

Imperial College London

MENG INDIVIDUAL PROJECT

IMPERIAL COLLEGE LONDON

DEPARTMENT OF COMPUTING

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

Author:
Filip Wojcicki

Supervisor:
Prof. Wayne Luk

Second Marker:
Prof. Alexander Tapper

January 27, 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

List of Figures	3
List of Tables	3
1 Introduction	4
1.1 Overview	4
1.2 Motivation	4
1.3 Objectives and Challenges	4
1.4 Contributions	5
1.5 Report outline	5
2 Background and related work	6
2.1 Particle Physics	6
2.1.1 Standard Model	6
2.1.2 Particle Accelerators and triggers at LHC	7
2.2 Machine Learning	7
2.2.1 Classification accuracy, Area-Under-Curve, and confusion matrix	7
2.2.2 Deep Neural Networks	7
2.2.3 Transformer Neural Networks and Attention	8
2.2.4 Dataset	8
2.3 Reconfigurable Hardware	9
2.3.1 Landscape of Hardware for Computing	9
2.3.2 High-Level Synthesis	9
2.3.3 Latency, throughput, and hardware resource utilization	10
2.3.4 Serial, parallel, and pipelined architectures	10
2.3.5 Pareto front and Roofline model	11
3 Project Plan	13
4 Implementation	15
4.1 Adaptation of the PyTorch ConstituentNet architecture	15
4.2 Parameter extraction tool with normalization embedding	15
4.3 Implementation of ConstituentNet in Vivado HLS	16
4.4 Research into the <i>hls4ml</i> library and integration potential	16
5 Evaluation Plan	17
5.1 Quantitative results	17
5.2 Qualitative results	17
6 Ethical Considerations	18
References	20

List of Figures

2.1	Representation of different decay processes, based on the number of resulting jet clusters	7
2.2	High-level overview of the ConstituentNet architecture	8
2.3	Diagram comparing serial and parallel configurations as well as showcasing designs with and without pipelining	10
2.4	Example graph with designs plotted against quantities A/B, Pareto front highlighted	11
2.5	Example graph with computational and memory bandwidth limitations showcasing the Roofline model	12
3.1	Project's Gantt chart representing initial plan of the work, past schedule has been updated to match ongoing progress accordingly	14

List of Tables

6.1	Overview of potential categorized ethical issues with an indication of their applicability	19
-----	--	----

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 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 collimated sprays of composite particles (hadrons) that are referred to as **jets** [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 phenomenon, 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 framework. 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.

1.5 Report outline

This report begins by discussing the necessary particle physics background to understand the scope of the work, followed by the explanation and related work in the field of machine learning, with an emphasis on the state-of-the-art neural networks, and a deeper dive into the reconfigurable hardware technology in [chapter 2](#). In [chapter 3](#) and [chapter 4](#), the plan for the research is firstly outlined, followed by an up-to-date state of the implementation. Lastly, the planned evaluation metrics are discussed in [chapter 5](#), concluded by a consideration of ethical issues that might arise from the work in [chapter 6](#).

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

To be able to understand the scope of the project and the applicability of the work in modern research, this chapter gives an overview of the key concepts from particle physics that appear through the paper. The explanation is written with readers with no prior background in physics in mind.

2.1.1 Standard Model

A non-exhaustive list of the elementary particles is described, with particles that this report is concerned about (as they appear in the proton-proton collisions) being highlighted.

- Fermions
 - Leptons - participate in electroweak interactions; include electron (e^-)
 - Quarks - participate in strong interactions; include **light quarks** (q)¹ and **top (t) quark**
- Bosons
 - Gauge bosons - force carriers; include photon (γ), **W boson** (W^+ , W^-), **Z boson**, **gluons**
 - Scalar bosons - give rise to mass; include **Higgs boson** (H^0)

The information about the following decay processes form the dataset of this report, with visualization in figure 2.1 (obtained from [6]). It is important to note, that where applicable, the particles on the left-hand side of the arrows undergo a series of decays before reaching the right-hand side, when the only particles left are those composed of quarks and antiquarks (denoted by the vertical bar), referred to as hadrons.

