX. This IR is based on ONNX [534], an open-source interchange format that uses a protobuf description to represent DNNs. It comes with several standard operators and allows the user to easily create their own operators to customize the model. The nodes represent layers and edges carry outputs from one layer to become inputs to another. The feature to customize the ONNX representation is used in the framework to add application-specific nodes and attributes. Each node is tagged with the quantization of its inputs, parameters (weights and activations), and outputs to enable quantization-aware optimizations and the mapping to backend primitives optimized for quantized computation. During the compiler flow the nodes will be transformed into a backend-specific variants via a series of transformation passes.

The main principle of the **FINN compiler** is **graph transformation** and **analysis passes**, which change or analyze the IR of the model. A pass is a function that takes the IR graph as input and either (a) transforms the DNN by looking for a certain pattern, changing the graph in a specific manner and outputs the modified graph, or (b) **analyzes** the DNN to produce metadata about its properties. To bring the model into a representation from which code can be produced and finally the hardware accelerator can be generated, various transformations must be applied. The main transformations involved are summarized below.

Although the PyTorch description of the network is mostly quantized, it may still contain some floating-point operations from e.g. preprocessing, channelwise scaling or batchnorm layers. In order to generate a hardware accelerator from the model, these floating-point operations must be absorbed into multi-level thresholds, so that a functionally identical network of integer operations is created. The transformation to achieve this is called **streamlining**, as described by Umuroglu and Jahre [535]. During streamlining, floating-point operations are moved next to each other, collapsed into a single operation, and absorbed into succeeding multi-thresholding nodes.

Next, high-level operations in the graph are **lowered** to simpler implementations that exist in the FINN HLS-based hardware library. For instance, convolutions will be lowered to a sliding window node followed by a matrix-vector node, while pooling operations will be implemented by a sliding window followed by an aggregation operator. The resulting graph now consists of layers that can be converted to hardware building block equivalents. Each node corresponds to a Vivado HLS C++ function call, from which an IP block per layer can be generated using Vivado. The resources utilized by each hardware building block can be controlled through specific attributes passed from FINN to Vivado. For example, multiplications can be performed with LUTs or DSP blocks, and parameters can be stored in distributed, Block, or Ultra RAM.

Finally, the **folding** process assigns compute resources to each layer to obtain the desired throughput with a balanced pipeline by fine-tuning their degree of parallelism. To enable per-layer specialization without reconfiguration and minimize latency, FINN creates dedicated per-layer hardware interconnected with FIFO channels, thus the outermost loop across L layers is always fully pipelined. Once the folding is specified, resource estimates can be produced for each node. There are several ways to estimate the resources. Even before IP blocks are generated from the HLS layers, an estimate of the resources per layer can be made by using analytical models based on the concepts from the FINN-R paper [463]. Estimations can also be extracted from Vivado HLS after IP generation, though these results are still estimations that may differ from the resource usage of the final implementation due to synthesis optimizations.

The **Backend** is responsible for consuming the IR graph and backend-specific information to create a deployment package, also implemented using the transformation concept. To get the inference accelerator, between the layers FIFOs are inserted, which can be sized automatically by the FINN compiler. Afterwards, the single IP blocks are stitched together and synthesized. The stitched IP can be manually integrated into a system, or inserted into an appropriate shell for the target platform. If the target platform is an Alveo card, the design is exported as a Vivado Design Checkpoint (DCP), followed by generation of Xilinx Vitis [536] object files and linking.

Summary of Hardware/Software Codesign and FPGA-based Systems

In summary, CPUs are the most general solution for CNN inference but high in power. GPUs and DPUs offer highest performance, whereby GPU are more expensive in regards to energy cost. FPGAs offer several tradeoffs that may well fit rapidly moving application domains. FPGAs can adopt any precision and numerical representation, which provides utmost flexibility and leverages optimization with quantization to the maximum, whereas hardened approaches need to default to the next higher supported precision where the reduced precision variable can be embedded. Furthermore, through the spatial dataflow approach, much lower latency can be achieved. However, the complexity of programming FPGAs limits their deployment. Tools such as hls4ml and FINN are frameworks specifically created for the ML domain where they automate the process of hardware generation for the end-user thus hiding the associated design complexity of FPGAs and enabling them for the previously discussed end applications.

4.5 Beyond-CMOS neuromorphic hardware

With rapidly growing machine learning applications comes the acute need for their efficient hardware implementations. Most of the efforts are focused on digital CMOS technology, such as implementations based on general-purpose TPUs/GPUs, FPGAs, and more specialized ML hardware accelerators. The steady improvements in such hardware platforms' performance and energy efficiency over the past decade are attributed to the use of very advanced, sub-10-nm CMOS processes and holistic optimization of circuits, architectures, and algorithms. It includes, for example, taking advantage of aggressive voltage supply scaling [537], very deep pipelines and extensive data reuse in architectures [538], and lowering

the precision of weights and activations of the algorithms [539]. As a result, very compact state-of-the-art neural networks, such as MobileNet based on 3.4M parameters and 300M multiply-and-add operations per inference [540], can now be fitted entirely on a single chip. However, on all these fronts, advances are saturating and cannot rely on the faltering Moore's law.

On the other hand, further progress would be essential because ML algorithms are getting increasingly more complex. For example, transformer networks [541], the state-of-the-art approach for many ML tasks today [541–543], could have hundreds of billions of parameters and perform hundreds of trillions of operations per inference. Moreover, the transformer's functional performance typically improves with the model size [544, 545]. Training such models requires enormous, data-center-scale (e.g., kiloTPU-year) resources while performing inference on resource-constrained edge devices would be extremely challenging.

The opportunities for building more efficient hardware may come from biological neural networks. Indeed, it is believed that the human brain, with its $>1000\times$ more synapses than the weights in the largest transformer network, is extremely energy efficient [546], which serves as a general motivation for developing neuromorphic hardware [547]. There is a long history of CMOS neuromorphic circuits [548]. However, unleashing the full potential of neuromorphic computing might require novel, beyond-CMOS device and circuit technologies [549] that allow for more efficient implementations of various functionalities of biological neural systems.

In this section, the most prominent emerging technology proposals, including those based on emerging dense analog memory device circuits, are grouped according to the targeted low-level neuromorphic functionality - see, e.g. reviews in [550–553] and original work utilizing volatile [554–564] and nonvolatile [565–580, 561, 581] memristors, phase change memories (PCM) [582–588], and nonvolatile NOR [589–591, 579], and NAND [592–594], and organic volatile [595] floating gate memories, as well as multiferroic and spintronic [596–600], photonic [601, 602, 588, 603–611], and superconductor [612, 609, 613] circuits. More discussion is devoted to analog vector-by-matrix multiplication circuits in the following subsection because of their immediate value for today's state-of-the-art algorithms. More biologically-realistic proposals described in the subsequent sections are less emphasized because they target algorithms with inferior performance. The least mature though very intriguing quantum neuromorphic computing [614, 615] is not discussed in this brief review.

Analog Vector-by-Matrix Multiplication

The emergence of dense analog-grade nonvolatile memories in the past two decades renewed interest in analog-circuit implementations of vector-by-matrix multiplication (VMMs) [547, 616, 565, 617, 618, 589, 591], which is the most common and frequently performed operation of any neural network in training or inference [619, 620]. In the simplest case, such a circuit is comprised of a matrix of memory cells that serve as configurable resistors for encoding the matrix (synaptic) weights and peripheral sense amplifiers playing the role of neurons (Fig. 13). The input vector is encoded as voltages applied to rows of the memory matrix so that the currents flowing into virtually grounded columns correspond to VMM results. Because addition and multiplication are performed on the physical level, via Kirchhoff's and Ohm's laws respectively, such an approach can be extremely fast and energy-efficient, provided that memory devices are dense and their conductances are adjustable (i.e., multi-state). The energy efficiency in part comes from performing "in-memory" computing that reduces the amount of data (corresponding to the synaptic weights) that are

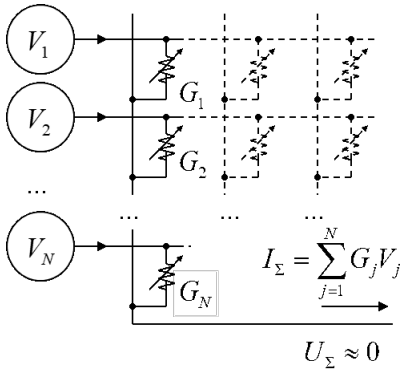

Figure 13. Analog vector-by-matrix multiplication (VMM) in a crossbar circuit with adjustable crosspoint devices. For clarity, the output signal is shown for just one column of the array, while sense amplifier circuitry is not shown. Note that other VMM designs, e.g. utilizing duration of applied voltage pulses, rather than their amplitudes, for encoding inputs/outputs, are now being actively explored – see, e.g., their brief review in Ref. [551]

.

moved across or in-and-out of the chip during computation. Such communication overhead could dominate the energy consumption in the most advanced digital CMOS implementations.

The general challenge towards practical adoption of such circuits, especially when using the most prospective emerging memory technologies, is variations in I-V characteristics, e.g., in the switching voltages applied to change the memory state. In light of this challenge, the most straightforward application is ex-situ trained inference accelerators for the earlier firing-rate neural networks [551], i.e., the so-called second generation of artificial neural networks (ANNs) with graded-response neurons. In such applications, memory devices are updated infrequently, only when new inference functionality should be programmed. Thus, crosspoint devices' conductances can be tuned with slower, more tolerant to device variations write schemes. For example, after the weights have been found in the software, memory cells are programmed, one by one, using feedback write-verify algorithms that can adapt to the unique I-V characteristics of each device [565]. For the same reason, the switching endurance, i.e., the number of times the memory devices can be reliably programmed, and the write speed/energy are less critical. Additionally, VMM operations in the inference of many neural networks could be performed with moderate, less than 8-bit precision, without incurring accuracy loss [621], which further relaxes requirements for analog properties and permits more I-V non-idealities and noise.

The most advanced neuromorphic inference circuits have been demonstrated with more mature floating-gate transistor memory circuits. Up until recently, such circuits were implemented primarily with "synaptic transistors" [622], which may be fabricated using the standard CMOS technology, and several sophisticated, efficient systems were demonstrated [546, 618, 623]. However, these devices have relatively large areas ($>10^3$ F^2, where F is the minimum feature size), leading to higher interconnect capacitance and hence larger time delays. More recent work focused on implementing mixed-signal networks with much denser ($\sim$40 F^2) commercial NOR-flash memory arrays redesigned for analog computing applications [591, 589]. For example, a prototype of a 100k+-cell two-layer perceptron network fabricated in a 180-nm process with modified NOR-flash memory technology was reported in Ref. [590]. It performed reliably, with negligible long-term drift and temperature sensitivity, and reproducible classification of the MNIST benchmark set images with $\sim 95\%$ fidelity and sub-1-μs time delay and sub-20-nJ energy consumption per pattern.

The energy-delay product was six orders of magnitude better than the best (at that time) 28-nm digital implementation performing the same task with a similar fidelity [590].

Recent theoretical studies showed that neuromorphic inference circuits could be also implemented with much denser 3D-NAND flash memories [592–594], projected to scale eventually to 10 terabits per square inch density. In the long term, the most promising are perhaps circuits based on metal-oxide resistive switching random access (ReRAM for short, which are also called metal-oxide memristors) [552, 553], especially their passively integrated (0T1R) technology variety [576]. Indeed, due to the ionic switching mechanism, ReRAM devices with dimensions below 10 nm still retain excellent analog properties and year-scale retention [567]. Furthermore, a low-temperature fabrication budget allows monolithic vertical integration of multiple ReRAM crossbar circuits, further increasing effective density [566]. There has been rapid progress in scaling up the complexity of ReRAM-based neuromorphic circuit demonstrations over the past several years [568, 571–576]. However, the ReRAM technology is still in much need of improvement. In addition to high device variations, another remaining issue is high write currents and operating conductances, which must be decreased by at least one order of magnitude to reduce the significant overhead of peripheral circuits [576].

The device requirements for training hardware accelerators are different and much more stringent. For instance, long retention is not required because weights are frequently updated. That allows using volatile memories in analog VMM circuits, such as interfacial memristors based on electron trapping/detrapping switching [554–556] and solid-state-electrolyte memories [595, 557, 549], or even capacitor-based memories controlling current via crosspoint transistors [584]. However, the toughest challenge is much higher computing and weight precision required for training operation and the need for efficient schemes for weight updates, which in turn necessitate drastically tighter device variations. The additional related requirement is that the change in device conductance upon applying the write pulse should not depend on its current state (the so-called linearity of update property). Otherwise, accurate conductance adjustment would require sending a unique write pulse based on the current device state, which would be hardly compatible with fast (in parallel) weight update.

Phase change memories have also been investigated as candidates for variable resistors in analog VMM circuits [582, 586], though their main drawback is significant drift in the conductive state over time. High write endurance, high density (with vertical 3D-NAND-like integrated structure), and long retention are demonstrated in 1T Ferroelectric RAM devices. There is much excitement about such devices' applications in training and inference accelerators [600], though their analog properties are probably inferior to ReRAM. The significant drawbacks of magnetic devices, such as magnetic tunnel junction memories, are smaller on/off current ratios, insufficient for practical VMM circuits, and poor analog properties for scaled-down devices [596].

The potentials of using light for implementing fast and large-fanout interconnect and linear computations, such as multiply-and-add operation, have motivated photonic neuromorphic computing research [549, 602, 601, 605]. Different implementation flavors, e.g., with fixed [603] and programmable [588, 604, 606, 607] functionalities, have been recently suggested in the context of modern neural networks. Specifically, Ref. [603] reports a system of multiple 3D-printed optical layers, each being a mesh of regions (neurons) with specifically chosen transmission-reflection properties, which can perform pattern classification inference similar to the convolutional neural networks. By sending a coherent light with amplitude-encoded input, a useful computation is performed at the speed of light. Specifically, the light diffracts and

interferes when passing through the optical system and is ultimately steered to the specific region at the output layer corresponding to the pattern class. Refs. [588, 604, 606, 607] report optical neuromorphic systems with configurable weights. The inputs are encoded in the light's energy, and the weights are encoded by optical attenuation in PCM devices in Ref. [588] so that a product is computed by passing the light via PCM device. Ref. [607] proposes encoding inputs with light amplitude and uses specific frequency for different VMM inputs. The light from inputs is combined and passed to the frequency selective weight banks based on a microring resonator (MRR) that features metal heaters to perform multiplication. In particular, the MRR coupling (i.e., weight) is controlled via heating by adjusting currents supplied to each MRR. In these reconfigurable implementations, the product accumulation (i.e., the summation operations in the VMM) is performed by integrating the light-induced charges on the photodetector. A very aggressive time-division multiplexing scheme for calculating VMM in which both weights and inputs are encoded in the coherent light's amplitude is proposed in Ref. [604]. At one step of such scheme, the input light is fanned out into n channels and combined with the light-encoded n weights using a beam splitter and then sent to n homodyne photodetectors to compute n products in parallel. All-optical feed-forward inference based on Mach-Zehnder interferometer meshes utilizes single-valued decomposition for the weight matrix [606]. Unitary matrix transformations are implemented with optical beam splitters and phase shifters, while the diagonal matrix is implemented with optical attenuators.

In principle, sub-aJ energy and sub-ps latency for a single multiply-and-add operation might be possible with optical computing [605]. However, the main challenge remains much large dimensions of the optical components and the very high I/O overhead of converting to and from optical domains [549, 601, 605]. The designs that rely on conversion to the electrical domain would be especially affected by poor integration density of optical devices due to larger electrical communication overheads, which were shown to overwhelm system-level performance of (much denser) ReRAM based circuits [551]. Optical systems would ultimately benefit from very wide ($\gg$10,000) dot-products and/or utilizing deep time-division multiplexing to amortize the I/O overhead. However, the possible issues of nonlinearities in charge integration and utility of such wide dot-product computations remain unclear [605].

Stochastic Vector-by-Matrix Multiplication

Computations performed by the brain are inherently stochastic, in that, e.g. substantially different neural responses are observed to the repeatable presentation of identical stimuli [624]. Such noisy operation is mimicked by probabilistic neural networks, such as Boltzmann machines [625] and deep belief neural networks [626]. In the simplest case, such a network is comprised of binary neurons that compute stochastic dot products, i.e., probabilistically generate output according to their pre-activation (dot-product) values.

The stochastic functionality can be realized at either the synapse or the neuron side. In the latter, more straightforward scenario, the neuron first computes a dot-product of its inputs and corresponding weights deterministically. The result is then passed to some "probabilistic" activation function, e.g., used as an argument in the sigmoid probability function, to determine the probability of generating high output. Because of the typically large (> 100) ratio of synapses to neurons, the efficient deterministic dot-product implementations, e.g., with the already discussed analog VMM circuits, is of primary importance for realizing high-performance probabilistic neural network hardware. Still, earlier work showed that even the simplest, deterministic neurons may incur substantial overhead, e.g., occupy up to 30% of the area and consume up to 40% of energy for some neural network models [551]. Hence neuromorphic hardware would also benefit from the efficient realization of stochastic neurons.

Emerging devices can be broadly employed in two ways to achieve stochastic functionality, namely by using either dynamic or static I-V characteristics of memory devices. Specifically, the former approach is to utilize intrinsically stochastic switching between memory states in emerging memory devices. For example, in MTJ memories, thermal fluctuation causes stochastic transition between the low resistance parallel and high resistance antiparallel states so that the probability of the final memory state upon switching could be controlled by the spin-torque current [596]. The melt-quench-induced reconfiguration of the atomic structure is intrinsically stochastic in phase-change memories (PCMs) [583]. These phenomena were suggested for implementing MTJ [597] and PCM [583] stochastic neurons. The second approach is to utilize intrinsic and extrinsic current fluctuations in memory devices, e.g., random telegraph [577] and thermal noise [578] in ReRAM devices, or shot-noise in nanoscale floating gate transistors [578, 579]. In such an approach, the noisy current flowing into the neuron is compared against a reference value, e.g. using a simple latch, to implement a probabilistic activation function [579].

The primary concern for the former approach is the limited endurance of many memories and the drift in the stochastic switching properties upon repeated switching. An additional drawback is a necessity for the co-integration of multiple memory device technologies for scalable stochastic dot-product circuits, e.g., integrating ReRAM-based artificial synapses and MTJ-based neurons. On the other hand, analog circuits based on ReRAM devices only (Fig. 13), though operating at a much lower signal-to-noise ratio (SNR), can be utilized to implement stochastic VMM of the second approach. Furthermore, adjusting read voltages in such a circuit allows for controlling SNR. Hence, the control of effective temperature, i.e. the slope of sigmoid probability function, enables efficient implementation of stochastic annealing in Boltzmann machines during runtime. The second approach's possible downside is slower operation because of lower read currents (which can be potentially addressed by utilizing external noise instead [579]). Finally, the impact of noise quality on functional performance is another common concern. This issue has not been systematically studied yet, though Gaussian-like thermal or shot noise should be more advantageous for truly random operation.

Spiking Neuron and Synaptic Plasticity

Despite much recent progress in algorithms [627, 628], the most biologically plausible, spiking neural networks (SNNs) [620] are still inferior in the functional performance to simpler ANNs. If simpler ANNs would remain superior, the work of efficient SNN hardware could still be justified by the need to efficiently interface to the brain and/or model it, which in turn could lead to the development of higher-cognition artificial intelligence algorithms. An additional intriguing feature of SNNs is local weight update rules, requiring only information from pre- and post-synaptic neurons that could enable large-scale neuromorphic hardware with real-time training capabilities [629].

