Imperial College London

MEng Individual Project

Imperial College London

Department of Computing

Reconfigurable Acceleration of Transformer Neural Networks with Meta-Programming Strategies for Particle Collisions Experiments

Author:
Filip Wojcicki

Supervisor:
Prof. Wayne Luk

Second Marker:
Dr. TODO

January 26, 2022

Acknowledgements

Although the project is at an early stage, I would like to express my gratitude to Professor Wayne Luk and Zhiqiang Que for guiding me through the project and always being available to answer any of my questions.

Contents

Chapter 1

Introduction

1.1 Overview

Particle physics is one of the key branches of modern physics, with the Standard Model theory at its core. It tackles the underlying questions about the nature of the universe by describing the fundamental forces and elementary particles. In order to verify the correctness of the theories, countless experiments have to be designed and carefully executed, with the main driving force of myriads of engineers, physicists and researchers at Large Hadron Collider (LHC) operated by the European Organization for Nuclear Research (CERN).

LHC is the world's highest-energy particle collider that is capable of producing and detecting the heaviest types of particles that emerge from collisions such as a proton-proton collisions. The detection is a challenging process as some particles like quarks and gluons cannot exist on their own, and they nearly instantly combine which results in a collimated spray of composite particles (hadrons) that is typically referred to as a **jet** [1]. The initial particles created upon collision and their behaviors are of main interest of the physicists, which leads to **jet tagging** - the challenge of associating particle jets with their origin.

1.2 Motivation

There are many detector types used for the analysis the particle collisions, each based on a different physical methodology, which result in availability of both higher and lower level features. The former have been successfully used in the past using more physically motivated machine learning (ML) algorithms, e.g. using computer vision [2]. However, more recently, various deep learning approaches have proven to outperform their predecessors [3]. It has also been found that all the detected features carry the same underlying information, with convolutional neural networks (CNN) trained on higher-level data achieving nearly identical accuracy as dense neural networks (DNN) trained on the data from the other end of the spectrum [4].

The Pb/s throughput of information collected by the LHC detectors outclasses the real-time inference capabilities of the typical state-of-the-art solutions. The real-time decision-making is often required, hence this paper is motivated by the successful adoption of ***hls4ml*** codesign workflow in particle physics experiments [5]. It allows ML researchers and physicists to easily deploy their solutions trained using common ML frameworks on reconfigurable or application specific hardware, vastly improving the detection algorithms throughput. However, *hls4ml* lacks support for a number of neural network architectures that have been proven to outperform the previous state-of-the-art, including graph neural networks (GNN) [6, 7] and transformer neural networks [8].

1.3 Objectives and Challenges

The purpose of this project is to develop state-of-the-art neural network architectures for Field-Programmable Gate Arrays (FPGA) technology. While working towards this goal, there is an

emphasis on creating parametrizable and reusable designs as the next objective is to use metaprogramming strategies to integrate them into the *hls4ml* library with various optimizations that offer trade-offs between speed and hardware resources usage.

The two main challenges of the project involve:

- Developing deep and complex neural networks in hardware which requires working at a much lower abstraction level than a typical ML frameworks. It is also crucial to stay aware of the underlying hardware architecture to exploit its strengths while still making it possible for users' to configure it towards their needs.
- Bridging the abstraction gap for the translation between *hls4ml* high-level representation of neural networks and their customizable instantiation in hardware.

1.4 Contributions

The project aims to benefit the open-source community of ML researches that are in need of faster and more parametrizable neural network inference. The targeted audience for that operation are physicists at LHC, nonetheless, the hope is for the work to positively contribute in many ML fields by both offering a reliable tool for acceleration of existing designs and providing a useful resource for learning about the nature of reconfigurable hardware and its potential use for neural networks.

how the report is structured

Chapter 2

Background and related work

This chapter provides a closer look at the concepts required to understand this work. The following sections firstly discuss background and related work for topics in particle physics, then machine learning and finally reconfigurable hardware research.