$$q/g$$

$$H^0/W/Z \rightarrow q\bar{q}$$

$$t \rightarrow Wq \rightarrow q\bar{q}q$$

¹Light flavor quarks: up (u), down (d), charm (c), and strange (s) quarks

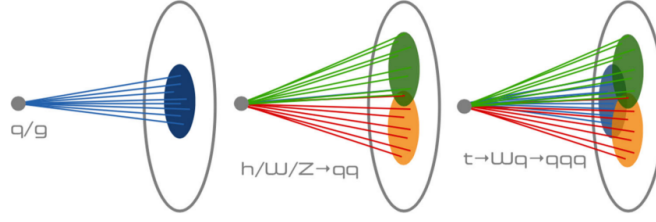


Figure 2.1: Representation of different decay processes, based on the number of resulting jet clusters

2.1.2 Particle Accelerators and triggers at LHC

The two experiments that are of most concern in this paper are CMS and ATLAS. They are both large general-purpose particle detectors, that were notably involved in the discovery of the Higgs boson [9]. For real-time processing, the detectors are composed of triggers split into levels [10, p.16]:

- **Level 1 trigger (L1T)** - it is implemented in hardware (FPGAs) and firmware, pipelined (a term explained in details in subsection 2.3.4) and it cannot allow for any dead-time, i.e. has to continuously process data with a fixed latency.
- **Level 2 trigger (L2T)** - it is implemented in hardware and software and can include regional processing.
- **Level 3 trigger (L3T)** - it is implemented in software, using farms of CPUs. It is close in behavior to non-real-time algorithms.

Since their origin, the L2T and L3T have been merged into High Level Trigger (HLT) [11, p.47], which is planned to rely on thousands of multithreaded CPUs and GPUs. As for the L1T key specifications that will be used to evaluate the design in this paper, its input data frequency is 40 MHz, which with a pipeline depth of 500 results in a $12.5 \mu s$ latency, and its output frequency is equal 750 kHz.

2.2 Machine Learning

The already mentioned neural networks belong to a wider field of machine learning (ML) - the study of using experience to improve algorithms. This section assumes a basic understanding of ML and gives a brief overview of the topics needed to understand the scope of the project. It then explains in more details the background and related work for the architectures involved in this research.

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

There are several key metrics used for assessing the success of an ML algorithm, and the following will be used throughout the report:

- **Classification accuracy** - a simple measure of the percentage of correctly classified samples.
- **Confusion matrix** - a tabular metric that compares the actual samples classes with the predicted ones, effectively dividing results into four groups: true positive, false positive, false negative, and true negative. This allows for an easy calculation of precision and recall values.
- **Area Under the Curve (AUC) for the Receiver Operator Characteristic (ROC)** - a more complex measure of the model's ability to correctly distinguish between classes. It can be used similarly to the classification accuracy, but it favors discriminative over representative models.

2.2.2 Deep Neural Networks

While there exist a number of ML techniques that has proven successful for various use cases at LHC, like Support Vector Machines [12] or Boosted Decision Trees [13], in the last years deep neural

networks (DNN) have been proposed with improved results for applications like infrastructure monitoring [14], offline data analysis [15], and the main interest of this report - detector trigger mechanisms.

In many use cases the neural networks architectures are optimized and accelerated to shorten the training time (measured in hours) to reduce the time needed for evaluating different design configurations and easily perform the hyperparameter search. However, this work focuses on accelerating the inference to match the extremely low latency required in the LHC detectors' L1 triggers. Although often measured in milliseconds, sub-milliseconds inference time has been achieved for this application with the use of FPGAs using architectures for basic DNN [16], and recently sub-microseconds latency for graph neural networks (GNN) [17, 18]. These implementations serve as a baseline latency for this project which aims to achieve comparable performance with higher AUC value. A commonality between the recent best performing designs is the use of the *hls4ml* codesign workflow that was mentioned in [section 1.2](#).