In the simplest SNN models, the information is encoded in spike-time correlations [620], while the network function is defined by the synaptic weights, which are adjusted based on the relative timing of spikes that are passed via synapses. In addition to VMM, the essential operations in SNNs are leaky-integrate-and-fire (LIF) functions performed by neurons and various types of synaptic plasticity, such as short-term plasticity (STP) and long-term potentiation (LTP), and spike-timing-dependent-plasticity (STDP) [620]. LIF neurons mimic the dynamic processes in the neuronal membrane, while synaptic plasticities mimic learning and memory mechanisms in biological networks. For example, STP is a temporary change in the synaptic strength implementing a short-term memory. Without immediate reinforcement of synaptic weight adjustment, the memory would be lost, i.e., the synaptic weight would relax to the original equilibrium state. On the other hand, the frequently repeated spiking stimulus causes long-term memory, e.g., permanent

potentiation via the LTP mechanism. STDP is a time-dependent specialization of Hebbian learning. Its specific goal is to strengthen the synaptic efficiency when pre- and post- synaptic spikes happen in the expected causal temporal order and weaken it otherwise.

A compact implementation of LIF neurons with biological, ms-scale integration times using conventional circuit technology is challenging because of the large capacitors that are required. Leaky integration circuits utilizing volatile memristors (e.g., based on filamentary [562], interfacial [563], and Mott insulator [564] switching mechanisms) have been suggested to address this problem. In such implementations, the integrated current is encoded with a conductive state of the volatile memory device. Neuron spiking functionality was demonstrated with threshold-switching (volatile) memory devices that feature S-type negative differential resistance (NDR) I-V characteristics [560]. This approach's general idea is similar to the oscillator circuits based on S-type (NDR) device connected to a resistor-capacitor circuit [630]. LIF neurons based on spin-torque magnetic memories were simulated in Ref. [598]. In such a neuron, spin-torque oscillations are employed to generate spikes, while incremental magnetization and its relaxation mimic integration and leakage, respectively.

STP to LTP transition has been emulated with solid-state-electrolyte devices – see, e.g., original work in Ref. [558] and more recent work on "diffusive" memristors [559]. Specifically, the short and infrequent write pulses result in the formation of thin filaments, which are unstable and quickly dissolve, representing a short memory. However, a thicker and more stable filament can be formed by applying repeated and/or longer write pulses, thus mimicking transition to the LTP. Different STDP window implementations, e.g., using PCM [587] or metal-oxide ReRAM [569] devices, have been suggested by carefully selecting the shape of pre and post-synaptic write voltage pulses—see a comprehensive review of the emulated synaptic plasticity with memristive devices in Refs. [631, 632].

Several small-scale spiking neuromorphic systems based on emerging device technologies were demonstrated, including coincidence detection via STDP mechanism based on metal-oxide memristors [570, 581] and temporal data classification with diffusive memristors [561]. However, the overall progress in such advanced hardware has been much slower compared to simpler ANNs inference accelerators. The main reason is more demanding functionality from emerging devices in such applications and hence the more severe impact of device variations on the SNN operation and performance. For example, SNNs rely on fixed-magnitude spikes to update the conductance of multiple devices in parallel. Because of that, change in the conductances could vary drastically even with minor variations in I-V's switching voltages, which in turn leads to very significant variations in STDP characteristics [570]. On the other hand, as already mentioned above, the implementation of simpler ex-situ trained ANNs is much less challenging because the write amplitude voltages in such networks can be adjusted uniquely for each device based on the feedback information during conductance tuning [565].

Superconductor circuits, e.g., based on rapid single flux quantum (RSFQ) variety [633], are naturally suited for spiking circuits due to information encoding in SFQ voltage pulses. For example, Josephson Junction spiking neurons operating at up to 50 GHz range have been demonstrated in Ref. [612]. The historical challenges of such an approach include inferior fabrication technology (which may finally change given the enormous investments in superconductor quantum computing), the low-temperature operation that limits its applications, and the lack of efficient analog memory circuits [634]. The photonic spiking neural networks (e.g., Ref. [608]) and hybrid superconductor / optoelectronic neuromorphic circuits [609] share the same challenges of the already discussed photonic neuromorphic inference approaches.

Reservoir Computing

Due to intrinsic memory properties, recurrent neural networks, such as Google Neural Machine Translation model, are especially suitable for processing sequential or temporal data. Reservoir computing (RC) networks are a special type of efficiently learning recurrent networks [635], that were motivated by cortical information processing [636]. Among its variants are liquid state machines [637], which is a spiking RC network, and echo state networks [638], an RC based on a very sparse recurrent network. The main component in RC networks is a reservoir, which is a nonlinear recurrent network that maps inputs into a higher-dimensional spatio-temporal representation and has the property of a fading memory of the previous inputs and network states. Another component is a readout layer, which maps the intermediate state to the outputs. All connections in the reservoir are fixed and only weights in the readout layer are trainable. Because of that and sparse intermediate representation, faster and online algorithms can be employed for training such networks, which is a primary strength of this approach.

Though both readout and the reservoir can also be realized with the discussed analog VMM circuits, intriguing opportunities for implementing the reservoir are presented by nonlinear physical phenomena in superconductor, magnetic, and photonic devices [639]. For example, spoken vowel recognition was demonstrated with RC in which the reservoir was implemented with four coupled MTJ-based spin-torque oscillators (STO) [599]. In such a demo, the temporal input corresponding to spoken vowels is first converted to the frequency domain, which is in turn mapped to the corresponding DC bias currents that are applied to the MTJ devices. The induced voltage on the STO devices is used as an output of the reservoir. The reservoir utilizes the nonlinear dependence of the frequency of STOs on the DC current and the history-dependent transient motions of the MTJ's free layer spins spin.

Various photonic reservoirs have been suggested [601], e.g. utilizing transient properties of optical systems with time-delayed feedback [610], or relying on superimposing lights that passively circulates via waveguides, splitters and combiners, and nonlinear conversion to the electronic domain [611], to achieve high-dimensional response. The dynamics in the superconductor circuits are recently studied for efficient and extremely fast reservoir implementation [613]. Specifically, the proposed reservoir is based on a Josephson transmission line (JTL) formed by a chain of biased JJs. An input pulse from one end of the JTL causes a rapid cascade of junction phase slips that propagate SFQ pulse to the other end. Because JJs modulate each others' currents, a complex dynamical state is achieved.

There are several general concerns with RC computing approaches. On the algorithmic level, RC is inferior in performance to state-of-the-art approaches and it is unclear whether without further algorithm improvements such a handicap can be outweighed by the advantages of online training. The main concern for various hardware implementations is again related to the device variations, e.g., whether the hardware would be able to produce repeatable results when applying the same input. An additional concern for magnetic devices is the limited coupling between devices which could impact the effectiveness of the reservoir.

Hyperdimensional Computing / Associative Memory

Hyperdimensional computing [640] circuits have been recently demonstrated with ReRAM [580] and PCM [585] devices. The low-level operation in hyperdimensional computing is closely related to that of associative or content addressable memory [619]. Specifically, at the core of such an approach is an associative memory array circuit that outputs the closest, in a Hamming distance sense, memory row entry to a binary input vector serving as a search key. Assuming symmetric binary representation, with -1 and

+1 encoding, Hamming distance is linearly related to a dot product, i.e., equal to output vector length minus dot product between the input vector and the stored memory row values. Therefore, the critical functionality in hyperdimensional computing is again a VMM operation. After the VMM operation has been completed, its results are passed to the winner-take-all circuit [619] (which is a harder version of a softmax function [641]) that determines the element with the smallest Hamming distance while discarding all other outputs. The additional simplification is that both input and weights in VMM are binary.

In principle, binary VMM can be more efficiently implemented in hardware than its fully analog version. Similar to binary neural networks [539], the apparent tradeoff is a worse functional performance of hyperdimensional computing. Another essential feature of hyperdimensional computing is the suitability for fast "one-shot" or incremental learning [640] though at the cost of having a much more redundant memory array. Note that fast "one-shot" learning is not unique to hyperdimensional computing. For example, Hebbian learning and its many variants used in training associative neural networks have recursive form and are naturally incremental in that the weights can be modified only based on current weight values and the new pattern stored in the network [619].

Concluding Remarks

Many emerging devices and circuit technologies are currently being explored for neuromorphic hardware implementations. Neuromorphic inference accelerators utilizing analog in-memory computing based on floating gate memories are perhaps the closest to widespread adoption, given the maturity of such technology, the practicality of its applications, and competitive performance as compared to conventional (digital CMOS) circuit implementations. Comparing the performance prospects of other neuromorphic approaches is not straightforward because many proposals target algorithms with inferior functional performance, especially those closely mimicking the brain's operation. Baring a substantial breakthrough in ML algorithms or the emergence of new applications that could benefit from high-performance low-accuracy neuromorphic hardware, the inferior functional performance may limit the practicality of other approaches. The main challenge, much more so for advanced neuromorphic computing concepts, remains significant variations in the operation of emerging devices.

5 OUTLOOK

This report has laid out exciting applications of fast ML to enable scientific discovery across a number of domains. This is a rapidly developing area with many exciting new studies and results appearing often. However, this is a relatively young area rich with potential and a number of open challenges across a number of fields. Beyond what has been laid out in the report, we hope that the discussion of scientific use-cases and their overlaps will provide readers with the inspiration to entertain and pursue additional applications.

In Section 4, we provided an overview of techniques for developing powerful ML algorithms that need to be operated in high throughput and low latency environments. This includes both system design and training as well as efficient deployment and implementation of those ML models. Implementation in hardware is discussed under two main categories—current conventional CMOS and more speculative beyond CMOS technologies. In the conventional CMOS case, in light of the end of Moore's Law, the recent emphasis has been focused on advanced hardware architectures designed for ML. We gave an overview of popular and emerging hardware architectures and their strengths and shortcomings. A key area of importance for the multitude of hardware is their codesign of a given ML algorithm for specific hardware including the architecture and programmability of that algorithm. An example of a particularly relevant and important hardware platform is for FPGAs and that is the use-case discussed in Section 4.4. Finally, we concluded with an overview of beyond CMOS technologies which offer exciting and ultra-efficient technologies on which we can implement ML models. While these technologies are speculative, they offer potential orders of magnitude improvement over conventional technologies.

Both ML training and deployment techniques and computer architectures are extremely rapidly moving fields with new works appearing at a pace difficult to keep up with, even for this report. While new methods are being introduced continuously in both spaces, it is particularly important to understand the codesign of new algorithms for different hardware and the ease of use of the tool flows for deploying those algorithms. Innovations here will allow rapid and broad adoption of powerful new ML hardware. In the case of beyond CMOS technologies, these practical considerations are important as well as considering the maturity of the technology, integration into computing architectures, and how to program such devices.

We look forward to revisiting these topics in the near future to see how quickly advances may come in applications, ML techniques, and hardware platforms—and most importantly their confluence to enable paradigm-shifting breakthroughs in science.

ACKNOWLEDGMENTS

We acknowledge the Fast Machine Learning collective as an open community of multi-domain experts and collaborators. This community was important for the development of this project.

REFERENCES

[1] Carleo G, Cirac I, Cranmer K, Daudet L, Schuld M, Tishby N, et al. Machine learning and the physical sciences. *Rev. Mod. Phys.* **91** (2019) 045002. doi:10.1103/RevModPhys.91.045002.

[2] Russakovsky O, Deng J, Su H, Krause J, Satheesh S, Ma S, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision* **115** (2015) 211. doi:10.1007/s11263-015-0816-y.

[3] Raina R, Madhavan A, Ng AY. Large-scale deep unsupervised learning using graphics processors. *Proceedings of the 26th Annual International Conference on Machine Learning* (New York, NY, USA: Association for Computing Machinery) (2009), ICML '09, 873. doi:10.1145/1553374.1553486.

[4] Cireşan DC, Meier U, Gambardella LM, Schmidhuber J. Deep, big, simple neural nets for handwritten digit recognition. *Neural Computation* **22** (2010) 3207. doi:10.1162/neco_a_00052.

[5] Krizhevsky A, Sutskever I, Hinton GE. Imagenet classification with deep convolutional neural networks. Pereira F, Burges CJC, Bottou L, Weinberger KQ, editors, *Advances in Neural Information Processing Systems* (Curran Associates, Inc.) (2012), vol. 25.

[6] University SM. Fast Machine Learning for Science Workshop (2019).

[7] Aad G, et al. Observation of a new particle in the search for the Standard Model Higgs boson with the ATLAS detector at the LHC. *Phys. Lett. B* **716** (2012) 1–29. doi:10.1016/j.physletb.2012.08.020.

[8] Chatrchyan S, et al. Observation of a New Boson at a Mass of 125 GeV with the CMS Experiment at the LHC. *Phys. Lett. B* **716** (2012) 30–61. doi:10.1016/j.physletb.2012.08.021.

[9] Collaboration TA. The ATLAS experiment at the CERN large hadron collider. *Journal of Instrumentation* **3** (2008) S08003–S08003. doi:10.1088/1748-0221/3/08/s08003.

[10] Calafiura P, Farrell S, Gray H, Vlimant J, Innocente V, Salzburger A, et al. Trackml: A high energy physics particle tracking challenge. *2018 IEEE 14th International Conference on e-Science (e-Science)* (2018), 344–344. doi:10.1109/eScience.2018.00088.

[11] Gray L, Klijnsma T, Ghosh S. A Dynamic Reduction Network for Point Clouds (2020).

[12] The Phase-2 Upgrade of the CMS Endcap Calorimeter (2017).

[13] Wang Y, Sun Y, Liu Z, Sarma SE, Bronstein MM, Solomon JM. Dynamic graph CNN for learning on point clouds. *CoRR* **abs/1801.07829** (2018).

[14] Qasim SR, Kieseler J, Iiyama Y, Pierini M. Learning representations of irregular particle-detector geometry with distance-weighted graph networks. *Eur. Phys. J. C* **79** (2019) 608. doi:10.1140/epjc/s10052-019-7113-9.

[15] Ju X, et al. Graph Neural Networks for Particle Reconstruction in High Energy Physics detectors. *33rd Annual Conference on Neural Information Processing Systems* (2020).

[16] Ju X, Farrell S, Calafiura P, Murnane D, Prabhat, Gray L, et al. Graph neural networks for particle reconstruction in high energy physics detectors (2020).

[17] Duarte J, Vlimant JR. Graph neural networks for particle tracking and reconstruction (2020).

[18] Tsaris A, Anderson D, Bendavid J, Calafiura P, Cerati G, Esseiva J, et al. The HEP.TrkX project: Deep learning for particle tracking. *Journal of Physics: Conference Series* **1085** (2018) 042023. doi:10.1088/1742-6596/1085/4/042023.

[19] Farrell S, Calafiura P, Mudigonda M, Prabhat, Anderson D, Vlimant JR, et al. Novel deep learning methods for track reconstruction (2018).

[20] Choma N, Murnane D, Ju X, Calafiura P, Conlon S, Farrell S, et al. Track seeding and labelling with embedded-space graph neural networks (2020).

[21] Battaglia PW, Pascanu R, Lai M, Rezende D, Kavukcuoglu K. Interaction networks for learning about objects, relations and physics (2016).

[22] DeZoort G, Thais S, Ojalvo I, Elmer P, Razavimaleki V, Duarte J, et al. Charged particle tracking via edge-classifying interaction networks (2021).

[23] Thais S, DeZoort G. Instance segmentation gnns for one-shot conformal tracking at the lhc (2021).

[24] Pata J, Duarte J, Vlimant JR, Pierini M, Spiropulu M. MLPF: Efficient machine-learned particle-flow reconstruction using graph neural networks (2021).

[25] Sirunyan AM, et al. Particle-flow reconstruction and global event description with the CMS detector. *JINST* **12** (2017) P10003. doi:10.1088/1748-0221/12/10/P10003.

[26] Paganini M, de Oliveira L, Nachman B. CaloGAN : Simulating 3D high energy particle showers in multilayer electromagnetic calorimeters with generative adversarial networks. *Phys. Rev. D* **97** (2018) 014021. doi:10.1103/PhysRevD.97.014021.

[27] Paganini M, de Oliveira L, Nachman B. Accelerating Science with Generative Adversarial Networks: An Application to 3D Particle Showers in Multilayer Calorimeters. *Phys. Rev. Lett.* **120** (2018) 042003. doi:10.1103/PhysRevLett.120.042003.

[28] de Oliveira L, Paganini M, Nachman B. Learning Particle Physics by Example: Location-Aware Generative Adversarial Networks for Physics Synthesis. *Comput. Softw. Big Sci.* **1** (2017) 4. doi:10.1007/s41781-017-0004-6.

[29] Musella P, Pandolfi F. Fast and Accurate Simulation of Particle Detectors Using Generative Adversarial Networks. *Comput. Softw. Big Sci.* **2** (2018) 8. doi:10.1007/s41781-018-0015-y.

[30] Bendavid J. Efficient Monte Carlo Integration Using Boosted Decision Trees and Generative Deep Neural Networks (2017).

[31] Hashemi B, Amin N, Datta K, Olivito D, Pierini M. LHC analysis-specific datasets with Generative Adversarial Networks (2019).

[32] Di Sipio R, Faucci Giannelli M, Ketabchi Haghighat S, Palazzo S. DijetGAN: A Generative-Adversarial Network Approach for the Simulation of QCD Dijet Events at the LHC. *JHEP* **08** (2019) 110. doi:10.1007/JHEP08(2019)110.

[33] Chen C, Cerri O, Nguyen TQ, Vlimant JR, Pierini M. Data Augmentation at the LHC through Analysis-specific Fast Simulation with Deep Learning (2020).

[34] Krupa J, et al. GPU coprocessors as a service for deep learning inference in high energy physics (2020).

[35] Duarte J, et al. FPGA-accelerated machine learning inference as a service for particle physics computing. *Comput. Softw. Big Sci.* **3** (2019) 13. doi:10.1007/s41781-019-0027-2.

[36] Rankin DS, et al. FPGAs-as-a-Service Toolkit (FaaST) (2020).

[37] Duarte J, et al. Fast inference of deep neural networks in FPGAs for particle physics. *J. Instrum.* **13** (2018) P07027. doi:10.1088/1748-0221/13/07/P07027.

[38] Aarrestad T, Loncar V, Ghielmetti N, Pierini M, Summers S, Ngadiuba J, et al. Fast convolutional neural networks on FPGAs with hls4ml. *Mach. Learn. Sci. Tech.* **2** (2021) 045015. doi:10.1088/2632-2153/ac0ea1.

[39] Loncar V, et al. Compressing deep neural networks on FPGAs to binary and ternary precision with `hls4ml`. *Mach. Learn.: Sci. Technol.* (2020) 015001. doi:10.1088/2632-2153/aba042.