2.1 Particle Physics

2.1.1 Standard Model

2.1.2 Particle Accelerators

2.1.3 Particle Collisions

2.1.4 Jets and prongs

2.1.5 Trigger levels at LHC

find more info about the structure and timings

2.2 Machine Learning

general short info about ML, tensors

2.2.1 Dataset

training-validation-test dataset and the dataset we use

2.2.2 Training and Inference

2.2.3 Classification accuracy, Area-Under-Curve, and confusion matrix

2.2.4 Neural Networks

2.2.5 Deep Learning

2.2.6 Machine Learning Frameworks and *hls4ml*

2.2.7 Transformer Neural Networks

2.2.8 Attention

2.2.9 ConstituentNet architecture

2.3 Reconfigurable Hardware

A significant portion of the project's work involves exploiting reconfigurable hardware to vastly reduce the inference time of state-of-the-art neural networks. This section explains in more detail the technology and behavior of Field-Programmable Gate Arrays (FPGA).

2.3.1 Landscape of Hardware for Computing

There modern landscape of digital integrated circuits (IC) is very rich can be divided into numerous categories depending on the technology used and expected functionality [9]. A list of platform types is described below, with the emphasis of their suitability for neural networks applications.

- **Central Processing Units (CPU)** - the most commonly found ICs that are at the core of personal computers, laptops and handheld devices. It is capable of executing a broad range of predefined instructions. As CPUs have become widely adopted in research long before the emergence of the other technologies from this list, they were the first platforms for the training and inference of neural networks with promising results back in the 1980s and 1990s for applications like high energy physics [10] or biology [11]. Although possible to achieve speed-ups of over 10x the baseline performance with careful optimizations [12], CPUs are consistently outperformed by more suitable technologies.

- **Graphic Processors (GPU)** - ICs specialized in graphics processing intended for displaying images. Since their original use case, due to the type of calculations involving matrix and vector operations, other applications related to cryptography and neural networks have also adopted GPUs as their main resource. In the former domain, cryptocurrency mining has transitioned from CPU to GPU to increase profitability [13], while for the latter, the more powerful hardware drastically reduced training and inference times, thus allowing for deeper and more complex architectures yielding higher accuracy [14, 15].

- **Application Specific Integrated Circuits (ASIC)** - as suggested by the name, those are custom designed ICs heavily specialized for a particular use. It is hard to generalize them, as the use cases can cover any modern computing problem, but the commonality is a vast improvement in performance and power usage compared to more general purpose solutions. However, the long and expensive development process pose an extremely high barrier to entry for most users. Fortunately, off-the-shelf products like the Graphcore Intelligence Processing Units [16] that are designed specifically with machine learning applications in mind as well as other custom designs [17, 18] are starting to offer a compelling platform for working with neural networks.

- **Field-Programmable Gate Arrays (FPGA)** - differently than the previous listed IC types, FPGAs are not manufactured for a specific use case, and in fact, they can be reprogrammed by a hardware designer to be a platform for a different application at any time. The reprogrammability comes at a cost of performance and power consumption compared to ASICs [19], but at the same time outperforms GPUs in these regards [20, 21]. It is also suggested, that with some technological improvements focused on ML applications, FPGAs can narrow the gap between ASICs without needing to stick to one particular design [19, 22, 23].

FPGAs offer an interesting trade-off between implementation effort and acceleration potential when it comes to neural networks and for that reason they have been chosen the target technology in this report. The following subsections give a closer look at some of their characteristics.

2.3.2 High-Level Synthesis

For many years, FPGAs have been modelled using Register-transfer level (RTL) design abstraction with the use of hardware description languages like Verilog or VHDL. However, to increase productivity and allow for a more convenient design state space exploration, a more abstract modelling process called High-Level Synthesis (HLS) can be adopted. The design can be expressed in a software programming language like C or C++, which are automatically optimized and transformed to an equivalent RTL. This is especially beneficial in research, where compared to industrial environment it is more likely that designers can afford slightly lower quality of results for increased productivity. In fact, a recent study shows that on average, only one third of design time and half of the lines of code are needed for an equivalent project done in HLS in comparison to RTL while the quality of results varies and can even outperform the RTL implementation [24].