2.2.3 Transformer Neural Networks and Attention

A promising architecture that has been chosen as the implementation for this project is the transformer neural network, which is a type of recurrent neural network (RNN). A recent implementation [8] called ConstituentNet outperforms previous state-of-the-art GNN implementations like JEDNet [6] using a version of the attention mechanism [19], called self-attention. A diagram with a high-level view of the ConstituentNet can be seen in [figure 2.2](#).

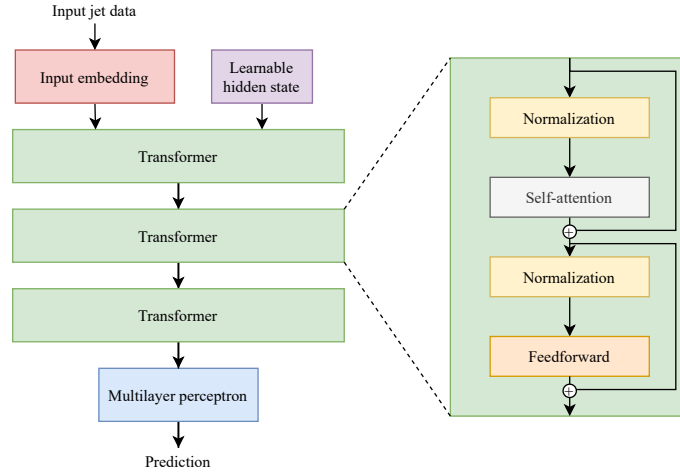


Figure 2.2: High-level overview of the ConstituentNet architecture

Given the strong results of its software implementation, an FPGA-mapped design can become an improvement over existing designs for L1 triggers at LHC. A number of techniques is planned to ensure the design matches the required performance, which includes quantization and pruning [20] and careful exploration of the trade-off between latency and hardware resource utilization, which is covered in [subsection 2.3.3](#).

2.2.4 Dataset

The datasets used in this work contains has been obtained from the 13 TeV proton-proton collisions performed at LHC, and it includes information about the most energetic jets [21, 22, 23, 24] that were constructed using the anti- K_t clustering algorithm [25]. A number of representations is available in the dataset and the one used for this work contains lists of jets' constituent hadrons from the following aforementioned hadrons: light quarks, top quarks, W bosons, Z bosons, and gluons.

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

The modern landscape of digital integrated circuits (IC) is very rich and can be divided into numerous categories depending on the technology used and expected functionality [26]. 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 [27] or biology [28]. Although possible to achieve speed-ups of over 10x the baseline performance with careful optimizations [29], 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 [30], 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 [31, 32].
- **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 poses an extremely high barrier to entry for most users. Fortunately, off-the-shelf products like the Graphcore Intelligence Processing Units [33] that are designed specifically with machine learning applications in mind as well as other custom designs [34, 35] 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 [36], but at the same time outperforms GPUs in these regards [37, 38]. 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 [36, 39, 40].

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 [41].

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 2.2.1](#), 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 [figure 2.3](#) and can be described as follows:

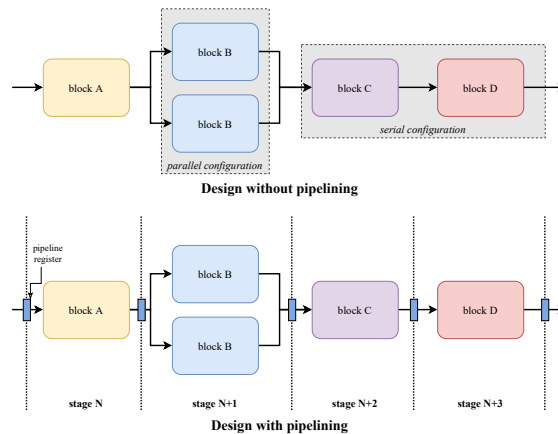


Figure 2.3: Diagram comparing serial and parallel configurations as well as showcasing designs with and without pipelining

- **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.

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 figure 2.4, which also highlights another use of this method - finding design configurations that are yet to be explored.