[40] Coelho CN, Kuusela A, Zhuang H, Aarrestad T, Loncar V, Ngadiuba J, et al. Automatic deep heterogeneous quantization of deep neural networks for ultra low-area, low-latency inference on the edge at particle colliders (2020). Submitted to *Nat. Mach. Intell.*

[41] Coelho C. QKeras (2019).
[42] The Phase-2 Upgrade of the CMS Level-1 Trigger. Tech. Rep. CERN-LHCC-2020-004. CMS-TDR-021, CERN, Geneva (2020). Final version.
[43] Alimena J, Iiyama Y, Kieseler J. Fast convolutional neural networks for identifying long-lived particles in a high-granularity calorimeter. *Journal of Instrumentation* **15** (2020) P12006–P12006. doi:10.1088/1748-0221/15/12/p12006.
[44] Iiyama Y, Cerminara G, Gupta A, Kieseler J, Loncar V, Pierini M, et al. Distance-weighted graph neural networks on fpgas for real-time particle reconstruction in high energy physics. *Frontiers in Big Data* **3** (2021) 44. doi:10.3389/fdata.2020.598927.
[45] Heintz A, Razavimaleki V, Duarte J, DeZoort G, Ojalvo I, Thais S, et al. Accelerated charged particle tracking with graph neural networks on fpgas (2020).
[46] Summers S, et al. Fast inference of Boosted Decision Trees in FPGAs for particle physics. *J. Instrum.* **15** (2020) P05026. doi:10.1088/1748-0221/15/05/P05026.
[47] Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, et al. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* **12** (2011) 2825–2830.
[48] Chen T, Guestrin C. Xgboost. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2016). doi:10.1145/2939672.2939785.
[49] Therhaag J, Team TCD. Tmva - toolkit for multivariate data analysis. *AIP Conference Proceedings* **1504** (2012) 1013–1016. doi:10.1063/1.4771869.
[50] Savard C. Level 1 trigger track quality machine learning models on FPGAs for the Phase 2 upgrade of the CMS experiment (2019).
[51] Nottbeck N, Schmitt C, Büscher V. Implementation of high-performance, sub-microsecond deep neural networks on FPGAs for trigger applications. *JINST* **14** (2019) P09014. doi:10.1088/1748-0221/14/09/p09014.
[52] Fritzsche N. *Development of Digital Signal Processing for the ATLAS LAr Calorimeters with Artificial Neural Networks using FPGAs*. Master's thesis, TU Dresden (2020).
[53] ATLAS Liquid-Argon calorimeter: Technical Design Report. Tech. rep., CERN, Geneva (1996).
[54] Cleland WE, Stern EG. Signal processing considerations for liquid ionization calorimeters in a high rate environment. *Nucl. Instrum. Meth. A* **338** (1994) 467–497. doi:10.1016/0168-9002(94)91332-3.
[55] Madysa, Nico. AREUS: A Software Framework for ATLAS Readout Electronics Upgrade Simulation. *EPJ Web Conf.* **214** (2019) 02006. doi:10.1051/epjconf/201921402006.
[56] Coelho CN, Kuusela A, Li S, Zhuang H, Ngadiuba J, Aarrestad TK, 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. doi:10.1038/s42256-021-00356-5.
[57] Lai YT, Aoyama M, Bae H, Bähr S, Chang MC, Hayashii H, et al. Development of the level-1 track trigger with central drift chamber detector in belle II experiment and its performance in SuperKEKB 2019 phase 3 operation. *Journal of Instrumentation* **15** (2020) C06063–C06063. doi:10.1088/1748-0221/15/06/c06063.
[58] Baehr S, McCarney S, Meggendorfer F, Poehler J, Skambraks S, Unger K, et al. Low latency neural networks using heterogenous resources on fpga for the belle ii trigger (2019).
[59] Skambraks S, Bähr S, Becker J, Kiesling C, McCarney S, Meggendorfer F, et al. A 3d track finder for the belle II CDC l1 trigger. *Journal of Physics: Conference Series* **1525** (2020) 012102. doi:10.1088/1742-6596/1525/1/012102.

[60] Pezzullo G. The Mu2e experiment at Fermilab: a search for lepton flavor violation. *Nuclear and Particle Physics Proceedings* **285-286** (2017) 3 – 7. doi:https://doi.org/10.1016/j.nuclphysbps.2017.03.002. Sixth Workshop on Theory, Phenomenology and Experiments in Flavour Physics Interplay of Flavour Physics with Electroweak symmetry breaking.

[61] *CTD2020: The Track finder algorithm for the Trigger System of the Mu2e experiment at Fermilab* (Zenodo) (2020). doi:10.5281/zenodo.4088480.

[62] Abusalma F, Ambrose D, Artikov A, Bernstein R, Blazey GC, Bloise C, et al. Expression of Interest for Evolution of the Mu2e Experiment (2018).

[63] Wang T, Lu X, Wang A. A review: 3D printing of microwave absorption ceramics. *Int. J. Appl. Ceram. Technol.* **17** (2020) 2477–2491.

[64] Parekh DP, Ladd C, Panich L, Moussa K, Dickey MD. 3D printing of liquid metals as fugitive inks for fabrication of 3D microfluidic channels. *Lab Chip* **16** (2016) 1812–1820.

[65] Visser CW, Pohl R, Sun C, Römer GW, Huis in 't Veld B, Lohse D. Toward 3D printing of pure metals by laser-induced forward transfer. *Adv. Mater.* **27** (2015) 4087–4092.

[66] Ligon SC, Liska R, Stampfl J, Gurr M, Mülhaupt R. Polymers for 3D printing and customized additive manufacturing. *Chem. Rev.* **117** (2017) 10212–10290.

[67] Zarek M, Layani M, Cooperstein I, Sachyani E, Cohn D, Magdassi S. 3D printing of shape memory polymers for flexible electronic devices. *Adv. Mater.* **28** (2016) 4449–4454.

[68] Chrisey DB, Hubler GK. Pulsed laser deposition of thin films (1994). Accessed: 2021-2-1.

[69] Richter W. M. a. herman, h. sitter: Molecular beam epitaxy, fundamentals and current status, vol. 7 aus der reihe: Springer series in materials science, Springer-Verlag, berlin 1989. 382 seiten, ISBN 3-540-19075-9, preis: Hardcover DM 128,-. *Ber. Bunsenges. Phys. Chem.* **94** (1990) 542–542.

[70] Yoshino Y, Makino T, Katayama Y, Hata T. Optimization of zinc oxide thin film for surface acoustic wave filters by radio frequency sputtering. *Vacuum* **59** (2000) 538–545.

[71] Kelly PJ, Arnell RD. Magnetron sputtering: a review of recent developments and applications. *Vacuum* **56** (2000) 159–172.

[72] Marvel RE, Appavoo K, Choi BK, Nag J, Haglund RF. Electron-beam deposition of vanadium dioxide thin films. *Appl. Phys. A: Mater. Sci. Process.* **111** (2013) 975–981.

[73] George SM. Atomic layer deposition: an overview. *Chem. Rev.* **110** (2010) 111–131.

[74] Park JH, Sudarshan TS. *Chemical Vapor Deposition* (ASM International) (2001).

[75] Ojeda-G-P A, Schneider CW, Döbeli M, Lippert T, Wokaun A. Plasma plume dynamics, rebound, and recoating of the ablation target in pulsed laser deposition. *J. Appl. Phys.* **121** (2017) 135306.

[76] Egelhoff WF Jr, Jacob I I. Reflection high-energy electron diffraction (RHEED) oscillations at 77 K. *Phys. Rev. Lett.* **62** (1989) 921–924.

[77] Thomas JM. Design, synthesis, and in situ characterization of new solid catalysts. *Angew. Chem. Int. Ed Engl.* **38** (1999) 3588–3628.

[78] Langereis E, Knoops HCM, Mackus AJM, Roozeboom F, Van de Sanden MCM, Kessels WMM. Synthesis and in situ characterization of low-resistivity ta N x films by remote plasma atomic layer deposition. *J. Appl. Phys.* **102** (2007) 083517.

[79] Hansen TN, Schou J, Lunney JG. Langmuir probe study of plasma expansion in pulsed laser ablation. *Appl. Phys. A: Mater. Sci. Process.* **69** (1999) S601–S604.

[80] Cooks RG, Yan X. Mass spectrometry for synthesis and analysis. *Annu. Rev. Anal. Chem.* **11** (2018) 1–28.

[81] Termopoli V, Torrisi E, Famiglini G, Palma P, Zappia G, Cappiello A, et al. Mass spectrometry based approach for organic synthesis monitoring. *Anal. Chem.* **91** (2019) 11916–11922.

[82] Dauchot JP, Edart S, Wautelet M, Hecq M. Synthesis of zirconium nitride films monitored by in situ soft x-ray spectrometry. *Vacuum* **46** (1995) 927–930.

[83] Aubriet F, Chaoui N, Chety R, Maunit B, Millon E, Muller JF. Laser ablation mass spectrometry: a tool to investigate matter transfer processes during pulsed-laser deposition experiments. *Appl. Surf. Sci.* **186** (2002) 282–287.

[84] Trigub MV, Platonov VV, Evtushenko GS, Osipov VV, Evtushenko TG. Laser monitors for high speed imaging of materials modification and production. *Vacuum* **143** (2017) 486–490.

[85] Ojeda-G-P A, Döbeli M, Lippert T. Influence of plume properties on thin film composition in pulsed laser deposition. *Advanced Materials Interfaces* **5** (2018) 1701062.

[86] Butler KT, Davies DW, Cartwright H, Isayev O, Walsh A. Machine learning for molecular and materials science. *Nature* **559** (2018) 547–555.

[87] Schmidt J, Marques MRG, Botti S, Marques MAL. Recent advances and applications of machine learning in solid-state materials science. *npj Computational Materials* **5** (2019) 83.

[88] Ramprasad R, Batra R, Pilania G, Mannodi-Kanakkithodi A, Kim C. Machine learning in materials informatics: recent applications and prospects. *npj Computational Materials* **3** (2017) 54.

[89] Oxley MP, Ziatdinov M, Dyck O, Lupini AR, Vasudevan R, Kalinin SV. Probing atomic-scale symmetry breaking by rotationally invariant machine learning of multidimensional electron scattering (2020).

[90] Kalinin SV, Zhang S, Valleti M, Pyles H, Baker D, De Yoreo JJ, et al. Exploring particle dynamics during self-organization processes via rotationally invariant latent representations (2020).

[91] Smidt TE, Geiger M, Miller BK. Finding symmetry breaking order parameters with euclidean neural networks. *Physical Review Research* **3** (2021).

[92] Smidt TE. Euclidean symmetry and equivariance in machine learning. *Trends in Chemistry* (2020).

[93] Kaheman K, Kutz JN, Brunton SL. SINDy-PI: a robust algorithm for parallel implicit sparse identification of nonlinear dynamics. *Proc. Math. Phys. Eng. Sci.* **476** (2020) 20200279.

[94] de Silva BM, Champion K, Quade M, others. PySINDy: a python package for the sparse identification of nonlinear dynamics from data. *arXiv preprint arXiv* (2020).

[95] Champion K, Lusch B, Kutz JN, Brunton SL. Data-driven discovery of coordinates and governing equations. *Proc. Natl. Acad. Sci. U. S. A.* **116** (2019) 22445–22451.

[96] Kaheman K, Kaiser E, Strom B, Nathan Kutz J, Brunton SL. Learning discrepancy models from experimental data (2019).

[97] Provence SR, Thapa S, Paudel R, Truttmann TK, Prakash A, Jalan B, et al. Machine learning analysis of perovskite oxides grown by molecular beam epitaxy. *Phys. Rev. Materials* **4** (2020) 083807.

[98] Trejo O, Dadlani AL, De La Paz F, Acharya S, Kravec R, Nordlund D, et al. Elucidating the evolving atomic structure in atomic layer deposition reactions with in situ XANES and machine learning. *Chem. Mater.* **31** (2019) 8937–8947.

[99] MacLeod BP, Parlane FGL, Morrissey TD, Häse F, Roch LM, Dettelbach KE, et al. Self-driving laboratory for accelerated discovery of thin-film materials. *Sci Adv* **6** (2020) eaaz8867.

[100] Langner S, Häse F, Perea JD, Stubhan T, Hauch J, Roch LM, et al. Film fabrication techniques: Beyond ternary OPV: High-throughput experimentation and self-driving laboratories optimize multicomponent systems (adv. mater. 14/2020). *Adv. Mater.* **32** (2020) 2070110.

[101] Binnig G, Quate CF, Gerber C. Atomic force microscope. *Phys. Rev. Lett.* **56** (1986) 930–933.

[102] Benstetter G, Biberger R, Liu D. A review of advanced scanning probe microscope analysis of functional films and semiconductor devices. *Thin Solid Films* **517** (2009) 5100–5105.

[103] King WP. Design analysis of heated atomic force microscope cantilevers for nanotopography measurements. *J. Micromech. Microeng.* **15** (2005) 2441.

[104] Oberhauser AF, Badilla-Fernandez C, Carrion-Vazquez M, Fernandez JM. The mechanical hierarchies of fibronectin observed with single-molecule AFM. *J. Mol. Biol.* **319** (2002) 43.

[105] Ariyaratne A, Bluvstein D, Myers BA, Jayich ACB. Nanoscale electrical conductivity imaging using a nitrogen-vacancy center in diamond. *Nat. Commun.* **9** (2018) 2406.

[106] Seidel J, Maksymovych P, Batra Y, Katan A, Yang SY, He Q, et al. Domain wall conductivity in la-doped BiFeO3. *Phys. Rev. Lett.* **105** (2010) 197603.

[107] Gómez-Navarro C, De Pablo PJ, Gómez-Herrero J, Biel B, Garcia-Vidal FJ, Rubio A, et al. Tuning the conductance of single-walled carbon nanotubes by ion irradiation in the anderson localization regime. *Nature Materials* **4** (2005) 534–539.

[108] Jesse S, Kalinin SV. Band excitation in scanning probe microscopy: sines of change. *J. Phys. D Appl. Phys.* **44** (2011) 464006.

[109] Jesse S, Kumar A, Arruda TM, Kim Y, Kalinin SV, Ciucci F. Electrochemical strain microscopy: Probing ionic and electrochemical phenomena in solids at the nanometer level. *MRS Bull.* **37** (2012) 651–658.

[110] Kazakova O, Puttock R, Barton C, Corte-León H, Jaafar M, Neu V, et al. Frontiers of magnetic force microscopy. *J. Appl. Phys.* **125** (2019) 060901.

[111] Casola F, van der Sar T, Yacoby A. Probing condensed matter physics with magnetometry based on nitrogen-vacancy centres in diamond. *Nature Reviews Materials* **3** (2018) 17088.

[112] Benaglia S, Gisbert VG, Perrino AP, Amo CA, Garcia R. Fast and high-resolution mapping of elastic properties of biomolecules and polymers with bimodal AFM. *Nat. Protoc.* **13** (2018) 2890–2907.

[113] Holstad TS, Ræder TM, Evans DM, Småbråten DR, Krohns S, Schaab J, et al. Application of a long short-term memory for deconvoluting conductance contributions at charged ferroelectric domain walls. *Npj Comput. Mater.* **6** (2020).

[114] Jesse S, Collins L, Neumayer S, Somnath S, Kalinin SV. Dynamic modes in kelvin probe force microscopy: Band excitation and G-Mode. Sadewasser S, Glatzel T, editors, *Kelvin Probe Force Microscopy: From Single Charge Detection to Device Characterization* (Cham: Springer International Publishing) (2018), 49–99.

[115] Somnath S, Belianinov A, Kalinin SV, Jesse S. Full information acquisition in piezoresponse force microscopy. *Appl. Phys. Lett.* **107** (2015) 263102.

[116] Ziatdinov M, Kim D, Neumayer S, Vasudevan RK, Collins L, Jesse S, et al. Imaging mechanism for hyperspectral scanning probe microscopy via gaussian process modelling. *npj Computational Materials* **6** (2020) 21.

[117] Collins L, Vasudevan RK, Sehirlioglu A. Visualizing charge transport and nanoscale electrochemistry by hyperspectral kelvin probe force microscopy. *ACS Appl. Mater. Interfaces* **12** (2020) 33361–33369.

[118] Kalinin SV, Kelley K, Vasudevan RK, Ziatdinov M. Toward decoding the relationship between domain structure and functionality in ferroelectrics via hidden latent variables. *ACS Applied Materials & Interfaces* **13** (2021) 1693–1703.

[119] Collins L, Somnath S, Kalinin SV, Belianinov A. Scanning probe microscopy in the information age. *Handbook on Big Data and Machine Learning in the Physical Sciences* (World Scientific), World Scientific Series on Emerging Technologies (2020), 231–282.

[120] Griffin LA, Gaponenko I, Bassiri-Gharb N. Better, faster, and less biased machine learning: Electromechanical switching in ferroelectric thin films. *Adv. Mater.* (2020) e2002425.

[121] Agar JC, Naul B, Pandya S, van der Walt S, Maher J, Ren Y, et al. Revealing ferroelectric switching character using deep recurrent neural networks. *Nat. Commun.* **10** (2019) 4809.

[122] Borodinov N, Neumayer S, Kalinin SV, Ovchinnikova OS, Vasudevan RK, Jesse S. Deep neural networks for understanding noisy data applied to physical property extraction in scanning probe microscopy. *npj Computational Materials* **5** (2019) 25.

[123] About Fermilab (2021). https://www.fnal.gov/pub/about/index.html.

[124] Fermilab. DOE CD-0 Mission Need for Accelerator Controls Operations Research Network (ACORN) (2021). https://acorn-docdb.fnal.gov/cgi-bin/sso/ShowDocument?docid=10.

[125] Fermilab operations department booster rookie book (2021).

[126] John JS, Herwig C, Kafkes D, Pellico WA, Perdue GN, Quintero-Parra A, et al. Real-time artificial intelligence for accelerator control: A study at the fermilab booster (2021).

[127] Data, artificial intelligence, and machine learning at doe scientific user facilities (2020).

[128] Bartoszek L, Barnes E, Miller JP, Mott J, Palladino A, Quirk J, et al. Mu2e technical design report (2015).

[129] Seiya K, Hazelwood KJ, Ibrahim MA, Nagaslaev VP, Nicklaus DJ, Schupbach BA, et al. Accelerator real-time edge ai for distributed systems (reads) proposal (2021).

[130] Adamson P, Aliaga L, Ambrose D, Anfimov N, Antoshkin A, Arrieta-Diaz E, et al. Constraints on oscillation parameters from ν_e appearance and ν_μ disappearance in nova. *Physical Review Letters* **118** (2017). doi:10.1103/physrevlett.118.231801.

[131] Abi B, Acciarri R, Acero M, Adamov G, Adams D, Adinolfi M, et al. Neutrino interaction classification with a convolutional neural network in the DUNE far detector. *Physical Review D* **102** (2020). doi:10.1103/physrevd.102.092003.

[132] et al GP. Reducing model bias in a deep learning classifier using domain adversarial neural networks in the minerva experiment. *Journal of Instrumentation* **13** (2018) P11020–P11020. doi:10.1088/1748-0221/13/11/p11020.

[133] Psihas F, Groh M, Tunnell C, Warburton K. A review on machine learning for neutrino experiments. *International Journal of Modern Physics A* **35** (2020) 2043005. doi:10.1142/s0217751x20430058.

[134] Abbott B, et al. Multi-messenger Observations of a Binary Neutron Star Merger. *Astrophys. J. Lett.* **848** (2017) L12. doi:10.3847/2041-8213/aa91c9.

[135] Aartsen M, et al. Multimessenger observations of a flaring blazar coincident with high-energy neutrino IceCube-170922A. *Science* **361** (2018) eaat1378. doi:10.1126/science.aat1378.

[136] Graham MJ, Ford KES, McKernan B, Ross NP, Stern D, Burdge K, et al. Candidate electromagnetic counterpart to the binary black hole merger gravitational-wave event s190521g. *Phys. Rev. Lett.* **124** (2020). doi:10.1103/physrevlett.124.251102.

[137] Scholberg K. Supernova Neutrino Detection. *Ann. Rev. Nucl. Part. Sci.* **62** (2012) 81–103. doi:10.1146/annurev-nucl-102711-095006.

[138] Mirizzi A, Tamborra I, Janka HT, Saviano N, Scholberg K, Bollig R, et al. Supernova Neutrinos: Production, Oscillations and Detection. *Riv. Nuovo Cim.* **39** (2016) 1–112. doi:10.1393/ncr/i2016-10120-8.

[139] Kistler MD, Haxton WC, Yüksel H. Tomography of Massive Stars from Core Collapse to Supernova Shock Breakout. *Astrophys. J.* **778** (2013) 81. doi:10.1088/0004-637X/778/1/81.

[140] Antonioli P, Fienberg RT, Fleurot F, Fukuda Y, Fulgione W, Habig A, et al. SNEWS: The supernova early warning system. *New J. Phys.* **6** (2004) 114. doi:10.1088/1367-2630/6/1/114.

[141] Al Kharusi S, et al. SNEWS 2.0: A Next-Generation SuperNova Early Warning System for Multi-messenger Astronomy (2020).

[142] Abi B, et al. Supernova Neutrino Burst Detection with the Deep Underground Neutrino Experiment (2020).

[143] Abi B, et al. Deep Underground Neutrino Experiment (DUNE), Far Detector Technical Design Report, Volume II DUNE Physics (2020).