This report's work is based on Xilinx Vivado HLS design suite. When developing a solution, it is important to note, that the synthesis process can take a significant amount of time (a couple of hours on a modern powerful computer), and so there exist two simulation methods - a C-simulation that can quickly and directly evaluate a C/C++ benchmark against the software implementation of the design, and a more truthful, cosimulation that firstly synthesizes the design and the test bench to RTL and then performs an RTL simulation. A final, definitive evaluation of the results requires programming a target FPGA with the generated bit stream of the design and exchanging input/output data with a program running on a CPU.

2.3.3 Latency, throughput, and hardware resource utilization

To understand the differences between hardware designs targeted at the same functionality, it is beneficial to consider their following characteristics:

- **Latency** - A time measure of a system between receiving an input signal and producing a *corresponding* output. It is crucial in real-time processing where it has to be lower than the period between subsequent input samples. Depending on the application, latency in the microseconds range can be expected from an FPGA.
- **Throughput** - A rate of samples processed in a unit of time. For architectures that only start to process new elements after the previous one has finished, it is equal to latency. However, in modern ICs, especially in FPGAs, it is one of the defining metrics of performance and designs tend to exploit pipelining and parallelizability to marginally trade off their latency to increase it.
- **Resource utilization** - A more complicated, often multidimensional, metric that describes the raw number or ratio of total usage of the hardware components on an FPGA. Typically, the higher it becomes, the more power is drawn by an FPGA, however, it is most often used to guide the design process to avoid running out of a certain resource and potentially deploy an alternative method that can be implemented using a different, less used one.

To fully understand the trade-offs between designs, one cannot forget about the metric related to the specific task that is accelerated in hardware. In the case of this report, classification accuracy described in subsection 2.2.3, will also play a key role in evaluating various configurations.

2.3.4 Serial, parallel, and pipelined architectures

Hardware architectures use components that can be configured in different ways depending on the overall goal or a limiting factor. The high-level configurations are displayed in fig. **??** and can be described as follows:

- **Serial** - elements are arranged in a chain, processing one after another. This way uses less resources than an equivalent parallel configuration.

- **Parallel** - elements share a common input and start processing data at the same time. This way has a lower latency than an equivalent serial configuration.
- **Pipelined** - a more sophisticated arrangement, in which subsequent processing blocks (that can be either placed serially or in parallel) form a pipeline of processing stations separated by simple storage elements. This maximizes the usage of the design blocks, hence increasing throughput with a minimal sacrifice of latency and resource usage.

serial + parallel + pipeline blocks

2.3.5 Pareto front and Roofline model

To make an informed design decision, various architectures can be compared by arranging them on a dependency graph (e.g. latency vs resource usage) and observing the Pareto front - the set of solutions for which there are no better ones with regard to one quality given the other measure is worse. The slightly complex definition can be easily understood from fig. **??**, which also highlights another use of this method - finding design configurations that are yet to be explored.

pareto front with a missing area highlighting it should be explored as there are likely interesting designs there

Another intuitive performance visualization comes in the form of the Roofline model, which compares the obtained results with theoretical limits coming from inherent hardware limitations like clock frequency or memory bandwidth. An example can be seen in fig **??**.

roofline model

some graphics to potentially add: fpga lattice, hls to rtl flow, rtl to bit stream flow

difficulty: rtl > hls > python hl4ml, draw comparison with assembly

FPGA are very hard-coded -> make the code deployable on any platform with optimal settings automatically

HLS is difficult, so coding hardware in Python is desired -> make it easy for engineers and physicists to design systems

Metaprogramming allows for optimizations and customisability

Chapter 3

Project Plan

The aims of the project cover a wide range of challenges that form subsequent steps of accelerating neural networks while raising the abstraction layers and reducing domain-specific knowledge requirements. This naturally divides the work into smaller objectives that are described in details in the following paragraphs.