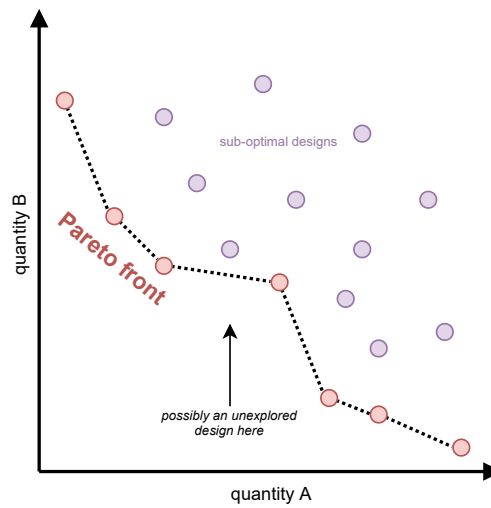


Figure 2.4: Example graph with designs plotted against quantities A/B, Pareto front highlighted

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 2.5.

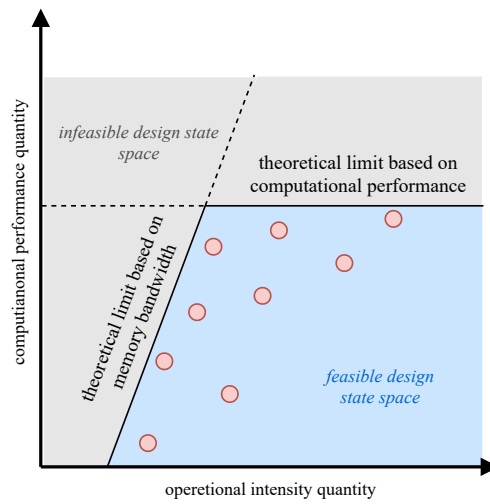


Figure 2.5: Example graph with computational and memory bandwidth limitations showcasing the Roofline model

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 been 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. It is important to note the project milestones and exam sessions at the bottom of the diagram, as they give a better context of the spread of the working time. Although very time-consuming, coursework deliverables have not been highlighted, as all combined they occur in every week of the Spring term and would introduce too much information clutter.

Chapter 4

Implementation

As depicted in the figure 3.1, part of the project plan from chapter 3 has already been implemented as of the time of publishing this report. This was done thanks to the smaller workload of the Autumn term in comparison to the Spring term as well as significant effort over the Winter break. This 'head start' is hoped to allow for a deeper state space exploration and a more refined final architecture and in case of faster than expected working pace, further extensions related to the *hls4ml* library and automatic optimizations will also be considered. The accomplishments so far can be categorized into four domains covered in the following sections.

4.1 Adaptation of the PyTorch ConstituentNet architecture

- thanks to an existing code base with the implementation, it was easier to understand the smaller details that were not fully explained in the original paper [8]. However, many aspects of the provided code served as proof-of-concept and are suspected to have been changed after the publishing, as a new model could not be trained, nor the provided one could be evaluated. Without the help of the original author, a severe investigation and fixing process were required to progress the software implementation into a usable state. Despite those difficulties, the time was well spent on finding potential optimization points for the later stage of the project. Moreover, frequent reporting hooks were added in between the existing network layers, which allows for generating an inner view of the calculations happening on the CPU that gives the opportunity for direct, step-by-step comparison with the HLS implementation. Ultimately, the code base has reached a state where it is convenient to train models with different parameters and evaluate them against the datasets.

4.2 Parameter extraction tool with normalization embedding

- in order to generate files with weights and biases that are required for initializing the memory on an FPGA, a tool was developed that takes a PyTorch pre-trained model, extracts all the information, and splits them accordingly with the required format. What is more, layers responsible for normalization can be chosen to have their mean and variance calculation embedded into weights and biases to significantly reduce the processing required on an FPGA by omitting the division and square root operations. The mathematical derivation of this approach starts with the batch norm formula:

$$y = \frac{x - E[x]}{\sqrt{Var[x] + \epsilon}} * \gamma + \beta$$

The expected value and variance are treated as learnable parameter of a dataset and are extracted after the training has been completed. Hence, the calculation becomes:

$$y = x * \frac{\gamma}{\sqrt{Var + \epsilon}} + \beta - \frac{\gamma * E}{\sqrt{Var + \epsilon}} = x * W + b$$

The newly calculated values for W and b represent the updated weights and biases of the normalization layer, that can be then implemented in hardware in a much simpler way. Independently

of the implementation in this work, a similar idea has been proposed and successfully used as an optimization in the past [42].