[144] Roeth, A J. Supernova Neutrino Pointing with DUNE (2020). https://indico.cern.ch/event/868940/contributions/3813598/attachments/2081577/3496427/Point_Res_ICHEP_2020_07_AJRoeth.pdf.

[145] Drielsma F, Terao K, Dominé L, Koh DH. Scalable, End-to-End, Deep-Learning-Based Data Reconstruction Chain for Particle Imaging Detectors. *34th Conference on Neural Information Processing Systems* (2021).

[146] Qian Z, et al. Vertex and Energy Reconstruction in JUNO with Machine Learning Methods (2021).

[147] Acciarri R, et al. Cosmic Background Removal with Deep Neural Networks in SBND (2020).

[148] Abratenko P, et al. A Convolutional Neural Network for Multiple Particle Identification in the MicroBooNE Liquid Argon Time Projection Chamber (2020).

[149] Wang M, Yang T, Acosta Flechas M, Harris P, Hawks B, Holzman B, et al. GPU-accelerated machine learning inference as a service for computing in neutrino experiments (2020).

[150] Alexander J, Battaglieri M, Echenard B, Essig R, Graham M, Izaguirre E, et al. Dark Sectors 2016 Workshop: Community Report (2016).

[151] Schumann M. Direct detection of WIMP dark matter: concepts and status. *Journal of Physics G: Nuclear and Particle Physics* **46** (2019) 103003. doi:10.1088/1361-6471/ab2ea5.

[152] Jungman G, Kamionkowski M, Griest K. Supersymmetric dark matter. *Physics Reports* **267** (1996) 195–373. doi:10.1016/0370-1573(95)00058-5.

[153] Bernabei R, Belli P, Cappella F, Caracciolo V, Castellano S, Cerulli R, et al. Final model independent result of DAMA/LIBRA–phase1. *The European Physical Journal C* **73** (2013). doi:10.1140/epjc/s10052-013-2648-7.

[154] Bœhm C, Fayet P. Scalar dark matter candidates. *Nuclear Physics B* **683** (2004) 219–263. doi:10.1016/j.nuclphysb.2004.01.015.

[155] Weinberg DH, Bullock JS, Governato F, Kuzio de Naray R, Peter AHG. Cold dark matter: Controversies on small scales. *Proceedings of the National Academy of Sciences* **112** (2015) 12249–12255. doi:10.1073/pnas.1308716112.

[156] Peccei RD. *The Strong CP problem and axions* (Springer, Berlin, Heidelberg), vol. 741 (2008), 3–17. doi:10.1007/978-3-540-73518-2_1.

[157] Svrcek P, Witten E. Axions In String Theory. *JHEP* **06** (2006) 051. doi:10.1088/1126-6708/2006/06/051.

[158] Holdom B. Two U(1)'s and Epsilon Charge Shifts. *Phys. Lett.* **166B** (1986) 196–198. doi:10.1016/0370-2693(86)91377-8.

[159] Khosa CK, Mars L, Richards J, Sanz V. Convolutional neural networks for direct detection of dark matter. *Journal of Physics G: Nuclear and Particle Physics* **47** (2020) 095201. doi:10.1088/1361-6471/ab8e94.

[160] Szydagis M, et al. A Review of Basic Energy Reconstruction Techniques in Liquid Xenon and Argon Detectors for Dark Matter and Neutrino Physics Using NEST (2021).

[161] Simola U, Pelssers B, Barge D, Conrad J, Corander J. Machine learning accelerated likelihood-free event reconstruction in dark matter direct detection. *Journal of Instrumentation* **14** (2019) P03004–P03004. doi:10.1088/1748-0221/14/03/p03004.

[162] Agnese R, Anderson A, Aramaki T, Arnquist I, Baker W, Barker D, et al. Projected sensitivity of the SuperCDMS SNOLAB experiment. *Physical Review D* **95** (2017). doi:10.1103/physrevd.95.082002.

[163] Accardi A, et al. Electron Ion Collider: The Next QCD Frontier: Understanding the glue that binds us all. *Eur. Phys. J. A* **52** (2016) 268. doi:10.1140/epja/i2016-16268-9.

[164] Aprahamian A, et al. Reaching for the horizon: The 2015 long range plan for nuclear science (2015).

[165] National Academies of Sciences, Engineering, and Medicine. An Assessment of U.S.-Based Electron-Ion Collider Science (2018). doi:https://doi.org/10.17226/25171.

[166] Abdul Khalek R, et al. Science Requirements and Detector Concepts for the Electron-Ion Collider: EIC Yellow Report (2021).

[167] Bedaque P, et al. A.I. for nuclear physics. *Eur. Phys. J. A* **57** (2021) 100. doi:10.1140/epja/s10050-020-00290-x.

[168] Abbott BP, Abbott R, Abbott TD, Abernathy MR, Acernese F, Ackley K, et al. Observation of gravitational waves from a binary black hole merger. *Phys. Rev. Lett.* **116** (2016) 061102. doi:10.1103/PhysRevLett.116.061102.

[169] Harry GM, LIGO Scientific Collaboration. Advanced LIGO: the next generation of gravitational wave detectors. *Class. Quantum Grav.* **27** (2010) 084006. doi:10.1088/0264-9381/27/8/084006.

[170] Acernese F, Agathos M, Agatsuma K, Aisa D, Allemandou N, Allocca A, et al. Advanced virgo: a second-generation interferometric gravitational wave detector. *Classical and Quantum Gravity* **32** (2014) 024001. doi:10.1088/0264-9381/32/2/024001.

[171] Affeldt C, Danzmann K, Dooley KL, Grote H, Hewitson M, Hild S, et al. Advanced techniques in GEO 600. *Classical and Quantum Gravity* **31** (2014) 224002. doi:10.1088/0264-9381/31/22/224002.

[172] Aso Y, Michimura Y, Somiya K, Ando M, Miyakawa O, Sekiguchi T, et al. Interferometer design of the kagra gravitational wave detector. *Phys. Rev. D* **88** (2013) 043007. doi:10.1103/PhysRevD.88.043007.

[173] Abbott BP, Abbott R, Abbott TD, Acernese F, Ackley K, Adams C, et al. Gw170817: Observation of gravitational waves from a binary neutron star inspiral. *Phys. Rev. Lett.* **119** (2017) 161101. doi:10.1103/PhysRevLett.119.161101.

[174] Vaseghi SV. *Wiener Filters* (John Wiley & Sons, Ltd) (2001), 178–204.

[175] Sathyaprakash BS, Dhurandhar SV. Choice of filters for the detection of gravitational waves from coalescing binaries. *Phys. Rev. D* **44** (1991) 3819–3834. doi:10.1103/PhysRevD.44.3819.

[176] Abbott B, Abbott R, Abbott T, Abernathy M, Acernese F, Ackley K, et al. Properties of the binary black hole merger gw150914. *Physical Review Letters* **116** (2016). doi:10.1103/physrevlett.116.241102.

[177] Cuoco E, Powell J, Cavaglià M, Ackley K, Bejger M, Chatterjee C, et al. Enhancing gravitational-wave science with machine learning. *Machine Learning: Science and Technology* **2** (2020) 011002. doi:10.1088/2632-2153/abb93a.

[178] Gabbard H, Williams M, Hayes F, Messenger C. Matching matched filtering with deep networks for gravitational-wave astronomy. *Phys. Rev. Lett.* **120** (2018) 141103. doi:10.1103/PhysRevLett.120.141103.

[179] Kim K, Harry IW, Hodge KA, Kim YM, Lee CH, Lee HK, et al. Application of artificial neural network to search for gravitational-wave signals associated with short gamma-ray bursts. *Classical and Quantum Gravity* **32** (2015) 245002. doi:10.1088/0264-9381/32/24/245002.

[180] Kim K, Li TGF, Lo RKL, Sachdev S, Yuen RSH. Ranking candidate signals with machine learning in low-latency searches for gravitational waves from compact binary mergers. *Phys. Rev. D* **101** (2020) 083006. doi:10.1103/PhysRevD.101.083006.

[181] George D, Huerta E. Deep learning for real-time gravitational wave detection and parameter estimation: Results with advanced ligo data. *Physics Letters B* **778** (2018) 64–70. doi:10.1016/j.physletb.2017.12.053.

[182] Gebhard TD, Kilbertus N, Harry I, Schölkopf B. Convolutional neural networks: A magic bullet for gravitational-wave detection? *Physical Review D* **100** (2019). doi:10.1103/physrevd.100.063015.

[183] Astone P, Cerdá-Durán P, Di Palma I, Drago M, Muciaccia F, Palomba C, et al. New method to observe gravitational waves emitted by core collapse supernovae. *Physical Review D* **98** (2018). doi:10.1103/physrevd.98.122002.

[184] Chan ML, Heng IS, Messenger C. Detection and classification of supernova gravitational wave signals: A deep learning approach. *Physical Review D* **102** (2020). doi:10.1103/physrevd.102.043022.

[185] Iess A, Cuoco E, Morawski F, Powell J. Core-collapse supernova gravitational-wave search and deep learning classification. *Machine Learning: Science and Technology* **1** (2020) 025014. doi:10.1088/2632-2153/ab7d31.

[186] Dreissigacker C, Sharma R, Messenger C, Zhao R, Prix R. Deep-learning continuous gravitational waves. *Physical Review D* **100** (2019). doi:10.1103/physrevd.100.044009.

[187] Beheshtipour B, Papa M. Deep learning for clustering of continuous gravitational wave candidates. *Physical Review D* **101** (2020). doi:10.1103/physrevd.101.064009.

[188] Moreno EA, Vlimant JR, Spiropulu M, Borzyszkowski B, Pierini M. Source-agnostic gravitational-wave detection with recurrent autoencoders. *arXiv preprint arXiv:2107.12698* (2021).

[189] Que Z, Wang E, Marikar U, Moreno E, Ngadiuba J, Javed H, et al. Accelerating recurrent neural networks for gravitational wave experiments. *arXiv preprint arXiv:2106.14089* (2021).

[190] Shen H, Huerta EA, Zhao Z, Jennings E, Sharma H. Deterministic and bayesian neural networks for low-latency gravitational wave parameter estimation of binary black hole mergers (2019).

[191] Gabbard H, Messenger C, Heng IS, Tonolini F, Murray-Smith R. Bayesian parameter estimation using conditional variational autoencoders for gravitational-wave astronomy (2020).

[192] Chua AJK, Vallisneri M. Learning bayesian posteriors with neural networks for gravitational-wave inference. *Phys. Rev. Lett.* **124** (2020) 041102. doi:10.1103/PhysRevLett.124.041102.

[193] Zevin M, Coughlin S, Bahaadini S, Besler E, Rohani N, Allen S, et al. Gravity spy: integrating advanced ligo detector characterization, machine learning, and citizen science. *Classical and Quantum Gravity* **34** (2017) 064003. doi:10.1088/1361-6382/aa5cea.

[194] Razzano M, Cuoco E. Image-based deep learning for classification of noise transients in gravitational wave detectors. *Classical and Quantum Gravity* **35** (2018) 095016. doi:10.1088/1361-6382/aab793.

[195] Biswas R, Blackburn L, Cao J, Essick R, Hodge KA, Katsavounidis E, et al. Application of machine learning algorithms to the study of noise artifacts in gravitational-wave data. *Phys. Rev. D* **88** (2013) 062003. doi:10.1103/PhysRevD.88.062003.

[196] Vajente G, Huang Y, Isi M, Driggers JC, Kissel JS, Szczepańczyk MJ, et al. Machine-learning nonstationary noise out of gravitational-wave detectors. *Phys. Rev. D* **101** (2020) 042003. doi:10.1103/PhysRevD.101.042003.

[197] Ormiston R, Nguyen T, Coughlin M, Adhikari RX, Katsavounidis E. Noise reduction in gravitational-wave data via deep learning. *Physical Review Research* **2** (2020). doi:10.1103/physrevresearch.2.033066.

[198] Chen PHC, Gadepalli K, MacDonald R, Liu Y, Kadowaki S, Nagpal K, et al. An augmented reality microscope with real-time artificial intelligence integration for cancer diagnosis. *Nat. Med.* **25** (2019) 1453–1457.

[199] Christiansen EM, Yang SJ, Ando DM, Javaherian A, Skibinski G, Lipnick S, et al. In silico labeling: Predicting fluorescent labels in unlabeled images. *Cell* **173** (2018) 792–803.e19.

[200] Wang S, Zhou Y, Qin X, Nair S, Huang X, Liu Y. Label-free detection of rare circulating tumor cells by image analysis and machine learning. *Sci. Rep.* **10** (2020) 12226.

[201] Siu DMD, Lee KCM, Lo MCK, Stassen SV, Wang M, Zhang IZQ, et al. Deep-learning-assisted biophysical imaging cytometry at massive throughput delineates cell population heterogeneity. *Lab Chip* (2020).

[202] Tang WB, Ji Y, Zhang MM, Chen ZY, Xu YY, Wang YW. A rapid detection method for morphological characteristics of biological cells based on phase imaging. *Biomed Res. Int.* **2018** (2018) 4651639.

[203] Li C, Liu B, Kang B, Liu Z, Liu Y, Chen C, et al. SciBet as a portable and fast single cell type identifier. *Nat. Commun.* **11** (2020) 1818.

[204] Nitta N, Sugimura T, Isozaki A, Mikami H, Hiraki K, Sakuma S, et al. Intelligent Image-Activated cell sorting. *Cell* **175** (2018) 266–276.e13.

[205] Sakellaropoulos T, Vougas K, Narang S, Koinis F, Kotsinas A, Polyzos A, et al. A deep learning framework for predicting response to therapy in cancer. *Cell Rep.* **29** (2019) 3367–3373.e4.

[206] Shukla S, Hassan MF, Khan MK, Jung LT, Awang A. An analytical model to minimize the latency in healthcare internet-of-things in fog computing environment. *PLoS One* **14** (2019) e0224934.

[207] Adarsh P, Rathi P, Kumar M. YOLO v3-tiny: Object detection and recognition using one stage improved model. *2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS)* (2020), 687–694.

[208] Satpathy S, Mohan P, Das S, Debbarma S. A new healthcare diagnosis system using an IoT-based fuzzy classifier with FPGA. *J. Supercomput.* **76** (2020) 5849–5861.

[209] Chang YW, Sheu TWH. GPU acceleration of a patient-specific airway image segmentation and its assessment (2020).

[210] Zhang S, Cao J, Zhang Q, Zhang Q, Zhang Y, Wang Y. An FPGA-Based reconfigurable CNN accelerator for YOLO. *2020 IEEE 3rd International Conference on Electronics Technology (ICET)* (2020), 74–78.

[211] Chen M, Yang J, Zhou J, Hao Y, Zhang J, Youn C. 5G-Smart diabetes: Toward personalized diabetes diagnosis with healthcare big data clouds. *IEEE Commun. Mag.* **56** (2018) 16–23.

[212] Zhang Y, Chen G, Du H, Yuan X, Kadoch M, Cheriet M. Real-Time remote health monitoring system driven by 5G MEC-IoT. *Electronics* **9** (2020) 1753.

[213] Morocho-Cayamcela ME, Lee H, Lim W. Machine learning for 5G/B5G mobile and wireless communications: Potential, limitations, and future directions. *IEEE Access* **7** (2019) 137184–137206.

[214] Lee H, Lee EJ, Ham S, Lee HB, Lee JS, Kwon SU, et al. Machine learning approach to identify stroke within 4.5 hours. *Stroke* **51** (2020) 860–866.

[215] Nafee T, Gibson CM, Travis R, Yee MK, Kerneis M, Chi G, et al. Machine learning to predict venous thrombosis in acutely ill medical patients. *Res Pract Thromb Haemost* **4** (2020) 230–237.

[216] Nogueira-Rodríguez A, Domínguez-Carbajales R, López-Fernández H, Iglesias Á, Cubiella J, Fdez-Riverola F, et al. Deep neural networks approaches for detecting and classifying colorectal polyps. *Neurocomputing* (2020).

[217] Bagheri E, Jin J, Dauwels J, Cash S, Westover MB. A fast machine learning approach to facilitate the detection of interictal epileptiform discharges in the scalp electroencephalogram. *J. Neurosci. Methods* **326** (2019) 108362.

[218] Horie Y, Yoshio T, Aoyama K, Yoshimizu S, Horiuchi Y, Ishiyama A, et al. Diagnostic outcomes of esophageal cancer by artificial intelligence using convolutional neural networks. *Gastrointest. Endosc.* **89** (2019) 25–32.

[219] Volkov M, Hashimoto DA, Rosman G, Meireles OR, Rus D. Machine learning and coresets for automated real-time video segmentation of laparoscopic and robot-assisted surgery. *2017 IEEE International Conference on Robotics and Automation (ICRA)* (IEEE) (2017).

[220] Choi B, Jo K, Choi S, Choi J. Surgical-tools detection based on convolutional neural network in laparoscopic robot-assisted surgery. *2017 39th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)* (2017), 1756–1759.

[221] Tonutti M, Gras G, Yang GZ. A machine learning approach for real-time modelling of tissue deformation in image-guided neurosurgery. *Artif. Intell. Med.* **80** (2017) 39–47.

[222] Baker F, Ainsworth SR, Dye JT, Crammer C, Thun MJ, Hoffmann D, et al. Health risks associated with cigar smoking. *Jama* **284** (2000) 735–740.

[223] Klesges RC, Meyers AW, Klesges LM, LaVasque ME. Smoking, body weight, and their effects on smoking behavior: a comprehensive review of the literature. *Psychological bulletin* **106** (1989) 204.

[224] Sokol MC, McGuigan KA, Verbrugge RR, Epstein RS. Impact of medication adherence on hospitalization risk and healthcare cost. *Medical care* (2005) 521–530.

[225] White A, Hingson R. The burden of alcohol use: excessive alcohol consumption and related consequences among college students. *Alcohol research: current reviews* (2013).

[226] Althubaiti A. Information bias in health research: definition, pitfalls, and adjustment methods. *Journal of multidisciplinary healthcare* **9** (2016) 211.

[227] Dong Y, Hoover A, Scisco J, Muth E. A new method for measuring meal intake in humans via automated wrist motion tracking. *Applied psychophysiology and biofeedback* **37** (2012) 205–215.

[228] Parate A, Chiu MC, Chadowitz C, Ganesan D, Kalogerakis E. Risq: Recognizing smoking gestures with inertial sensors on a wristband. *Proceedings of the 12th annual international conference on Mobile systems, applications, and services* (2014), 149–161.

[229] Ali AA, Hossain SM, Hovsepian K, Rahman MM, Plarre K, Kumar S. mpuff: automated detection of cigarette smoking puffs from respiration measurements. *Proceedings of the 11th international conference on Information Processing in Sensor Networks* (2012), 269–280.

[230] Sen S, Subbaraju V, Misra A, Balan R, Lee Y. Annapurna: An automated smartwatch-based eating detection and food journaling system. *Pervasive and Mobile Computing* **68** (2020) 101259.

[231] Bi S, Wang T, Tobias N, Nordrum J, Wang S, Halvorsen G, et al. Auracle: Detecting eating episodes with an ear-mounted sensor. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* **2** (2018). doi:10.1145/3264902.

[232] Zhang S, Zhao Y, Nguyen DT, Xu R, Sen S, Hester J, et al. Necksense: A multi-sensor necklace for detecting eating activities in free-living conditions. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* **4** (2020) 1. doi:10.1145/3397313.

[233] Mishra V, Pope G, Lord S, Lewia S, Lowens B, Caine K, et al. Continuous detection of physiological stress with commodity hardware. *ACM transactions on computing for healthcare* **1** (2020) 1–30.

[234] Holz C, Wang EJ. Glabella: Continuously sensing blood pressure behavior using an unobtrusive wearable device. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* **1** (2017). doi:10.1145/3132024.