Firstly, the existing transformer neural network architecture has to be redesigned to accommodate for easier adaptation to non-general-purpose hardware. This comprises splitting layers into more basic components that are easier to map to hardware and abstract about as well as introducing hooks that collect different information during training and inference passes (e.g. running mean and variance for normalization layers, tensor sizes and values). At this phase some design choices are highlighted for further inspection where simplification or improvements can be made to greatly reduce the complexity and resource usage without crippling performance.

With the adapted software implementation, the next step involves recreating the architecture in HLS. Building the initial prototype tackles the difficulties related to the underlying differences between software and hardware development and results in an accurate, yet not optimal design. From there, an iterative process begins with acceleration hypothesis firstly tested in the original software model to ensure satisfactory accuracy and then getting expressed in HLS to quantitatively measure the latency and throughput differences. That is expected to yield a highly performant solution to the initial problem that is tailored to the specific FPGA constraints.

In order to overcome the innate limitations of "hand-tuning" a solution to a problem that varies both in time and between applications, the final step of the project relies on meta-programming strategies that automatically adapt the solution according to users' criteria, available platforms and overall experiment's aim. The list of approaches that can be taken here is nearly endless, however two key areas have be designated - adjusting the model according to the existing hardware to exploit its strengths as well as allowing for more abstract representation of an architecture in a well-known machine learning framework.

As previously mentioned, some initially planned ideas have already been implemented. The distinction between these and a more detailed look at the specific project tasks can be seen in figure 3.1.

mention exams and cw deadlines

ID	Task Name	Duration (weeks)
1	MEng Individual Project	35
2	Literature review & planning & organisation	35
3	First discussion with supervisor	1
4	Literature review of software implementation	1
5	Understanding requirements and setting goals	2
6	Further literature review	13
7	Writing Interim Report and Ethics Checklist	2
8	Writing Final Report	16
9	Preparing Final Presentation	2
10	Neural Net Python implementation	7
11	Understanding existing network architecture	2
12	Testing starting software implementation	2
13	Improving and enhancing architecture	3
14	Neural Net HLS implementation	27
15	Tool for network details extraction	1
16	Tool for weights/biases encoding	2
17	Initial HLS implementation	4
18	Optimising HLS implementation for set FPGA	4
19	Contingency, debugging, polishing	16
20	Meta-programming	13
21	Automatic platform dependent optimisations	5
22	Support for PyTorch to HLS interpretation	6
23	Support for TensorFlow to HLS interpretation	2
24	Contingency, debugging, polishing	3