4.3 Implementation of ConstituentNet in Vivado HLS

- an effort has been made to express the original version of ConstituentNet in HLS based off an empty skeleton code generated by *hls4ml*. So far, the architecture, including the blocks responsible for the transformer and the self-attention calculations, has been designed properly connected. Several layer types, namely the densely-connected, normalization and activation ones have already been successfully validated. A variant of the densely-connected layer used for matrix-matrix multiplication, along with a few smaller layers, are yet to be validated, at which point, the architecture would be ready for an evaluation against the PyTorch implementation.

4.4 Research into the *hls4ml* library and integration potential

- the current HLS implementation benefits from a number of existing layers included as part of the *hls4ml* library. This has been done as a way to shorten the time needed for the initial HLS implementation that will serve as the baseline for further optimizations as well as to research the structure and methods used in the library to allow for a smoother integration into the codesign workflow once the previous goal is met.

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 to be measured for each neural network design:

- Classification accuracy, AUC and confusion matrix 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) and 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. A consideration is also planned for the specific hardware used in the LHC L1T detectors and its available resources, which cannot be directly tested on, but can guide the design state space exploration.

Apart from clear design improvements, it is predicted that most evaluated designs will offer trade-offs between classification accuracy, AUC, inference throughput and hardware utilization. It is not possible to find a design that is superior in every way, hence a Pareto front and the Roofline model will play key roles in understanding the overall performance and selecting configuration with specific needs in mind.

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 [43], 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 [44, 45]) 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](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](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] Brian Greene. How the higgs boson was found, July 2013. URL <https://www.smithsonianmag.com/science-nature/how-the-higgs-boson-was-found-4723520/>.
- [10] Trigger, DAQ and FPGAs. URL <http://www.hep.ph.imperial.ac.uk/~tapper/lecture/trigger.pdf>.
- [11] Alex Tapper. Triggering at collider experiments. URL <http://www.hep.ph.imperial.ac.uk/~tapper/lecture/CMSIndia-2020.pdf>.
- [12] G. Valentino, R. W. Assmann, R. Bruce, and N. Sammut. Classification of LHC beam loss spikes using support vector machines. pages 355–358. IEEE, Jan 2012. doi: 10.1109/SAMI.2012.6208988. URL <https://ieeexplore.ieee.org/document/6208988>.