[235] Pham N, Dinh T, Raghebi Z, Kim T, Bui N, Nguyen P, et al. Wake: a behind-the-ear wearable system for microsleep detection. *Proceedings of the 18th International Conference on Mobile Systems, Applications, and Services* (2020), 404–418.

[236] Chun KS, Bhattacharya S, Dolbear C, Kashanchi J, Thomaz E. Intraoral temperature and inertial sensing in automated dietary assessment: a feasibility study. *Proceedings of the 2020 International Symposium on Wearable Computers* (2020), 27–31.

[237] Bui N, Pham N, Barnitz JJ, Zou Z, Nguyen P, Truong H, et al. ebp: A wearable system for frequent and comfortable blood pressure monitoring from user's ear. *The 25th Annual International Conference on Mobile Computing and Networking* (2019), 1–17.

[238] Li T, Bai D, Prioleau T, Bui N, Vu T, Zhou X. Noninvasive glucose monitoring using polarized light. *Proceedings of the 18th Conference on Embedded Networked Sensor Systems* (2020), 544–557.

[239] Echterhoff JM, Wang EJ. Par: Personal activity radius camera view for contextual sensing. *arXiv preprint arXiv:2008.07204* (2020).

[240] Bedri A, Li D, Khurana R, Bhuwalka K, Goel M. Fitbyte: Automatic diet monitoring in unconstrained situations using multimodal sensing on eyeglasses. *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems* (New York, NY, USA: Association for Computing Machinery) (2020), CHI '20, 1–12. doi:10.1145/3313831.3376869.

[241] Nahum-Shani I, Smith SN, Spring BJ, Collins LM, Witkiewitz K, Tewari A, et al. Just-in-time adaptive interventions (jitais) in mobile health: key components and design principles for ongoing health behavior support. *Annals of Behavioral Medicine* **52** (2018) 446–462.

[242] Thomas JG, Bond DS. Behavioral response to a just-in-time adaptive intervention (jitai) to reduce sedentary behavior in obese adults: Implications for jitai optimization. *Health Psychology* **34** (2015) 1261.

[243] Sadek I, Rehman SU, Codjo J, Abdulrazak B. Privacy and security of iot based healthcare systems: Concerns, solutions, and recommendations. Pagán J, Mokhtari M, Aloulou H, Abdulrazak B, Cabrera MF, editors, *How AI Impacts Urban Living and Public Health* (Cham: Springer International Publishing) (2019), 3–17.

[244] Rieke N, Hancox J, Li W, Milletari F, Roth HR, Albarqouni S, et al. The future of digital health with federated learning. *NPJ digital medicine* **3** (2020) 1–7.

[245] Riess AG, Filippenko AV, Challis P, Clocchiatti A, Diercks A, Garnavich PM, et al. Observational Evidence from Supernovae for an Accelerating Universe and a Cosmological Constant. *Astron. J.* **116** (1998) 1009–1038. doi:10.1086/300499.

[246] Perlmutter S, Aldering G, Goldhaber G, Knop RA, Nugent P, Castro PG, et al. Measurements of Ω and Λ from 42 High-Redshift Supernovae. *Astrophys. J.* **517** (1999) 565–586. doi:10.1086/307221.

[247] Betoule M, Kessler R, Guy J, Mosher J, Hardin D, Biswas R, et al. Improved cosmological constraints from a joint analysis of the SDSS-II and SNLS supernova samples. *Astron. Astrophys.* **568** (2014) A22. doi:10.1051/0004-6361/201423413.

[248] Scolnic DM, Jones DO, Rest A, Pan YC, Chornock R, Foley RJ, et al. The Complete Light-curve Sample of Spectroscopically Confirmed SNe Ia from Pan-STARRS1 and Cosmological Constraints

from the Combined Pantheon Sample. *Astrophys. J.* **859** (2018) 101. doi:10.3847/1538-4357/aab9bb.

[249] Abbott TMC, Allam S, Andersen P, Angus C, Asorey J, Avelino A, et al. First Cosmology Results using Type Ia Supernovae from the Dark Energy Survey: Constraints on Cosmological Parameters. *Astrophys. J. Lett.* **872** (2019) L30. doi:10.3847/2041-8213/ab04fa.

[250] Fixsen DJ, Cheng ES, Gales JM, Mather JC, Shafer RA, Wright EL. The Cosmic Microwave Background Spectrum from the Full COBE FIRAS Data Set. *Astrophys. J.* **473** (1996) 576. doi:10.1086/178173.

[251] Spergel DN, Verde L, Peiris HV, Komatsu E, Nolta MR, Bennett CL, et al. First-Year Wilkinson Microwave Anisotropy Probe (WMAP) Observations: Determination of Cosmological Parameters. *The Astrophysical Journal Supplement Series* **148** (2003) 175–194. doi:10.1086/377226.

[252] Komatsu E, Smith KM, Dunkley J, Bennett CL, Gold B, Hinshaw G, et al. Seven-year Wilkinson Microwave Anisotropy Probe (WMAP) Observations: Cosmological Interpretation. *The Astrophysical Journal Supplement Series* **192** (2011) 18. doi:10.1088/0067-0049/192/2/18.

[253] Planck Collaboration, Aghanim N, Arnaud M, Ashdown M, Aumont J, Baccigalupi C, et al. Planck 2015 results. XI. CMB power spectra, likelihoods, and robustness of parameters. *Astron. Astrophys.* **594** (2016) A11. doi:10.1051/0004-6361/201526926.

[254] Planck Collaboration, Aghanim N, Akrami Y, Ashdown M, Aumont J, Baccigalupi C, et al. Planck 2018 results. VI. Cosmological parameters. *Astron. Astrophys.* **641** (2020) A6. doi:10.1051/0004-6361/201833910.

[255] Eisenstein DJ, Zehavi I, Hogg DW, Scoccimarro R, Blanton MR, Nichol RC, et al. Detection of the Baryon Acoustic Peak in the Large-Scale Correlation Function of SDSS Luminous Red Galaxies. *Astrophys. J.* **633** (2005) 560–574. doi:10.1086/466512.

[256] Percival WJ, Reid BA, Eisenstein DJ, Bahcall NA, Budavari T, Frieman JA, et al. Baryon acoustic oscillations in the Sloan Digital Sky Survey Data Release 7 galaxy sample. *Mon. Not. R. Astron. Soc.* **401** (2010) 2148–2168. doi:10.1111/j.1365-2966.2009.15812.x.

[257] Delubac T, Bautista JE, Busca NG, Rich J, Kirkby D, Bailey S, et al. Baryon acoustic oscillations in the Lyα forest of BOSS DR11 quasars. *Astron. Astrophys.* **574** (2015) A59. doi:10.1051/0004-6361/201423969.

[258] Abbott TMC, Abdalla FB, Alarcon A, Allam S, Andrade-Oliveira F, Annis J, et al. Dark Energy Survey Year 1 Results: Measurement of the Baryon Acoustic Oscillation scale in the distribution of galaxies to redshift 1. *Mon. Not. R. Astron. Soc.* **483** (2019) 4866–4883. doi:10.1093/mnras/sty3351.

[259] Bacon DJ, Refregier AR, Ellis RS. Detection of weak gravitational lensing by large-scale structure. *Mon. Not. R. Astron. Soc.* **318** (2000) 625–640. doi:10.1046/j.1365-8711.2000.03851.x.

[260] Bacon DJ, Massey RJ, Refregier AR, Ellis RS. Joint cosmic shear measurements with the Keck and William Herschel Telescopes. *Mon. Not. R. Astron. Soc.* **344** (2003) 673–685. doi:10.1046/j.1365-8711.2003.06877.x.

[261] Collett TE, Auger MW. Cosmological constraints from the double source plane lens SDSSJ0946+1006. *Mon. Not. R. Astron. Soc.* **443** (2014) 969–976. doi:10.1093/mnras/stu1190.

[262] Suyu SH, Bonvin V, Courbin F, Fassnacht CD, Rusu CE, Sluse D, et al. H0LiCOW - I. H_0 Lenses in COSMOGRAIL's Wellspring: program overview. *Mon. Not. R. Astron. Soc.* **468** (2017) 2590–2604. doi:10.1093/mnras/stx483.

[263] Heymans C, Tröster T, Asgari M, Blake C, Hildebrandt H, Joachimi B, et al. KiDS-1000 Cosmology: Multi-probe weak gravitational lensing and spectroscopic galaxy clustering constraints. *arXiv e-prints* (2020) arXiv:2007.15632.

[264] McQuinn M, Lidz A, Zahn O, Dutta S, Hernquist L, Zaldarriaga M. The morphology of HII regions during reionization. *Mon. Not. R. Astron. Soc.* **377** (2007) 1043–1063. doi:10.1111/j.1365-2966.2007.11489.x.

[265] Pritchard JR, Loeb A. 21 cm cosmology in the 21st century. *Reports on Progress in Physics* **75** (2012) 086901. doi:10.1088/0034-4885/75/8/086901.

[266] Maartens R, Abdalla FB, Jarvis M, Santos MG. Cosmology with the SKA – overview. *arXiv e-prints* (2015) arXiv:1501.04076.

[267] Beardsley AP, Hazelton BJ, Sullivan IS, Carroll P, Barry N, Rahimi M, et al. First Season MWA EoR Power spectrum Results at Redshift 7. *Astrophys. J.* **833** (2016) 102. doi:10.3847/1538-4357/833/1/102.

[268] Perraudin N, Defferrard M, Kacprzak T, Sgier R. Deepsphere: Efficient spherical convolutional neural network with healpix sampling for cosmological applications. *Astronomy and Computing* **27** (2019) 130 – 146. doi:https://doi.org/10.1016/j.ascom.2019.03.004.

[269] Petroff MA, Addison GE, Bennett CL, Weiland JL. Full-sky cosmic microwave background foreground cleaning using machine learning. *The Astrophysical Journal* **903** (2020) 104. doi:10.3847/1538-4357/abb9a7.

[270] Narayan G, Zaidi T, Soraisam MD, Wang Z, Lochner M, Matheson T, et al. Machine-learning-based Brokers for Real-time Classification of the LSST Alert Stream. *The Astrophysical Journal Supplement Series* **236** (2018) 9. doi:10.3847/1538-4365/aab781.

[271] Mahabal A, Rebbapragada U, Walters R, Masci FJ, Blagorodnova N, van Roestel J, et al. Machine Learning for the Zwicky Transient Facility. *Publ. Astron. Soc. Pac.* **131** (2019) 038002. doi:10.1088/1538-3873/aaf3fa.

[272] Förster F, Cabrera-Vives G, Castillo-Navarrete E, Estévez PA, Sánchez-Sáez P, Arredondo J, et al. The Automatic Learning for the Rapid Classification of Events (ALeRCE) Alert Broker. *arXiv e-prints* (2020) arXiv:2008.03303.

[273] Möller A, Peloton J, Ishida EEO, Arnault C, Bachelet E, Blaineau T, et al. fink, a new generation of broker for the LSST community. *Monthly Notices of the Royal Astronomical Society* **501** (2020) 3272–3288. doi:10.1093/mnras/staa3602.

[274] Kamdar HM, Turk MJ, Brunner RJ. Machine learning and cosmological simulations - I. Semi-analytical models. *Mon. Not. R. Astron. Soc.* **455** (2016) 642–658. doi:10.1093/mnras/stv2310.

[275] Rodríguez AC, Kacprzak T, Lucchi A, Amara A, Sgier R, Fluri J, et al. Fast cosmic web simulations with generative adversarial networks. *Computational Astrophysics and Cosmology* **5** (2018) 4. doi:10.1186/s40668-018-0026-4.

[276] Villaescusa-Navarro F, Anglés-Alcázar D, Genel S, Spergel DN, Somerville RS, Dave R, et al. The CAMELS project: Cosmology and Astrophysics with MachinE Learning Simulations. *arXiv e-prints* (2020) arXiv:2010.00619.

[277] J Kates-Harbeck WT A Svyatkovskiy. Predicting disruptive instabilities in controlled fusion plasmas through deep learning. *NATURE* (2019). doi:10.1038/s41586-019-1116-4).

[278] Lin Z, Hahm TS, Lee WW, Tang WM, White RB. Turbulent transport reduction by zonal flows: Massively parallel simulations. *Science* **281** (1998) 1835–1837. doi:10.1126/science.281.5384.1835.

[279] Calabrese FD, Wang L, Ghadimi E, Peters G, Hanzo L, Soldati P. Learning Radio Resource Management in RANs: Framework, Opportunities, and Challenges. *IEEE Commun. Mag.* **56** (2018). doi:10.1109/MCOM.2018.1701031.

[280] Challita U, Dong L, Saad W. Proactive resource management for LTE in unlicensed spectrum: A deep learning perspective. *IEEE Transactions on Wireless Communications* **17** (2018) 4674–4689. doi:10.1109/TWC.2018.2829773.

[281] Huang H, Guo S, Gui G, Yang Z, Zhang J, Sari H, et al. Deep learning for physical-layer 5G wireless techniques: Opportunities, challenges and solutions. *IEEE Wireless Communications* **27** (2020) 214–222. doi:10.1109/MWC.2019.1900027.

[282] Zhu G, Liu D, Du Y, You C, Zhang J, Huang K. Toward an intelligent edge: Wireless communication meets machine learning. *IEEE Communications Magazine* **58** (2020) 19–25. doi:10.1109/MCOM.001.1900103.

[283] Sun H, Chen X, Shi Q, Hong M, Fu X, Sidiropoulos ND. Learning to optimize: Training deep neural networks for interference management. *IEEE Trans. Signal Process.* **66** (2018). doi:10.1109/TSP.2018.2866382.

[284] Liang F, Shen C, Yu W, Wu F. Towards optimal power control via ensembling deep neural networks. *IEEE Transactions on Communications* **68** (2020) 1760–1776. doi:10.1109/TCOMM.2019.2957482.

[285] Nasir YS, Guo D. Deep actor-critic learning for distributed power control in wireless mobile networks. *Proc. Asilomar Conf. Signals Systems Computers* (Pacific Grove, CA) (2020).

[286] Zhao N, Liang YC, Niyato D, Pei Y, Wu M, Jiang Y. Deep reinforcement learning for user association and resource allocation in heterogeneous cellular networks. *IEEE Transactions on Wireless Communications* **18** (2019) 5141–5152. doi:10.1109/TWC.2019.2933417.

[287] Liang L, Ye H, Li GY. Spectrum sharing in vehicular networks based on multi-agent reinforcement learning. *IEEE Journal on Selected Areas in Communications* **37** (2019) 2282–2292. doi:10.1109/JSAC.2019.2933962.

[288] Meng F, Chen P, Wu L, Cheng J. Power allocation in multi-user cellular networks: Deep reinforcement learning approaches. *IEEE Transactions on Wireless Communications* **19** (2020) 6255–6267. doi:10.1109/TWC.2020.3001736.

[289] Chen J, Ran X. Deep learning with edge computing: A review. *Proceedings of the IEEE* **107** (2019) 1655–1674. doi:10.1109/JPROC.2019.2921977.

[290] Zhang K, Zhu Y, Leng S, He Y, Maharjan S, Zhang Y. Deep learning empowered task offloading for mobile edge computing in urban informatics. *IEEE Internet of Things Journal* **6** (2019) 7635–7647. doi:10.1109/JIOT.2019.2903191.

[291] Wang X, Han Y, Leung VCM, Niyato D, Yan X, Chen X. Convergence of edge computing and deep learning: A comprehensive survey. *IEEE Communications Surveys Tutorials* **22** (2020) 869–904. doi:10.1109/COMST.2020.2970550.

[292] Niknam S, Dhillon HS, Reed JH. Federated learning for wireless communications: Motivation, opportunities, and challenges. *IEEE Communications Magazine* **58** (2020) 46–51. doi:10.1109/MCOM.001.1900461.

[293] Amiri MM, Gündüz D. Federated learning over wireless fading channels. *IEEE Transactions on Wireless Communications* **19** (2020) 3546–3557. doi:10.1109/TWC.2020.2974748.

[294] Chen M, Poor HV, Saad W, Cui S. Convergence time optimization for federated learning over wireless networks. *IEEE Transactions on Wireless Communications* **20** (2021) 2457–2471. doi:10.1109/TWC.2020.3042530.

[295] Ren J, He Y, Wen D, Yu G, Huang K, Guo D. Scheduling for Cellular Federated Edge Learning with Importance and Channel Awareness. *IEEE Trans. Wirel. Commun.* **19** (2020). doi:10.1109/TWC.2020.3015671.

[296] Hart T, Tong AHY, Chan K, Van Leeuwen J, Seetharaman A, Aregger M, et al. Evaluation and Design of Genome-Wide CRISPR/SpCas9 Knockout Screens. *G3 Genes—Genomes—Genetics* **7** (2017) 271. doi:10.1534/g3.117.041277.

[297] Möller A, de Boissière T. Supernnova: an open-source framework for bayesian, neural network-based supernova classification. *Monthly Notices of the Royal Astronomical Society* **491** (2019) 4277–4293. doi:10.1093/mnras/stz3312.

[298] Herwig C, Carloni L, Guglielmo GD, Fahim F, Hawks B, Hirschauer JF, et al. Design of a reconfigurable autoencoder algorithm for detector front-end ASICs. *IEEE Nuclear Science Symposium & Medical Imaging Conference* (2020).

[299] Agar JC, Naul B, Pandya S, van der Walt S, Maher J, Ren Y, et al. Revealing ferroelectric switching character using deep recurrent neural networks. *Nature Communications* **10** (2019) 4809. doi:10.1038/s41467-019-12750-0.

[300] Schmidt J, Marques MRG, Botti S, Marques MAL. Recent advances and applications of machine learning in solid-state materials science. *npj Computational Materials* **5** (2019) 83. doi:10.1038/s41524-019-0221-0.

[301] Goodfellow I, Bengio Y, Courville A. *Deep Learning* (The MIT Press) (2016).

[302] Hinton GE, Salakhutdinov RR. Reducing the dimensionality of data with neural networks. *Science* **313** (2006) 504–507. doi:10.1126/science.1127647.

[303] Bengio Y, Courville A, Vincent P. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **35** (2013) 1798–1828. doi:10.1109/TPAMI.2013.50.

[304] Dominguez Sanchez H, Huertas-Company M, Bernardi M, Tuccillo D, Fischer JL. Improving galaxy morphologies for SDSS with Deep Learning. *Monthly Notices of the Royal Astronomical Society* **476** (2018) 3661–3676. doi:10.1093/mnras/sty338.

[305] Huertas-Company M, Primack JR, Dekel A, Koo DC, Lapiner S, Ceverino D, et al. Deep learning identifies high-z galaxies in a central blue nugget phase in a characteristic mass range. *The Astrophysical Journal* **858** (2018) 114. doi:10.3847/1538-4357/aabfed.

[306] Belayneh D, Carminati F, Farbin A, et al. Calorimetry with deep learning: particle simulation and reconstruction for collider. *The European Physical Journal C* **80** (2020). doi:10.1140/epjc/s10052-020-8251-9.

[307] Bychkov D, Linder N, Turkki R, et al. Deep learning based tissue analysis predicts outcome in colorectal cancer. *Nature* **8** (2018). doi:10.1038/s41598-018-21758-3.

[308] Xie X, Niu J, Liu X, Chen Z, Tang S, Yu S. A survey on incorporating domain knowledge into deep learning for medical image analysis. *Medical Image Analysis* (2021) 101985. doi:https://doi.org/10.1016/j.media.2021.101985.

[309] Chakraborty S, Tomsett R, Raghavendra R, Harborne D, Alzantot M, Cerutti F, et al. Interpretability of deep learning models: A survey of results. *2017 IEEE SmartWorld, Ubiquitous Intelligence Computing, Advanced Trusted Computed, Scalable Computing Communications, Cloud Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)* (2017), 1–6. doi:10.1109/UIC-ATC.2017.8397411.