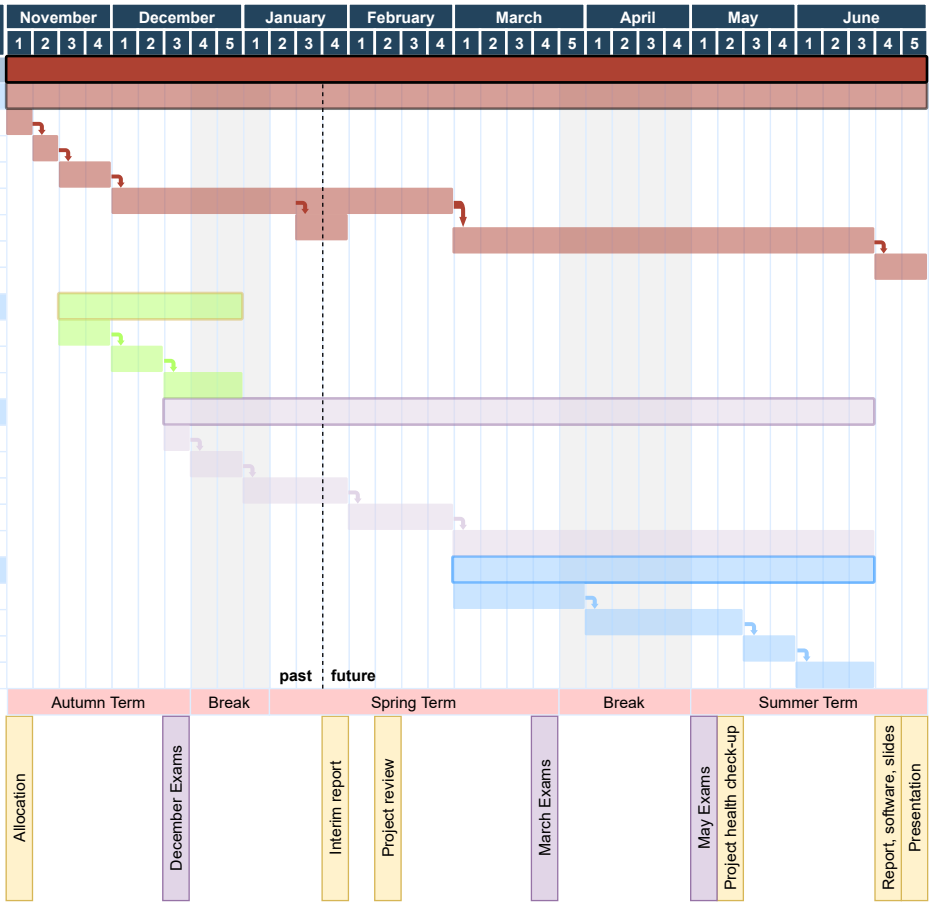

Figure 3.1: Project's Gantt chart representing initial plan of the work, past schedule has been updated to match ongoing progress accordingly

Chapter 4

Implementation

mention what has been done so far and how it relates to the project plan

Chapter 5

Evaluation Plan

This chapter outlines the proposed evaluation plan for the project. The first objective of developing and optimizing a state-of-the-art neural network in hardware can be evaluated quantitatively, while integrating it into the *hls4ml* library and making it easy for new users to use requires a more qualitative approach.

5.1 Quantitative results

The following describes the quantities that are planned to be measured:

- Classification accuracy for each designed neural network on a validation dataset
- Inference latency and throughput when running on the target platform
- Hardware resource utilization (exact values for comparison with other platforms and percentage of available resources for understanding limitations):
  - Block RAM (BRAM)
  - Ultra RAM (URAM)
  - Digital Signal Processing units (DSP)
  - Flip-Flops (FF)
  - Look-Up Tables (LUT)

In the early stages of the project, the above quantities will be measured from the results from simulation and synthesis reports. At a later stage, the best designs will be run on actual hardware platforms to validate them under real-life use cases. The platform planned for this part is an Intel Stratix V FPGA hosted in a Maxeler MPC-X dataflow node with 8 Maia dataflow engines and 48 GB of DRAM.

Apart from clear design improvements, it is predicted that most evaluated designs will offer trade-offs between classification accuracy, inference throughput and hardware utilization. It is not possible to find a design that is superior in every way, hence a **Pareto front** will play a key role in understanding the overall performance and selecting configuration with specific needs in mind. The **Roofline model** can also be used to compare the obtained results with theoretical limits coming from inherent hardware limitations like clock frequency or memory bandwidth.

area under curve, confusion matrix (true false positive nagative)

shorten pareto and roofline and maybe more coz theyre explained in background now

5.2 Qualitative results

To assess the success of enhancing the *hls4ml* library, qualitative comparisons will be drawn between it and the already existing neural network components and architectures. Depending on the project's timeline, it is possible that the improvements can get official approval and get merged, however if this is not feasible within the final deadline, current users of the library will be surveyed and their opinion will be taken into consideration instead.

Chapter 6

Ethical Considerations

The purpose of this project is to advance the next-generation particle physics experiments. There are two main aspects that need to be considered - the development of a hardware-mapped transformer neural network architecture and the easy-to-access translation and optimization toolchain for efficiently expressing networks in common machine learning frameworks.