- [13] Tianqi Chen and Tong He. Higgs Boson Discovery with Boosted Trees. In Glen Cowan, Cécile Germain, Isabelle Guyon, Balázs Kégl, and David Rousseau, editors, *Proceedings of the NIPS 2014 Workshop on High-energy Physics and Machine Learning*, volume 42 of *Proceedings of Machine Learning Research*, pages 69–80, Montreal, Canada, 13 Dec 2015. PMLR. URL <https://proceedings.mlr.press/v42/chen14.html>.
- [14] Andrzej Skoczen, Maciej Wielgosz, and Matej Mertik. Using LSTM recurrent neural networks for monitoring the LHC superconducting magnets. Technical Report 867, Elsevier B.V, Nov 18, 2016. URL <http://cds.cern.ch/record/2234465>.
- [15] Jie Ren, Lei Wu, and Jin Min Yang. Unveiling CP property of top-higgs coupling with graph neural networks at the LHC. *Physics letters. B*, 802:135198, Mar 10, 2020. doi: 10.1016/j.physletb.2020.135198. URL <https://dx.doi.org/10.1016/j.physletb.2020.135198>.
- [16] Edward Kreinar, Jennifer Ngadiuba, Zhenbin Wu, Philip Harris, Maurizio Pierini, Ryan Rivera, Song Han, Javier Duarte, Benjamin Kreis, Nhan Tran, and Sergo Jindariani. Fast inference of deep neural networks in FPGAs for particle physics. Technical Report 13, Institute of Physics (IOP), Apr 16, 2018. URL <http://cds.cern.ch/record/2316331>.
- [17] Edward Kreinar, Zhenbin Wu, Gianluca Cerminara, Kinga Wozniak, Gerrit Van Onsem, Marcel Rieger, Giuseppe Di Guglielmo, Jan Kieseler, Shah Rukh Qasim, Sioni Summers, Sergo Jindariani, Jennifer Ngadiuba, Mia Liu, Philip Harris, Maurizio Pierini, Vladimir Loncar, Kevin Pedro, Yutaro Iiyama, Javier Duarte, Dylan Rankin, Nhan Tran, and Abhijay Gupta. Distance-weighted graph neural networks on FPGAs for real-time particle reconstruction in high energy physics. Technical Report 3, Frontiers Media S.A, 2020. URL <http://cds.cern.ch/record/2728798>.
- [18] 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>.
- [19] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. Jun 12, 2017. URL <https://arxiv.org/abs/1706.03762>.
- [20] Tailin Liang, John Glossner, Lei Wang, Shaobo Shi, and Xiaotong Zhang. Pruning and quantization for deep neural network acceleration: A survey. *Neurocomputing (Amsterdam)*, 461:370–403, Oct 21, 2021. doi: 10.1016/j.neucom.2021.07.045. URL <https://dx.doi.org/10.1016/j.neucom.2021.07.045>.
- [21] Maurizio Pierini, Javier Mauricio Duarte, Nhan Tran, and Marat Freytsis. HLS4ML LHC jet dataset (30 particles), . URL <https://doi.org/10.5281/zenodo.3601436>.
- [22] Maurizio Pierini, Javier Mauricio Duarte, Nhan Tran, and Marat Freytsis. HLS4ML LHC jet dataset (50 particles), . URL <https://doi.org/10.5281/zenodo.3601443>.
- [23] Maurizio Pierini, Javier Mauricio Duarte, Nhan Tran, and Marat Freytsis. HLS4ML LHC jet dataset (100 particles), . URL <https://doi.org/10.5281/zenodo.3602254>.
- [24] Maurizio Pierini, Javier Mauricio Duarte, Nhan Tran, and Marat Freytsis. HLS4ML LHC jet dataset (150 particles), . URL <https://doi.org/10.5281/zenodo.3602260>.
- [25] Matteo Cacciari, Gavin P. Salam, and Gregory Soyez. The anti-kt jet clustering algorithm. *The journal of high energy physics*, 2008:063, Apr 1, 2008. doi: 10.1088/1126-6708/2008/04/063. URL <http://iopscience.iop.org/1126-6708/2008/04/063>.
- [26] 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>.
- [27] 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>.

- [28] 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.
- [29] 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.
- [30] 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>.
- [31] 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>.
- [32] 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>.
- [33] Graphcore. Graphcore intelligence processing unit. URL <https://www.graphcore.ai/products/ipu>.
- [34] 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>.
- [35] 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.
- [36] Andrew Boutros, Sadeh 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>.
- [37] 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>.
- [38] 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>.
- [39] 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>.
- [40] 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>.
- [41] Sakari Lahti, Panu Sjoval, 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>.
- [42] Hongxiang Fan, Shuanglong Liu, Martin Ferianc, Ho-Cheung Ng, Zhiqiang Que, Shen Liu, Xinyu Niu, and Wayne Luk. A real-time object detection accelerator with compressed SSDLite

- on FPGA. pages 14–21. IEEE, Dec 2018. doi: 10.1109/FPT.2018.00014. URL <https://ieeexplore.ieee.org/document/8742299>.
- [43] CERN. Facts and figures about the LHC | CERN, . URL <https://home.cern/resources/faqs/facts-and-figures-about-lhc>.
- [44] 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>.
- [45] 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.