[310] Vo K, Pham D, Nguyen M, Mai T, Quan T. Combination of domain knowledge and deep learning for sentiment analysis. Phon-Amnuaisuk S, Ang SP, Lee SY, editors, *Multi-disciplinary Trends in Artificial Intelligence* (Springer International Publishing) (2017), 162–173.

[311] Particle tracking at cern with machine learning (2021). https://towardsdatascience.com/particle-tracking-at-cern-with-machine-learning-4cb6b255613c.

[312] Ophus C. Four-dimensional scanning transmission electron microscopy (4d-stem): From scanning nanodiffraction to ptychography and beyond. *Microscopy and Microanalysis* **25** (2019) 563–582. doi:10.1017/S1431927619000497.

[313] St John J, et al. Real-time Artificial Intelligence for Accelerator Control: A Study at the Fermilab Booster (2020).

[314] Krupa J, Lin K, Flechas MA, Dinsmore J, Duarte J, Harris P, et al. Gpu coprocessors as a service for deep learning inference in high energy physics (2020).

[315] Kuznetsov V, Giommi L, Bonacorsi D. Mlaas4hep: Machine learning as a service for hep (2020).

[316] Iandola FN, Han S, Moskewicz MW, Ashraf K, Dally WJ, Keutzer K. SqueezeNet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size. *arXiv preprint arXiv:1602.07360* (2016).

[317] Ioannou Y, Robertson D, Cipolla R, Criminisi A. Deep roots: Improving cnn efficiency with hierarchical filter groups. *Proceedings of the IEEE conference on computer vision and pattern recognition* (2017), 1231–1240.

[318] Howard AG, Zhu M, Chen B, Kalenichenko D, Wang W, Weyand T, et al. MobileNets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861* (2017).

[319] Mamalet F, Garcia C. Simplifying convnets for fast learning. *International Conference on Artificial Neural Networks* (Springer) (2012), 58–65.

[320] Ma N, Zhang X, Zheng HT, Sun J. Shufflenet V2: Practical guidelines for efficient cnn architecture design. *Proceedings of the European Conference on Computer Vision (ECCV)* (2018), 116–131.

[321] Wu B, Wan A, Yue X, Jin P, Zhao S, Golmant N, et al. Shift: A zero flop, zero parameter alternative to spatial convolutions. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2018), 9127–9135.

[322] He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. *Proceedings of the IEEE conference on computer vision and pattern recognition* (2016), 770–778.

[323] Huang G, Liu Z, Van Der Maaten L, Weinberger KQ. Densely connected convolutional networks. *Proceedings of the IEEE conference on computer vision and pattern recognition* (2017), 4700–4708.

[324] Hu J, Shen L, Sun G. Squeeze-and-excitation networks. *Proceedings of the IEEE conference on computer vision and pattern recognition* (2018), 7132–7141.

[325] Sandler M, Howard A, Zhu M, Zhmoginov A, Chen LC. MobilenetV2: Inverted residuals and linear bottlenecks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2018), 4510–4520.

[326] Zoph B, Le QV. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578* (2016).

[327] Pham H, Guan M, Zoph B, Le Q, Dean J. Efficient neural architecture search via parameters sharing. *International Conference on Machine Learning* (PMLR) (2018), 4095–4104.

[328] Tan M, Chen B, Pang R, Vasudevan V, Sandler M, Howard A, et al. Mnasnet: Platform-aware neural architecture search for mobile. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2019), 2820–2828.

[329] Liu H, Simonyan K, Yang Y. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055* (2018).

[330] Wu B, Dai X, Zhang P, Wang Y, Sun F, Wu Y, et al. FBNet: Hardware-aware efficient convnet design via differentiable neural architecture search. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2019), 10734–10742.

[331] Gholami A, Kwon K, Wu B, Tai Z, Yue X, Jin P, et al. SqueezeNext: Hardware-aware neural network design. *Workshop paper in CVPR* (2018).

[332] Cai H, Zhu L, Han S. Proxylessnas: Direct neural architecture search on target task and hardware. *arXiv preprint arXiv:1812.00332* (2018).

[333] Cai H, Gan C, Wang T, Zhang Z, Han S. Once-for-all: Train one network and specialize it for efficient deployment. *arXiv preprint arXiv:1908.09791* (2019).

[334] Howard A, Sandler M, Chu G, Chen LC, Chen B, Tan M, et al. Searching for MobilenetV3. *Proceedings of the IEEE International Conference on Computer Vision* (2019), 1314–1324.

[335] Asanovic K, Morgan N. *Experimental determination of precision requirements for back-propagation training of artificial neural networks* (International Computer Science Institute) (1991).

[336] Hubara I, Courbariaux M, Soudry D, El-Yaniv R, Bengio Y. Binarized neural networks. *Advances in neural information processing systems* (2016), 4107–4115.

[337] Rastegari M, Ordonez V, Redmon J, Farhadi A. XNOR-Net: Imagenet classification using binary convolutional neural networks. *European Conference on Computer Vision* (Springer) (2016), 525–542.

[338] Zhou A, Yao A, Guo Y, Xu L, Chen Y. Incremental network quantization: Towards lossless cnns with low-precision weights. *arXiv preprint arXiv:1702.03044* (2017).

[339] Zhou S, Wu Y, Ni Z, Zhou X, Wen H, Zou Y. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *arXiv preprint arXiv:1606.06160* (2016).

[340] Cai Z, He X, Sun J, Vasconcelos N. Deep learning with low precision by half-wave gaussian quantization. *Proceedings of the IEEE conference on computer vision and pattern recognition* (2017), 5918–5926.

[341] Jacob B, Kligys S, Chen B, Zhu M, Tang M, Howard A, et al. Quantization and training of neural networks for efficient integer-arithmetic-only inference. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2018), 2704–2713.

[342] Zhang D, Yang J, Ye D, Hua G. Lq-nets: Learned quantization for highly accurate and compact deep neural networks. *European conference on computer vision (ECCV)* (2018).

[343] Choi J, Wang Z, Venkataramani S, Chuang PIJ, Srinivasan V, Gopalakrishnan K. Pact: Parameterized clipping activation for quantized neural networks. *arXiv preprint arXiv:1805.06085* (2018).

[344] Wang K, Liu Z, Lin Y, Lin J, Han S. Haq: Hardware-aware automated quantization with mixed precision. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2019), 8612–8620.

[345] Dong Z, Yao Z, Gholami A, Mahoney MW, Keutzer K. Hawq: Hessian aware quantization of neural networks with mixed-precision. *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2019), 293–302.

[346] Chin TW, Chuang PIJ, Chandra V, Marculescu D. One weight bitwidth to rule them all. *arXiv preprint arXiv:2008.09916* (2020).

[347] Cai Y, Yao Z, Dong Z, Gholami A, Mahoney MW, Keutzer K. Zeroq: A novel zero shot quantization framework. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2020), 13169–13178.

[348] Gholami A, Kim S, Dong Z, Yao Z, Mahoney MW, Keutzer K. A survey of quantization methods for efficient neural network inference. *arXiv preprint arXiv:2103.13630* (2021).

[349] Han S, Mao H, Dally WJ. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *International Conference on Learning Representations* (2016).

[350] Banner R, Nahshan Y, Hoffer E, Soudry D. Post-training 4-bit quantization of convolution networks for rapid-deployment. *arXiv preprint arXiv:1810.05723* (2018).

[351] Meller E, Finkelstein A, Almog U, Grobman M. Same, same but different: Recovering neural network quantization error through weight factorization. *International Conference on Machine Learning* (PMLR) (2019), 4486–4495.

[352] Choukroun Y, Kravchik E, Yang F, Kisilev P. Low-bit quantization of neural networks for efficient inference. *ICCV Workshops* (2019), 3009–3018.

[353] Zhao R, Hu Y, Dotzel J, De Sa C, Zhang Z. Improving neural network quantization without retraining using outlier channel splitting. *Proceedings of Machine Learning Research* (2019).

[354] Fang J, Shafiee A, Abdel-Aziz H, Thorsley D, Georgiadis G, Hassoun JH. Post-training piecewise linear quantization for deep neural networks. *European Conference on Computer Vision* (Springer) (2020), 69–86.

[355] Fang J, Shafiee A, Abdel-Aziz H, Thorsley D, Georgiadis G, Hassoun J. Near-lossless post-training quantization of deep neural networks via a piecewise linear approximation. *arXiv preprint arXiv:2002.00104* (2020).

[356] Lee JH, Ha S, Choi S, Lee WJ, Lee S. Quantization for rapid deployment of deep neural networks. *arXiv preprint arXiv:1810.05488* (2018).

[357] Nagel M, Baalen Mv, Blankevoort T, Welling M. Data-free quantization through weight equalization and bias correction. *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2019), 1325–1334.

[358] Hawks B, Duarte J, Fraser NJ, Pappalardo A, Tran N, Umuroglu Y. Ps and qs: Quantization-aware pruning for efficient low latency neural network inference (2021).

[359] Haroush M, Hubara I, Hoffer E, Soudry D. The knowledge within: Methods for data-free model compression. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2020), 8494–8502.

[360] Courbariaux M, Bengio Y, David JP. BinaryConnect: Training deep neural networks with binary weights during propagations. *Advances in neural information processing systems* (2015), 3123–3131.

[361] Lin Z, Courbariaux M, Memisevic R, Bengio Y. Neural networks with few multiplications. *arXiv preprint arXiv:1510.03009* (2015).

[362] Zhu C, Han S, Mao H, Dally WJ. Trained ternary quantization. *arXiv preprint arXiv:1612.01064* (2016).

[363] Hou L, Yao Q, Kwok JT. Loss-aware binarization of deep networks. *arXiv preprint arXiv:1611.01600* (2016).

[364] Gysel P, Pimentel J, Motamedi M, Ghiasi S. Ristretto: A framework for empirical study of resource-efficient inference in convolutional neural networks. *IEEE transactions on neural networks and learning systems* **29** (2018) 5784–5789.

[365] Huang Q, Wang D, Dong Z, Gao Y, Cai Y, Li T, et al. Codenet: Efficient deployment of input-adaptive object detection on embedded fpgas. *The 2021 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays* (2021), 206–216.

[366] Zhou A, Yao A, Wang K, Chen Y. Explicit loss-error-aware quantization for low-bit deep neural networks. *Proceedings of the IEEE conference on computer vision and pattern recognition* (2018), 9426–9435.

[367] Yao Z, Dong Z, Zheng Z, Gholami A, Yu J, Tan E, et al. HAWQV3: Dyadic neural network quantization. *arXiv preprint arXiv:2011.10680* (2020).

[368] Kim S, Gholami A, Yao Z, Mahoney MW, Keutzer K. I-bert: Integer-only bert quantization. *arXiv preprint arXiv:2101.01321* (2021).

[369] Lin D, Talathi S, Annapureddy S. Fixed point quantization of deep convolutional networks. *International conference on machine learning* (PMLR) (2016), 2849–2858.

[370] Zhou Y, Moosavi-Dezfooli SM, Cheung NM, Frossard P. Adaptive quantization for deep neural network. *Proceedings of the AAAI Conference on Artificial Intelligence* (2018), vol. 32.

[371] Wang K, Liu Z, Lin Y, Lin J, Han S. HAQ: Hardware-aware automated quantization. *In Proceedings of the IEEE conference on computer vision and pattern recognition* (2019).

[372] Wu B, Wang Y, Zhang P, Tian Y, Vajda P, Keutzer K. Mixed precision quantization of convnets via differentiable neural architecture search. *arXiv preprint arXiv:1812.00090* (2018).

[373] Dong Z, Yao Z, Gholami A, Mahoney M, Keutzer K. HAWQ: Hessian aware quantization of neural networks with mixed-precision. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)* (2019), 293.

[374] Shen S, Dong Z, Ye J, Ma L, Yao Z, Gholami A, et al. Q-bert: Hessian based ultra low precision quantization of bert. *Proceedings of the AAAI Conference on Artificial Intelligence* (2020), vol. 34, 8815–8821.

[375] Dong Z, Yao Z, Cai Y, Arfeen D, Gholami A, Mahoney MW, et al. HAWQ-V2: Hessian aware trace-weighted quantization of neural networks. *Advances in Neural Information Processing Systems 33* (2020).

[376] Dong Z, Gao Y, Huang Q, Wawrzynek J, So HK, Keutzer K. Hao: Hardware-aware neural architecture optimization for efficient inference. *2021 IEEE 29th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)* (IEEE) (2021), 50–59.

[377] LeCun Y, Denker JS, Solla SA. Optimal brain damage. *Advances in neural information processing systems* (1990), 598–605.

[378] Hassibi B, Stork DG. *Second order derivatives for network pruning: Optimal brain surgeon* (Morgan Kaufmann) (1993).

[379] Dong X, Chen S, Pan SJ. Learning to prune deep neural networks via layer-wise optimal brain surgeon. *arXiv preprint arXiv:1705.07565* (2017).

[380] Lee N, Ajanthan T, Torr PH. Snip: Single-shot network pruning based on connection sensitivity. *arXiv preprint arXiv:1810.02340* (2018).

[381] Xiao X, Wang Z, Rajasekaran S. Autoprune: Automatic network pruning by regularizing auxiliary parameters. *Advances in Neural Information Processing Systems* (2019), 13681–13691.

[382] Park S, Lee J, Mo S, Shin J. Lookahead: a far-sighted alternative of magnitude-based pruning. *arXiv preprint arXiv:2002.04809* (2020).

[383] Luo JH, Wu J, Lin W. Thinet: A filter level pruning method for deep neural network compression. *Proceedings of the IEEE international conference on computer vision* (2017), 5058–5066.

[384] He Y, Lin J, Liu Z, Wang H, Li LJ, Han S. Amc: Automl for model compression and acceleration on mobile devices. *Proceedings of the European Conference on Computer Vision (ECCV)* (2018), 784–800.

[385] Yu R, Li A, Chen CF, Lai JH, Morariu VI, Han X, et al. Nisp: Pruning networks using neuron importance score propagation. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2018), 9194–9203.

[386] Lin S, Ji R, Li Y, Wu Y, Huang F, Zhang B. Accelerating convolutional networks via global & dynamic filter pruning. *IJCAI* (2018), 2425–2432.

[387] Huang Z, Wang N. Data-driven sparse structure selection for deep neural networks. *Proceedings of the European conference on computer vision (ECCV)* (2018), 304–320.

[388] Zhao C, Ni B, Zhang J, Zhao Q, Zhang W, Tian Q. Variational convolutional neural network pruning. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2019), 2780–2789.

[389] Buluc A, Gilbert JR. Challenges and advances in parallel sparse matrix-matrix multiplication. *2008 37th International Conference on Parallel Processing* (IEEE) (2008), 503–510.

[390] Gale T, Elsen E, Hooker S. The state of sparsity in deep neural networks. *arXiv preprint arXiv:1902.09574* (2019).

[391] Hoefler T, Alistarh D, Ben-Nun T, Dryden N, Peste A. Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks. *arXiv preprint arXiv:2102.00554* (2021).

[392] Blalock D, Ortiz JJG, Frankle J, Guttag J. What is the state of neural network pruning? *arXiv preprint arXiv:2003.03033* (2020).

[393] Chauvin Y. A back propagation network with optimal use of hidden units. *Advances in neural information processing* (1989).

[394] Hanson S, Pratt L. Comparing biases for minimal network construction with back-propagation. *Advances in neural information processing systems* **1** (1988) 177–185.

[395] Mozer MC, Smolensky P. Skeletonization: A technique for trimming the fat from a network via relevance assessment. *Proceedings of the 1st International Conference on Neural Information Processing Systems* (1988), 107–115.

[396] Li H, Kadav A, Durdanovic I, Samet H, Graf HP. Pruning filters for efficient convnets. *arXiv preprint arXiv:1608.08710* (2016).

[397] He Y, Zhang X, Sun J. Channel pruning for accelerating very deep neural networks. *Proceedings of the IEEE International Conference on Computer Vision* (2017), 1389–1397.

[398] Liu Z, Li J, Shen Z, Huang G, Yan S, Zhang C. Learning efficient convolutional networks through network slimming. *Proceedings of the IEEE International Conference on Computer Vision* (2017), 2736–2744.

[399] He Y, Liu P, Wang Z, Hu Z, Yang Y. Filter pruning via geometric median for deep convolutional neural networks acceleration. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2019), 4340–4349.

[400] Lin M, Ji R, Wang Y, Zhang Y, Zhang B, Tian Y, et al. Hrank: Filter pruning using high-rank feature map. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2020), 1529–1538.

[401] Hassibi B, Stork DG, Wolff GJ. Optimal brain surgeon and general network pruning. *IEEE international conference on neural networks* (IEEE) (1993), 293–299.

[402] Wang C, Grosse R, Fidler S, Zhang G. Eigendamage: Structured pruning in the kronecker-factored eigenbasis. *arXiv preprint arXiv:1905.05934* (2019).

[403] Yu S, Yao Z, Gholami A, Dong Z, Mahoney MW, Keutzer K. Hessian-aware pruning and optimal neural implant. *arXiv preprint arXiv:2101.08940* (2021).

[404] Romero A, Ballas N, Kahou SE, Chassang A, Gatta C, Bengio Y. Fitnets: Hints for thin deep nets. *arXiv preprint arXiv:1412.6550* (2014).

[405] Hinton G, Vinyals O, Dean J. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531* (2015).

[406] Mishra A, Marr D. Apprentice: Using knowledge distillation techniques to improve low-precision network accuracy. *arXiv preprint arXiv:1711.05852* (2017).

[407] Li Y, Yang J, Song Y, Cao L, Luo J, Li LJ. Learning from noisy labels with distillation. *Proceedings of the IEEE International Conference on Computer Vision* (2017), 1910–1918.

[408] Yim J, Joo D, Bae J, Kim J. A gift from knowledge distillation: Fast optimization, network minimization and transfer learning. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2017), 4133–4141.

[409] Polino A, Pascanu R, Alistarh D. Model compression via distillation and quantization. *arXiv preprint arXiv:1802.05668* (2018).

[410] Ahn S, Hu SX, Damianou A, Lawrence ND, Dai Z. Variational information distillation for knowledge transfer. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2019), 9163–9171.

[411] Yin H, Molchanov P, Alvarez JM, Li Z, Mallya A, Hoiem D, et al. Dreaming to distill: Data-free knowledge transfer via deepinversion. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2020), 8715–8724.

[412] Park W, Kim D, Lu Y, Cho M. Relational knowledge distillation. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2019), 3967–3976.

[413] You S, Xu C, Xu C, Tao D. Learning from multiple teacher networks. *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2017), 1285–1294.

[414] Tarvainen A, Valpola H. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. *arXiv preprint arXiv:1703.01780* (2017).

[415] Crowley EJ, Gray G, Storkey AJ. Moonshine: Distilling with cheap convolutions. *NeurIPS* (2018), 2893–2903.

[416] Zhang L, Song J, Gao A, Chen J, Bao C, Ma K. Be your own teacher: Improve the performance of convolutional neural networks via self distillation. *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2019), 3713–3722.

[417] Lopes RG, Fenu S, Starner T. Data-free knowledge distillation for deep neural networks. *arXiv preprint arXiv:1710.07535* (2017).

[418] Nayak GK, Mopuri KR, Shaj V, Radhakrishnan VB, Chakraborty A. Zero-shot knowledge distillation in deep networks. *International Conference on Machine Learning* (PMLR) (2019), 4743–4751.

[419] Wang X, Zhang R, Sun Y, Qi J. Kdgan: Knowledge distillation with generative adversarial networks. *NeurIPS* (2018), 783–794.

[420] Wang Y, Gonzalez-Garcia A, Berga D, Herranz L, Khan FS, Weijer Jvd. Minegan: effective knowledge transfer from gans to target domains with few images. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2020), 9332–9341.

[421] Mao Y, Wang Y, Wu C, Zhang C, Wang Y, Yang Y, et al. Ladabert: Lightweight adaptation of bert through hybrid model compression. *arXiv preprint arXiv:2004.04124* (2020).