The first feature is aimed at a purely civilian, scientific audience and it is tailored towards particle collision datasets. With that in mind, it is important to mention that, as with most machine learning research, there is potential for a misuse of the acceleration techniques towards a military or malevolent application that could negatively impact the society (issues A in table 6.1). However, this also means that there is a low risk for new emerging threats, rather the already present ones could become more serious. Fortunately, this should result in existing harm prevention measures to stay intact or solely require adjustments to their accuracy or speed thresholds.

With the second element's goal of making the creation and deployment of neural networks more accessible, it could be argued that this may in turn increase the number of physics experiments requiring high energy consumption, like those at LHC [25], thus negatively effecting the environment (issue B in table 6.1). However, this is considered a very low likely cause of action, as the research work of this project is aimed at helping already running experiments and more importantly, the negative environmental implications (for which there are various mitigation strategies [26, 27]) are heavily outweighed by potential beneficial technological advancements coming from the scientific discoveries.

Despite the aforementioned ethical issues, the project is aimed at benefitting the open-source scientific community world-wide. Its outcome could lead to a much more accessible and efficient inference methods that are applicable in many domains outside physics.

Table 6.1: Overview of potential categorized ethical issues with an indication of their applicability

	Involvement of...	Exists?
Humans	human participants	No
Personal data	personal data collection and/or processing	No
	collection and/or processing of sensitive personal data	No
	processing of genetic information	No
	tracking or observation of participants	No
	further processing of previously collected personal data	No
Animals	animals	No
Developing countries	developing countries	No
	low and/or lower-middle income countries	No
	putting the individuals taking part in the project at risk	No
Environment	elements that may cause harm to the environment, animals or plants	Yes (B)
	elements that may cause harm to humans	No
Dual use	potential for military applications	No
	strictly civilian application focus	Yes
	goods or information requiring export licenses	No
	affection of current standards in military ethics	No
Misuse	potential for malevolent/criminal/terrorist abuse	Yes (A)
	information on/or the use of sensitive materials and explosives	No
	technologies that could negatively impact human rights standards	Yes (A)
Legal	software for which there are copyright licensing implications	No
	information for which there are data protection or other legal implications	No
Other	any other ethics issues that should be taken into consideration	No

References

[1] CERN. Jets at CMS and the determination of their energy scale | CMS experiment, .

[2] Josh Cogan, Michael Kagan, Emanuel Strauss, and Ariel Schwartzman. Jet-images: computer vision inspired techniques for jet tagging. *The journal of high energy physics*, 2015(2):1–16, Feb 18, 2015. doi: 10.1007/JHEP02(2015)118. URL https://link.springer.com/article/10.1007/JHEP02(2015)118.

[3] Luke de Oliveira, Michael Kagan, Lester Mackey, Benjamin Nachman, and Ariel Schwartzman. Jet-images — deep learning edition. *The journal of high energy physics*, 2016(7):1–32, Jul 13, 2016. doi: 10.1007/JHEP07(2016)069. URL https://link.springer.com/article/10.1007/JHEP07(2016)069.

[4] Liam Moore, Karl Nordstrom, Sreedevi Varma, and Malcolm Fairbairn. Reports of my demise are greatly exaggerated: n-subjettiness taggers take on jet images. *SciPost physics*, 7(3):036, Sep 24, 2019. doi: 10.21468/SciPostPhys.7.3.036. URL https://hal.archives-ouvertes.fr/hal-01851157.

[5] Farah Fahim, Benjamin Hawks, Christian Herwig, James Hirschauer, Sergo Jindariani, Nhan Tran, Luca P. Carloni, Giuseppe Di Guglielmo, Philip Harris, Jeffrey Krupa, Dylan Rankin, Manuel Blanco Valentin, Josiah Hester, Yingyi Luo, John Mamish, Seda Orgrenci-Memik, Thea Aarrestad, Hamza Javed, Vladimir Loncar, Maurizio Pierini, Adrian Alan Pol, Sioni Summers, Javier Duarte, Scott Hauck, Shih-Chieh Hsu, Jennifer Ngadiuba, Mia Liu, Duc Hoang, Edward Kreinar, and Zhenbin Wu. hls4ml: An open-source codesign workflow to empower scientific low-power machine learning devices. Mar 9, 2021. URL https://arxiv.org/abs/2103.05579.

[6] Harvey B. Newman, Avikar Periwal, Maria Spiropulu, Javier M. Duarte, Maurizio Pierini, Eric A. Moreno, Aidana Serikova, Olmo Cerri, Jean-Roch Vlimant, and Thong Q. Nguyen. JEDI-net: a jet identification algorithm based on interaction networks. *The European physical journal. C, Particles and fields*, 80(1):1–15, Aug 14, 2019. doi: 10.1140/epjc/s10052-020-7608-4. URL http://cds.cern.ch/record/2688535.

[7] Abdelrahman Elabd, Vesal Razavimaleki, Shi-Yu Huang, Javier Duarte, Markus Atkinson, Gage DeZoort, Peter Elmer, Jin-Xuan Hu, Shih-Chieh Hsu, Bo-Cheng Lai, Mark Neubauer, Isobel Ojalvo, and Savannah Thais. Graph neural networks for charged particle tracking on FPGAs. Dec 3, 2021. URL https://arxiv.org/abs/2112.02048.

[8] Xinyang Yuan. Constituentnet: Learn to solve jet tagging through attention. Technical report, -09-22 2021.

[9] Mohammadreza Najafi, Kaiwen Zhang, Mohammad Sadoghi, and Hans-Arno Jacobsen. Hardware acceleration landscape for distributed real-time analytics: Virtues and limitations. pages 1938–1948. IEEE, Jun 2017. ISBN 1063-6927. doi: 10.1109/ICDCS.2017.194. URL https://ieeexplore.ieee.org/document/7980135.

[10] Dagli and Lammers. Possible applications of neural networks in manufacturing. page 605 vol.2. IEEE TAB Neural Network Committee, 1989. doi: 10.1109/IJCNN.1989.118423. URL https://ieeexplore.ieee.org/document/118423.

[11] Cathy Wu, Michael Berry, Sailaja Shivakumar, and Jerry McLarty. Neural networks for full-

scale protein sequence classification: Sequence encoding with singular value decomposition. *Machine learning*, 21(1):177, Oct 1, 1995. doi: 10.1023/A:1022677900508.

[12] Vincent Vanhoucke, Andrew Senior, and Mark Z. Mao. Improving the speed of neural networks on CPUs. In *Deep Learning and Unsupervised Feature Learning Workshop, NIPS 2011*, 2011.

[13] Sainathan Ganesh Iyer and Anurag Dipakumar Pawar. GPU and CPU accelerated mining of cryptocurrencies and their financial analysis. pages 599–604. IEEE, Aug 2018. doi: 10.1109/I-SMAC.2018.8653733. URL https://ieeexplore.ieee.org/document/8653733.

[14] Gang Chen, Haitao Meng, Yucheng Liang, and Kai Huang. GPU-accelerated real-time stereo estimation with binary neural network. *IEEE transactions on parallel and distributed systems*, 31(12):2896–2907, Dec 1, 2020. doi: 10.1109/TPDS.2020.3006238. URL https://ieeexplore.ieee.org/document/9130887.

[15] Qianru Zhang, Meng Zhang, Tinghuan Chen, Zhifei Sun, Yuzhe Ma, and Bei Yu. Recent advances in convolutional neural network acceleration. *Neurocomputing (Amsterdam)*, 323:37–51, Jan 5, 2019. doi: 10.1016/j.neucom.2018.09.038. URL https://dx.doi.org/10.1016/j.neucom.2018.09.038.

[16] Graphcore. Graphcore intelligence processing unit. URL https://www.graphcore.ai/products/ipu.