[422] Reddi S, Zaheer M, Sachan D, Kale S, Kumar S. Adaptive methods for nonconvex optimization. *Proceeding of 32nd Conference on Neural Information Processing Systems (NIPS 2018)* (2018).

[423] Shazeer N, Stern M. Adafactor: Adaptive learning rates with sublinear memory cost. *International Conference on Machine Learning* (2018), 4596–4604.

[424] Zhang MR, Lucas J, Hinton G, Ba J. Lookahead optimizer: k steps forward, 1 step back (2019).

[425] Zhuang J, Tang T, Ding Y, Tatikonda S, Dvornek N, Papademetris X, et al. Adabelief optimizer: Adapting stepsizes by the belief in observed gradients (2020).

[426] Liu L, Jiang H, He P, Chen W, Liu X, Gao J, et al. On the variance of the adaptive learning rate and beyond (2020).

[427] Ginsburg B, Castonguay P, Hrinchuk O, Kuchaiev O, Lavrukhin V, Leary R, et al. Stochastic gradient methods with layer-wise adaptive moments for training of deep networks (2020).

[428] Yao Z, Gholami A, Shen S, Keutzer K, Mahoney MW. Adahessian: An adaptive second order optimizer for machine learning. *arXiv preprint arXiv:2006.00719* (2020).

[429] Ma X. Apollo: An adaptive parameter-wise diagonal quasi-newton method for nonconvex stochastic optimization. *arXiv preprint arXiv:2009.13586* (2020).

[430] Gupta V, Koren T, Singer Y. Shampoo: Preconditioned stochastic tensor optimization. *arXiv preprint arXiv:1802.09568* (2018).

[431] Yao Z, Gholami A, Keutzer K, Mahoney M. Pyhessian: Neural networks through the lens of the hessian. *arXiv preprint arXiv:1912.07145* (2019).

[432] Raissi M, Perdikaris P, Karniadakis GE. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics* **378** (2019) 686–707.

[433] Trimberger SMS. Three ages of fpgas: A retrospective on the first thirty years of fpga technology: This paper reflects on how moore's law has driven the design of fpgas through three epochs: the age of invention, the age of expansion, and the age of accumulation. *IEEE Solid-State Circuits Magazine* **10** (2018) 16–29.

[434] Esmaeilzadeh H, Blem E, Amant RS, Sankaralingam K, Burger D. Dark silicon and the end of multicore scaling. *2011 38th Annual international symposium on computer architecture (ISCA)* (IEEE) (2011), 365–376.

[435] cerebras. Cerebras (2019).

[436] hotchips. Hotchips'2019 (hc31-k2): Dr. phillip wong (tsmc): What will the next node offer us? (2019).

[437] aspinity. Aspinity (2019).

[438] Yüzügüler AC, Celik F, Drumond M, Falsafi B, Frossard P. Analog neural networks with deep-submicrometer nonlinear synapses. *IEEE Micro* **39** (2019) 55–63.

[439] dwave. D-wave (2019).

[440] neuroblade. Neuroblade (2019).

[441] Essera SK, Merollaa PA, Arthura JV, Cassidya AS, Appuswamya R, Andreopoulosa A, et al. Convolutional networks for fast energy-efficient neuromorphic computing. *Proc. Nat. Acad. Sci. USA* **113** (2016) 11441–11446.

[442] Luiz André Barroso UH. The datacenter as a computer: An introduction to the design of warehouse-scale machines. *Synthesis Lectures on Computer Architecture* **6** (2009).

[443] cascadelake. Intel cascade lake (2019).

[444] Skillman A, Edso T. A technical overview of cortex-m55 and ethos-u55: Arm's most capable processors for endpoint ai. *2020 IEEE Hot Chips 32 Symposium (HCS)* (IEEE Computer Society) (2020), 1–20.

[445] Durant BL, Giroux O, Harris M, Stam N. Inside volta: The world's most advanced data center gpu (2017).
[446] Exxactcorp. Taking a deeper look at the amd radeon instinct gpus for deep learning (2017).
[447] epyc. Amd launches epyc rome, first 7nm cpu (2019).
[448] Hardawar D. Amd's radeon vega gpu is headed everywhere, even to machine learning (2018).
[449] Franklin D. Nvidia jetson tx2 delivers twice the intelligence to the edge (2017).
[450] agx. Nvidia ax (2019).
[451] turing. Nvidia turing gpu architecture (2019).
[452] Sato K, et al. An in-depth look at google's first tensor processing unit (tpu) (2017).
[453] Jouppi NP, Young C, Patil N, Patterson D, Agrawal G, Bajwa R, et al. In-datacenter performance analysis of a tensor processing unit. *ISCA 2017* (ACM) (2017), 1–12.
[454] Teich P. Tearing apart google's tpu 3.0 ai coprocessor (2018).
[455] Armasu L. Deep learning on a stick: Movidius' 'fathom' neural compute stick (2016).
[456] Moons B, Bankman D, Yang L, Murmann B, Verhelst M. Binareye: An always-on energy-accuracy-scalable binary cnn processor with all memory on chip in 28nm cmos. *Custom Integrated Circuits Conference (CICC), 2018 IEEE* (IEEE) (2018), 1–4.
[457] Ando K, Ueyoshi K, Orimo K, Yonekawa H, Sato S, Nakahara H, et al. Brein memory: A 13-layer 4.2 k neuron/0.8 m synapse binary/ternary reconfigurable in-memory deep neural network accelerator in 65 nm cmos. *VLSI Circuits, 2017 Symposium on* (IEEE) (2017), C24–C25.
[458] IBM. Blog: Unlocking the promise of approximate computing for on-chip ai accelerator (2018).
[459] Lattice. Binarized neural network (bnn) accelerator ip (2018).
[460] Chung E, Fowers J, Ovtcharov K, Papamichael M, Caulfield A, Massengill T, et al. Serving dnns in real time at datacenter scale with project brainwave. *IEEE Micro* **38** (2018) 8–20.
[461] Pierini M, Duarte JM, Tran N, Freytsis M. `hls4ml` LHC jet dataset (150 particles) (2020). doi:10.5281/zenodo.3602260.
[462] Umuroglu Y, Fraser NJ, Gambardella G, Blott M, Leong P, Jahre M, et al. Finn: A framework for fast, scalable binarized neural network inference. *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays* (2017), 65–74.
[463] Blott M, Preusser T, Fraser N, Gambardella G, O'Brien K, Umuroglu Y. FINN-R: An end-to-end deep-learning framework for fast exploration of quantized neural networks. *ACM Trans. Reconfigurable Technol. Syst.* **11** (2018). doi:10.1145/3242897.
[464] Fraser NJ, Umuroglu Y, Gambardella G, Blott M, Leong P, Jahre M, et al. Scaling binarized neural networks on reconfigurable logic. *PARMA DITAM2017* (2017), 25–30.
[465] Han S, Kang J, Mao H, Hu Y, Li X, Li Y, et al. Ese: Efficient speech recognition engine with sparse lstm on fpga. *FPGA 2017* (ACM) (2017), 75–84.
[466] Parashar A, Rhu M, et al. Scnn: An accelerator for compressed-sparse convolutional neural networks. *International Symposium on Computer Architecture (ISCA)* (IEEE) (2017), 27–40.
[467] Albericio J, Judd P, et al. Cnvlutin: Ineffectual-neuron-free deep neural network computing. *Computer Architecture News* **44** (2016) 1–13.
[468] Zhang S, Du Z, et al. Cambricon-x: An accelerator for sparse neural networks. *International Symposium on Microarchitecture* (IEEE Press) (2016), 20.
[469] Umuroglu Y, Rasnayake L, Sjalander M. Bismo: A scalable bit-serial matrix multiplication overlay for reconfigurable computing. *arXiv preprint arXiv:1806.08862* (2018).

[470] Judd P, Albericio J, Hetherington T, Aamodt TM, Moshovos A. Stripes: Bit-serial deep neural network computing. *MICRO 2016* (2016), 1–12.

[471] Singh S, Greaves DJ. Kiwi: Synthesis of fpga circuits from parallel programs. *2008 16th International Symposium on Field-Programmable Custom Computing Machines* (2008), 3–12. doi:10.1109/FCCM.2008.46.

[472] Fingeroff M. *High-Level Synthesis Blue Book* (2010).

[473] Nane R, Sima V, Pilato C, Choi J, Fort B, Canis A, et al. A survey and evaluation of fpga high-level synthesis tools. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* **35** (2016) 1591–1604. doi:10.1109/TCAD.2015.2513673.

[474] Ren H. A brief introduction on contemporary high-level synthesis. *2014 IEEE International Conference on IC Design Technology* (2014), 1–4. doi:10.1109/ICICDT.2014.6838614.

[475] Reinders J, Ashbaugh B, Brodman J, Kinsner M, Pennycook J, Tian X. *Data Parallel C++: Mastering DPC++ for Programming of Heterogeneous Systems using C++ and SYCL* (Apress) (2020).

[476] Kapre N, Bayliss S. Survey of domain-specific languages for fpga computing. *2016 26th International Conference on Field Programmable Logic and Applications (FPL)* (2016), 1–12. doi:10.1109/FPL.2016.7577380.

[477] Nordin G, Milder P, Hoe J, Puschel M. Automatic generation of customized discrete fourier transform ips. *Proceedings. 42nd Design Automation Conference, 2005.* (2005), 471–474. doi:10.1145/1065579.1065703.

[478] Janka R, Judd R, Lebak J, Richards M, Campbell D. Vsipl: an object-based open standard api for vector, signal, and image processing. *2001 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No.01CH37221)* (2001), vol. 2, 949–952 vol.2. doi:10.1109/ICASSP.2001.941073.

[479] Mitra A, Najjar W, Bhuyan L. Compiling pcre to fpga for accelerating snort ids (New York, NY, USA: Association for Computing Machinery) (2007), ANCS '07, 127–136. doi:10.1145/1323548.1323571.

[480] Kohler E, Morris R, Chen B, Jannotti J, Kaashoek MF. The click modular router. *ACM Trans. Comput. Syst.* (2000).

[481] Bosshart P, Daly D, Gibb G, Izzard M, McKeown N, Rexford J, et al. P4: Programming protocol-independent packet processors. *SIGCOMM Comput. Commun. Rev.* (2014).

[482] Phothilimthana PM, Liu M, Kaufmann A, Peter S, Bodik R, Anderson T. Floem: A programming system for nic-accelerated network applications. *Proceedings of the 13th USENIX Conference on Operating Systems Design and Implementation* (USENIX Association) (2018).

[483] Mueller R, Teubner J, Alonso G. Glacier: A query-to-hardware compiler (Association for Computing Machinery) (2010), SIGMOD '10.

[484] Sujeeth AK, Lee H, Brown KJ, Chafi H, Wu M, Atreya AR, et al. Optiml: An implicitly parallel domain-specific language for machine learning. *Proceedings of the 28th International Conference on International Conference on Machine Learning* (2011).

[485] Kapre N, DeHon A. Accelerating spice model-evaluation using fpgas. *2009 17th IEEE Symposium on Field Programmable Custom Computing Machines* (2009).

[486] Pell O, Mencer O. Surviving the end of frequency scaling with reconfigurable dataflow computing. *SIGARCH Comput. Archit. News* (2011).

[487] Kapre N, DeHon A. Vliw-score: Beyond c for sequential control of spice fpga acceleration. *2011 International Conference on Field-Programmable Technology* (2011), 1–9. doi:10.1109/FPT.2011.6132678.

[488] Bacon D, Rabbah R, Shukla S. Fpga programming for the masses: The programmability of fpgas must improve if they are to be part of mainstream computing. *Queue* (2013).

[489] Durst D, Feldman M, Huff D, Akeley D, Daly R, Bernstein GL, et al. Type-directed scheduling of streaming accelerators (Association for Computing Machinery) (2020), PLDI 2020.

[490] Bhattacharyya SS, Brebner G, Janneck JW, Eker J, von Platen C, Mattavelli M, et al. Opendf: A dataflow toolset for reconfigurable hardware and multicore systems. *SIGARCH Comput. Archit. News* (2009).

[491] Delorimier M, Kapre N, Mehta N, Dehon A. Spatial hardware implementation for sparse graph algorithms in graphstep. *ACM Trans. Auton. Adapt. Syst.* (2011).

[492] Nurvitadhi E, Weisz G, Wang Y, Hurkat S, Nguyen M, Hoe JC, et al. Graphgen: An fpga framework for vertex-centric graph computation. *2014 IEEE 22nd Annual International Symposium on Field-Programmable Custom Computing Machines* (2014).

[493] Bond B, Hammil K, Litchev L, Singh S. Fpga circuit synthesis of accelerator data-parallel programs. *2010 18th IEEE Annual International Symposium on Field-Programmable Custom Computing Machines* (2010).

[494] Papakonstantinou A, Gururaj K, Stratton JA, Chen D, Cong J, Hwu WMW. Fcuda: Enabling efficient compilation of cuda kernels onto fpgas. *2009 IEEE 7th Symposium on Application Specific Processors* (2009).

[495] Lai YH, Rong H, Zheng S, Zhang W, Cui X, Jia Y, et al. Susy: A programming model for productive construction of high-performance systolic arrays on fpgas. *2020 IEEE/ACM International Conference On Computer Aided Design (ICCAD)* (2020).

[496] de Dinechin F, Klein C, Pasca B. Generating high-performance custom floating-point pipelines. *2009 International Conference on Field Programmable Logic and Applications* (2009).

[497] Bellows P, Hutchings B. Jhdl-an hdl for reconfigurable systems. *Proceedings. IEEE Symposium on FPGAs for Custom Computing Machines (Cat. No.98TB100251)* (1998).

[498] Bertin P, Touati H. Pam programming environments: practice and experience. *Proceedings of IEEE Workshop on FPGA's for Custom Computing Machines* (1994).

[499] Reiche O, Özkan MA, Membarth R, Teich J, Hannig F. Generating fpga-based image processing accelerators with hipacc: (invited paper). *2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)* (2017).

[500] Del Sozzo E, Baghdadi R, Amarasinghe S, Santambrogio MD. A common backend for hardware acceleration on fpga. *2017 IEEE International Conference on Computer Design (ICCD)* (2017).

[501] Hegarty J, Brunhaver J, DeVito Z, Ragan-Kelley J, Cohen N, Bell S, et al. Darkroom: Compiling high-level image processing code into hardware pipelines. *ACM Trans. Graph.* (2014).

[502] Stewart R, Duncan K, Michaelson G, Garcia P, Bhowmik D, Wallace A. Ripl: A parallel image processing language for fpgas (2018).

[503] Chugh N, Vasista V, Purini S, Bondhugula U. A dsl compiler for accelerating image processing pipelines on fpgas. *2016 International Conference on Parallel Architecture and Compilation Techniques (PACT)* (2016).

[504] Guccione S, Levi D, Sundararajan P. Jbits: Java based interface for reconfigurable computing (2000).

[505] Moreau T, Chen T, Ceze L. Leveraging the vta-tvm hardware-software stack for fpga acceleration of 8-bit resnet-18 inference. *Proceedings of the 1st on Reproducible Quality-Efficient Systems Tournament on Co-designing Pareto-efficient Deep Learning. ReQuEST.* (2018). doi:doi:10.1145/3229762.3229766.

[506] Chi Y, Guo L, Lau J, Choi Yk, Wang J, Cong J. Extending high-level synthesis for task-parallel programs. *2021 IEEE 29th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)* (2021).

[507] Clow J, Tzimpragos G, Dangwal D, Guo S, McMahan J, Sherwood T. A pythonic approach for rapid hardware prototyping and instrumentation. *2017 27th International Conference on Field Programmable Logic and Applications (FPL)* (2017), 1–7. doi:10.23919/FPL.2017.8056860.

[508] Segal O, Margala M, Chalamalasetti Sr, Wright M. High level programming for heterogeneous architectures. *1st International Workshop on FPGAs for Software Programmers. FSP. Aug. 21, 2014. arXiv: 1408. 4964 [cs.PF].* (2014).

[509] Fumero J, Papadimitriou M, Zakkak FS, Xekalaki M, Clarkson J, Kotselidis C. Dynamic application reconfiguration on heterogeneous hardware. *In: Proceedings of the 15th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments. VEE. 2019. doi: 10.1145/3313808.3313819* (2019).

[510] Caldeira P, Penha JC, Bragança L, Ferreira R, Nacif JAM, Ferreira R, et al. From java to fpga: An experience with the intel harp system. *2018 30th International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD)* (2018).

[511] Chung ES, Davis JD, Lee J. Linqits: Big data on little clients. *Proceedings of the 40th Annual International Symposium on Computer Architecture* (Association for Computing Machinery) (2013).

[512] Koeplinger D, Prabhakar R, Zhang Y, Delimitrou C, Kozyrakis C, Olukotun K. Automatic generation of efficient accelerators for reconfigurable hardware. *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)* (2016).

[513] Koeplinger D, Feldman M, Prabhakar R, Zhang Y, Hadjis S, Fiszel R, et al. Spatial: A language and compiler for application accelerators. *SIGPLAN Not.* (2018).

[514] Nigam R, Thomas S, Li Z, Sampson A. A compiler infrastructure for accelerator generators. *Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems* (Association for Computing Machinery) (2021), ASPLOS 2021.

[515] Choi J, Vijayaraghavan M, Sherman B, Chlipala A, Arvind. Kami: A platform for high-level parametric hardware specification and its modular verification. *Proc. ACM Program. Lang.* (2017).

[516] Schkufza E, Wei M, Rossbach CJ. Just-in-time compilation for verilog: A new technique for improving the fpga programming experience. *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems* (Association for Computing Machinery) (2019).

[517] Papadimitrioua M, Fumeroa J, Stratikopoulosa A, Zakkakb FS, Kotselidisa C. Transparent compiler and runtime specializations for accelerating managed languages on fpgas. *ACM* (2012).

[518] Todman T, Constantinides G, Wilton S, Mencer O, Luk W, Cheung P. Reconfigurable computing: Architectures and design methods. *Computers and Digital Techniques, IEE Proceedings* - **152** (2005) 193 – 207. doi:10.1049/ip-cdt:20045086.

[519] Choi YK, Cong J, Fang Z, Hao Y, Reinman G, Wei P. In-depth analysis on microarchitectures of modern heterogeneous cpu-fpga platforms. *ACM Trans. Reconfigurable Technol. Syst.* **12** (2019).

[520] Xilinx. Xilinx runtime (xrt) architecture (2021). (Last accessed: 2021-02-16).

[521] Xilinx. Sdsoc development environment (2020). (Last accessed: 2020-02-16).

[522] Xilinx. Sdaccel development environment (2020). (Last accessed: 2020-02-16).

[523] Xilinx. Vitis unified software platform overview (2020). (Last accessed: 2020-02-16).

[524] Xilinx. What's an acap adaptive compute acceleration platform (2020). (Last accessed: 2020-02-16).

[525] Enno Luebbers MC Song Liu. White paper: Simplify software integration for fpga accelerators with opae (2020).
[526] Intel. What is oneapi? (2020). (Last accessed: 2020-02-16).
[527] Eckert M, Meyer D, Haase J, Klauer B. Operating system concepts for reconfigurable computing: Review and survey. *International Journal of Reconfigurable Computing* **2016** (2016) 1–11. doi:10.1155/2016/2478907.
[528] King M, Hicks J, Ankcorn J. Software-driven hardware development (Association for Computing Machinery) (2015), FPGA '15. doi:10.1145/2684746.2689064.
[529] Vipin K, Shreejith S, Gunasekera D, Fahmy SA, Kapre N. System-level fpga device driver with high-level synthesis support. *2013 International Conference on Field-Programmable Technology (FPT)* (2013), 128–135. doi:10.1109/FPT.2013.6718342.
[530] Jacobsen M, Richmond D, Hogains M, Kastner R. Riffa 2.1: A reusable integration framework for fpga accelerators. *ACM Trans. Reconfigurable Technol. Syst.* (2015). doi:10.1145/2815631.
[531] CCIX. Ccix consortium (2020). (Last accessed: 2020-02-16).
[532] HSA. Heterogeneous system architecture (2020). (Last accessed: 2020-02-16).
[533] Alessandro, Franco G, Fraser N. Xilinx/brevitas: bnn_pynq-r1 (2020). doi:10.5281/zenodo.4020996.
[534] Bai J, Lu F, Zhang K, et al. Onnx: Open neural network exchange. *GitHub repository* (2019).
[535] Umuroglu Y, Jahre M. Streamlined deployment for quantized neural networks. *CoRR* **abs/1709.04060** (2017).
[536] Kathail V. Xilinx vitis unified software platform. *Proceedings of the 2020 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays* (2020), 173–174.
[537] Moons B, Uytterhoeven R, Dehaene W, Verhelst M. Envision: A 26-to-10 TOps/w subword-parallel dynamic-voltage-accuracy-frequency-scalable convolutional neural network processor in 28nm FDSOI. *IEEE International Solid-State Circuits Conference (ISSCC'17)* (2017), 246–257.
[538] Chen YH, Krishna T, Emer J, Sze V. Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks. *IEEE Journal of Solid State Circuits* **52** (2017) 127–138.
[539] Simons T, Lee D. A review of binarized neural networks. *Electronics* **8** (2019) 661.
[540] Sandler M, Howard A, Zhu M, Zhmoginov A, Chen LC. MobileNetV2: Inverted residuals and linear bottlenecks. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'18)* (2018), 4510–4520.
[541] Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez A, et al. Attention is all you need. *International Conference on Neural Information Processing Systems (NIPS'17)* (2017), 6000–6010.
[542] Vinyals O, et al. Grandmaster level in StarCraft ii using multi-agent reinforcement learning. *Nature* **575** (2019) 350–354.
[543] Dosovitskiy A, et al. An image is worth 16×16 words: Transformers for image recognition at scale. *ArXive Preprint* (2020) arXive:2010.11929.
[544] Rajbhandari S, Rasley J, Ruwase O, He Y. ZeRO: Memory optimizations toward training trillion parameter models. *International Conference for High Performance Computing, Networking, Storage and Analysis (SC'20)* (2020), 20.
[545] Brown T, et al. Language models are few-shot learners. *Advances in Neural Information Processing Systems* **33** (2020) 877–901.
[546] Hasler J, Marr H. Finding a roadmap to achieve large neuromorphic hardware systems. *Frontiers in Neuroscience* **7** (2013) 118.
[547] Mead C. Neuromorphic electronic systems. *Proceedings of the IEEE* **78** (1990) 1629–1636.

[548] Indiveri G, et al. Neuromorphic silicon neuron circuits. *Frontiers in Neuroscience* **5** (2011) 73.

[549] Berggren K, Xia Q, Likharev KK, Strukov DB, Jiang H, Mikolajick T, et al. Roadmap on emerging hardware and technology for machine learning. *Nanotechnology* **32** (2020) 012002. doi:10.1088/1361-6528/aba70f.

[550] Burr G, et al. Neuromorphic computing using nonvolatile memory. *Advances in Physics: X* **2** (2017) 89–124.

[551] Bavandpour M, et al. Mixed-signal neuromorphic inference accelerators: Recent results and future prospects. *International Electron Device Meeting (IEDM'18)* (2018), 20.4.1–20.4.4.

[552] Yang J, Strukov D, Stewart D. Memristive devices for computing. *Nature Nanotechnology* **8** (2013) 13–24.

[553] Yu S. Neuro-inspired computing with emerging nonvolatile memories. *Proceedings of the IEEE* **106** (2018) 260–285.

[554] Sheridan PM, Cai F, Du C, Ma W, Zhang Z, Lu WD. Sparse coding with memristor networks. *Nature Nanotechnology* **12** (2017) 784–789.

[555] Cai F, et al. A fully integrated reprogrammable memristor–CMOS system for efficient multiply-accumulate operations. *Nature Electronics* **2** (2019) 290–299.

[556] Chu M, Kim B, Park S, Hwang H, Jeon M, Lee BH, et al. Neuromorphic hardware system for visual pattern recognition with memristor array and CMOS neuron. *IEEE Transactions on Industrial Electronics* **62** (2014) 2410–2419.

[557] Yeon H, et al. Alloying conducting channels for reliable neuromorphic computing. *Nature Nanotechnology* **15** (2020) 574–579.

[558] Ohno T, Hasegawa T, Tsuruoka T, Terabe K, Gimzewski JK, Aono M. Short-term plasticity and long-term potentiation mimicked in single inorganic synapses. *Nature Materials* **10** (2011) 591–595.

[559] Wang Z, et al. Memristors with diffusive dynamics as synaptic emulators for neuromorphic computing. *Nature Materials* **16** (2017) 101–108.

[560] Pickett MD, Medeiros-Ribeiro G, Williams RS. A scalable neuristor built with Mott memristors. *Nature Materials* **12** (2013) 114–117.

[561] Wang Z, et al. Fully memristive neural networks for pattern classification with unsupervised learning. *Nature Electronics* **1** (2018) 137–145.

[562] Zhang Y, et al. Highly compact artificial memristive neuron with low energy consumption. *Small* **14** (2018) 1802188.

[563] Lashkare S, Chouhan S, Chavan T, Bhat A, Kumbhare P, Ganguly U. PCMO RRAM for integrate-and-fire neuron in spiking neural networks. *IEEE Electron Device Letters* **39** (2018) 484–487.

[564] Adda C, et al. First demonstration of "Leaky Integrate and Fire" artificial neuron behavior on $(V_{0.95}Cr_{0.05})_2O_3$ thin film. *MRS Communications* **8** (2018) 835–841.

[565] Alibart F, Gao L, Hoskins B, Strukov DB. High precision tuning of state for memrsitive devices by adaptable variation-tolerant algorithm. *Nanotechnology* **23** (2012) 075201.

[566] Adam G, Hoskins B, Prezioso M, Merrikh-Bayat F, Chakrabarti B, Strukov D. 3-D memristor crossbars for analog and neuromorphic computing applications. *IEEE Transactions on Electron Devices* **64** (2017) 312–318.

[567] Govoreanu B, et al. Vacancy-modulated conductive oxide resistive RAM (VMCO-RRAM). *IEEE International Electron Device Meeting (IEDM'13)* (2013), 10.2.1–10.2.4.

[568] Prezioso M, et al. Modelling and implementation of firing-rate neuromorphic-network classifiers with bilayer Pt/Al_2O_3/TiO_{2-x}/Pt memristors. *IEEE International Electron Device Meeting (IEDM'15)*

(2015), 17.4.1–17.4.4.

[569] Prezioso M, Bayat FM, Hoskins BD, Likharev KK, Strukov D. Self-adaptive spike-time-dependent plasticity of metal-oxide memristors. *Nature Scientific Reports* **6** (2016) 21331.

[570] Prezioso M, Mahmoodi M, Bayat FM, Kim H, Nili H, Vincent A, et al. Spike-timing-dependent plasticity learning of coincidence detection with passively integrated memristive circuits. *Nature Communications* **9** (2018) 5311.

[571] Bayat FM, et al. Implementation of multilayer perceptron network with highly uniform passive memristive crossbar circuits. *Nature Communications* **9** (2018) 2331.

[572] Lin P, et al. Three-dimensional memristor circuits as complex neural networks. *Nature Electronics* **3** (2020) 225–232.

[573] Hu M, et al. Memristor-based analog computation and neural network classification with a dot product engine. *Advanced Materials* **30** (2018) 1705914.

[574] Yao P, et al. Fully hardware-implemented memristor convolutional neural network. *Nature* **577** (2020) 641–646.

[575] Liu Q, et al. A fully integrated analog ReRAM based 78.4 TOps/W compute-in-memory chip with fully parallel MAC computing. *IEEE International Solid-State Circuits Conference (ISSCC'20)* (2020), 500–501.

[576] Kim H, Nili H, Mahmoodi M, Strukov D. 4K-memristor analog-grade passive crossbar circuit. *ArXive Preprint* (2019) arXiv:1906.12045.

[577] Cai F, et al. Power-efficient combinatorial optimization using intrinsic noise in memristor hopfield neural networks. *Nature Electronics* **3** (2020) 409–418.

[578] Mahmoodi M, Kim H, Fahimi Z, Nili H, Sedov L, Polishchuk V, et al. An analog neuro-optimizer with adaptable annealing based on 64×64 $0t1r$ crossbar circuit. *IEEE International Electron Device Meeting (IEDM'19)* (2009), 14.7.1–14.7.4.

[579] Mahmoodi M, Prezioso M, Strukov D. Versatile stochastic dot product circuits based on nonvolatile memories for high-performance neurocomputing and neurooptimization. *Nature Communications* **10** (2019) 5113.

[580] Li H, et al. Hyperdimensional computing with 3D VRRAM in-memory kernels: Device-architecture co-design for energy-efficient, error-resilient language recognition. *IEEE International Electron Devices Meeting (IEDM'16)* (2016), 16.1.1–16.1.4.

[581] Pedretti G, et al. Memristive neural network for on-line learning and tracking with brain-inspired spike timing dependent plasticity. *Nature Scientific Reports* **7** (2017) 5288.

[582] Burr G, et al. Experimental demonstration and tolerancing of a large-scale neural network (165000 synapses) using phase-change memory as the synaptic weight element. *IEEE Transactions on Electron Devices* **62** (2015) 3498–3507.

[583] Tuma T, Pantazi A, Gallo M, Sebastian A, Eleftheriou E. Stochastic phase-change neurons. *Nature Nanotechnology* **11** (2016) 693–699.

[584] Ambrogio S, et al. Equivalent-accuracy accelerated neural-network training using analogue memory. *Nature* **558** (2018) 60–67.

[585] Karunaratne G, Gallo M, Cherubini G, Benini L, Rahimi A, Sebastian A. In-memory hyperdimensional computing. *Nature Electronics* **3** (2020) 327–337.

[586] Joshi V, Gallo ML, Haefeli S, Boybat I, Nandakumar SR, Piveteau C, et al. Accurate deep neural network inference using computational phase-change memory. *Nature Communications* **11** (2020) 2473.

[587] Kuzum D, Jeyasingh RG, Lee B, Wong HSP. Nanoelectronic programmable synapses based on phase change materials for brain-inspired computing. *Nano Letters* **12** (2011) 2179–2186.

[588] Ríos C, et al. In-memory computing on a photonic platform. *Science Advances* **5** (2019) eaau5759.

[589] Guo X, Bayat FM, Prezioso M, Chen Y, Nguyen B, Do N, et al. Temperature-insensitive analog vector-by-matrix multiplier based on 55 nm NOR flash memory cells. *IEEE Custom Integrated Circuits Conference (CICC'17)* (2017), 1–4.

[590] Guo X, Bayat FM, Bavandpour M, Klachko M, Mahmoodi M, Prezioso M, et al. Fast, energy-efficient, robust, and reproducible mixed-signal neuromorphic classifier based on embedded NOR flash memory technology. *IEEE International Electron Device Meeting (IEDM'17)* (2017), 6.5.1–6.5.4.

[591] Bayat FM, Guo X, Om'mani H, Do N, Likharev K, Strukov D. Redesigning commercial floating-gate memory for analog computing applications. *International Symposium on Circuits and Systems (ISCAS'15)* (2015), 1921–1924.

[592] Bavandpour M, Sahay S, Mahmoodi MR, Strukov D. 3D-aCortex: An ultra-compact energy-efficient neurocomputing platform based on commercial 3D-NAND flash memories. *ArXiv Preprint* (2019) arXiv:1908.02472.

[593] Bavandpour M, Sahay S, Mahmoodi M, Strukov D. Mixed-signal vector-by-matrix multiplier circuits based on 3D-NAND memories for neurocomputing. *Design Automation and Test in Europe (DATE'20)* (2020), 696–701.

[594] Lee ST, et al. High-density and highly-reliable binary neural networks using NAND flash memory cells as synaptic devices. *IEEE International Electron Devices Meeting (IEDM'19)* (2019), 38.4.1–38.4.4.

[595] Fuller EJ, et al. Parallel programming of an ionic floating-gate memory array for scalable neuromorphic computing. *Science* **364** (2019) 570–574.

[596] Grollier J, Querlioz D, Camsari K, Everschor-Sitte K, Fukami S, Stiles M. Neuromorphic spintronics. *Nature Electronics* **3** (2020) 360–370.

[597] Ostwal V, Debashis P, Faria R, Chen Z, Appenzeller J. Spin-torque devices with hard axis initialization as stochastic binary neurons. *Nature Scientific Reports* **8** (2018) 16689.

[598] Sengupta A, Panda P, Wijesinghe P, Kim Y, Roy K. Magnetic tunnel junction mimics stochastic cortical spiking neurons. *Nature Scientific Reports* **6** (2016) 30039.

[599] Romera M, et al. Vowel recognition with a four coupled spin-torque nano-oscillators. *Nature* **563** (2018) 230–234.

[600] Ni K, et al. SoC logic compatible multi-bit FeMFET weight cell for neuromorphic applications. *IEEE International Electron Devices Meeting (IEDM'18)* (2018), 13.2.1–13.2.4.

[601] Shasti BJ, et al. Photonics for artificial intelligence and neuromorphic computing. *Nature Photonics* **15** (2021) 102–114.

[602] Goi E, Zhang Q, Chen X, Luan H, Gu M. Perspective on photonic memristive neuromorphic computing. *PhotoniX* **1** (2020) 3.

[603] Lin X, et al. All optical machine learning using diffractive deep learning networks. *Science* **361** (2019) 1004–1008.

[604] Hamerly R, Bernstein L, Sludds A, Soljacic M, Englund D. Large-scale optical neural networks based on photoelectric multiplication. *Physical Review X* **9** (2019) 021032.

[605] Hamley R, et al. Towards large-scale photonics neural-network accelerators. *IEEE International Electron Device Meeting (IEDM'19)* (2019), 22.8.1–22.8.4.

[606] Shen Y, et al. Deep learning with coherent nanophotonic circuits. *Nature Photonics* **11** (2017) 441–447.
[607] Tait AN, et al. Microring weight banks. *IEEE Journal of Selected Topics in Quantum Electronics* **22** (2016) 312–325.
[608] Feldmann J, Youngblood N, Wright CD, Bhaskaran H, Pernice WHP. All-optical spiking neurosynaptic networks with self-learning capabilities. *Nature* **569** (2019) 208–214.
[609] Buckley S, Chiles J, Mccaughan AN, Moody G, Silverman KL, Stevens MJ, et al. All-silicon light-emitting diodes waveguide-integrated with superconducting single-photon detectors. *Applied Physics Letters* **111** (2017) 141101.
[610] Bruiner D, Soriano MC, Mirasso CR, Fisher I. Parallel photonic information processing at gigabyte per second data rates using transient states. *Nature Communications* **4** (2013) 1364.
[611] Vandoorne K, et al. Experimental demonstration of reservoir computing on a silicon photonics chip. *Nature Communications* **5** (2014) 3541.
[612] Segall K, Legro M, Kaplan S, Svitelskiy O, Khadka S, Crotty P, et al. Synchronization dynamics on the picosecond time scale in coupled Josephson junction networks. *Physics Review E* **95** (2017) 032220.
[613] Rowlands GE, Nguyen MH, Ribeill GJ, Wagner AP, Govia L, Barbosa W, et al. Reservoir computing with superconducting electronics. *ArXive Preprint* (2021) arXiv:2103.02522.
[614] Yamamoto Y, Leleu T, Ganguli S, Mabuchi H. Coherent Ising machines – Quantum optics and neural network perspectives. *Applied Physics Letters* **117** (2020) 160501.
[615] Markovich D, Grolier J. Quantum neuromorphic computing. *Applied Physics Letters* **117** (2020) 150501.
[616] Holmes AJ, et al. Use of a-Si:H memory devices for nonvolatile weight storage in artificial neural network. *Journal of Non-Crystalline Solids* **164–166** (1993) 817–820.
[617] Widrow B, Angel JB. Reliable, trainable networks for computing and control. *Aerospace Engineering* (1962) 78–123.
[618] Chawla R, Bandyopadhyay A, Srinivasan V, Hasler P. A 531 nW/MHz, 128×32 current-mode programmable analog vector-matrix multiplier with over two decades of linearity. *IEEE Custom Integrated Circuits Conference (CICC'04)* (2004), 651–654.
[619] Hertz J, Krogh A, Palmer RG (Cambridge, MA: Perseus) (1991).
[620] Gerstner W, Kistler W (Cambridge, MA: Cambridge University Press) (2002).
[621] Yang T, Sze V. Design considerations for efficient deep neural networks on processing-in-memory accelerators. *IEEE International Electron Device Meeting (IEDM'19)* (2019), 22.1.1–22.1.4.
[622] Diorio C, Hasler P, Minch A, Mead CA. A single-transistor silicon synapse. *IEEE Transactions on Electron Devices* **43** (1996) 1972–1980.
[623] George S, Kim S, Shah S, Hasler J, Collins M, Adil F, et al. A programmable and configurable mixed-mode FPAA SoC. *IEEE Transactions on Very Large Scale Integration Systems* **24** (2016) 2253–2261.
[624] Rolls ET, Deco G (Oxford University Press) (2010).
[625] Hinton GE, Sejnowski TJ. Optimal perceptual inference. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'83)* (1983), 448–451.
[626] Hinton G. Deep belief networks. *Scholarpedia* **4** (2009) 5947.
[627] Neftci EO, Mostafa H, Zenke F. Surrogate gradient learning in spiking neural networks: Bringing the power of gradient-based optimization to spiking neural networks. *IEEE Signal Processing Magazine*

36 (2019) 51–63.

[628] Tavanaei A, Ghodrati M, Kheradpisheh S, Masquelier T, Maida A. Deep learning in spiking neural networks. *Neural Networks* **111** (2019) 47–63.

[629] Thakur CS, et al. Large-scale neuromorphic spiking array processors: A quest to mimic the brain. *Frontiers in Neuroscience* **12** (2018) 891.

[630] Kesim YE (Carnegie Mellon University: Ph.D. Thesis) (2019).

[631] Serrano-Gotarredona T, et al. STDP and STDP variations with memristors for spiking neuromorphic learning systems. *Frontiers in Neuroscience* **7** (2013) 2.

[632] Saighi S, et al. Plasticity in memristive devices for spiking neural networks. *Frontiers in Neuroscience* **9** (2015) 5.

[633] Likharev KK, Semenov VK. RSFQ logic/memory family: A new Josephson-junction technology for sub-terahertz-clock-frequency digital systems. *IEEE Transactions on Applied Superconductivity* **1** (1991) 3–28.

[634] Likharev K. Superconductor digital electronics. *Physica C: Superconductivity and Its Applications* **482** (2012) 6–18.

[635] Lukoševičius M, Jaeger H. Reservoir computing approaches to recurrent neural network training. *Computer Science Review* **3** (2009) 127–149.

[636] Maass W, Natschläger T, Markram H. Fading memory and kernel properties of generic cortical microcircuit models. *Journal of Physiology-Paris* **98** (2004) 315–330.

[637] Maass W, Natschläger T, Markram H. Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural Computation* **14** (2002) 2531–2560.

[638] Jaeger H (Bonn, Germany: German National Research Center for Information Technology, Technical Report) (2001).

[639] Tanaka G, et al. Recent advances in physical reservoir computing: A review. *Neural Networks* **115** (2019) 100–123.

[640] Kanerva P. Hyperdimensional computing: An introduction to computing in distributed representation with high-dimensional random vectors. *Cognitive Computing* **1** (2009) 139–159.

[641] Bridle JS. Training stochastic model recognition algorithms as networks can lead to maximum mutual information estimation of parameters. *International Conference on Neural Information Processing Systems (NIPS'89)* (1989), 211–217.