[17] Phil Knag, Jung Kuk Kim, Thomas Chen, and Zhengya Zhang. A sparse coding neural network ASIC with on-chip learning for feature extraction and encoding. *IEEE journal of solid-state circuits*, 50(4):1070–1079, Apr 2015. doi: 10.1109/JSSC.2014.2386892. URL https://ieeexplore.ieee.org/document/7015626.

[18] K. Venkata Ramanaiah and Cyril Prasanna Raj. ASIC implementation of neural network based image compression. *International Journal of Computer Theory and Engineering*, pages 494–498, 2011. doi: 10.7763/IJCTE.2011.V3.356.

[19] Andrew Boutros, Sadegh Yazdanshenas, and Vaughn Betz. You cannot improve what you do not measure. *ACM transactions on reconfigurable technology and systems*, 11(3):1–23, Dec 22, 2018. doi: 10.1145/3242898. URL http://dl.acm.org/citation.cfm?id=3242898.

[20] Eriko Nurvitadhi, Ganesh Venkatesh, Jaewoong Sim, Debbie Marr, Randy Huang, Jason Ong Gee Hock, Yeong Tat Liew, Krishnan Srivatsan, Duncan Moss, Suchit Subhaschandra, and Guy Boudoukh. Can FPGAs beat GPUs in accelerating next-generation deep neural networks? FPGA '17, pages 5–14. ACM, Feb 22, 2017. doi: 10.1145/3020078.3021740. URL http://dl.acm.org/citation.cfm?id=3021740.

[21] Yixing Li, Zichuan Liu, Kai Xu, Hao Yu, and Fengbo Ren. A GPU-outperforming FPGA accelerator architecture for binary convolutional neural networks. *ACM journal on emerging technologies in computing systems*, 14(2):1–16, Jul 27, 2018. doi: 10.1145/3154839. URL http://dl.acm.org/citation.cfm?id=3154839.

[22] Eriko Nurvitadhi, David Sheffield, Jaewoong Sim, Asit Mishra, Ganesh Venkatesh, and Debbie Marr. Accelerating binarized neural networks: Comparison of FPGA, CPU, GPU, and ASIC. pages 77–84. IEEE, Dec 2016. doi: 10.1109/FPT.2016.7929192. URL https://ieeexplore.ieee.org/document/7929192.

[23] Eriko Nurvitadhi, Jaewoong Sim, David Sheffield, Asit Mishra, Srivatsan Krishnan, and Debbie Marr. Accelerating recurrent neural networks in analytics servers: Comparison of FPGA, CPU, GPU, and ASIC. pages 1–4. EPFL, Aug 2016. doi: 10.1109/FPL.2016.7577314. URL https://ieeexplore.ieee.org/document/7577314.

[24] Sakari Lahti, Panu Sjovall, Jarno Vanne, and Timo D. Hamalainen. Are we there yet? a study on the state of high-level synthesis. *IEEE transactions on computer-aided design of integrated circuits and systems*, 38(5):898–911, May 2019. doi: 10.1109/TCAD.2018.2834439. URL https://ieeexplore.ieee.org/document/8356004.

[25] CERN. Facts and figures about the LHC | CERN, . URL https://home.cern/resources/faqs/facts-and-figures-about-lhc.

[26] R. Guida, M. Capeans, and B. Mandelli. Characterization of RPC operation with new environmental friendly mixtures for LHC application and beyond. *Journal of Instrumentation*, 11(07):C07016–C07016, jul 2016. doi: 10.1088/1748-0221/11/07/c07016. URL https://doi.org/10.1088/1748-0221/11/07/c07016.

[27] M. Capeans, R. Guida, and B. Mandelli. Strategies for reducing the environmental impact of gaseous detector operation at the CERN LHC experiments. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 845:253–256, 2017. doi: https://doi.org/10.1016/j.nima.2016.04.067. URL https://www.sciencedirect.com/science/article/pii/S0168900216302807. ID: 271580.

Notes