Accelerating Trigger Performance of the ALICE Detector Using FPGA-Based Neural Networks

Steven Thijs Bos

Dated: July 20, 2023

A Master's thesis
Under supervision of:
Dr. Alessandro Grelli and Marcel Rossewij

Institute for Gravitational and Subatomic Physics
Utrecht University
The Netherlands

1 Abstract

The newly proposed heavy-ion detector at CERN Large Hadron Collider, ALICE3, will face a hundredfold increase in data rate due to the increased multiplicity and luminosity for both the proton-proton(pp) and lead-lead (Pb-Pb) collisions. Current CPU and GPU hardware combinations account for only 3.5 Gb/s, more than an order of magnitude lower than future demands. This research explores the use of machine learning algorithms on custom Field-Programmable Gate Arrays (FPGAs), a combination not yet used for hardware triggers in particle physics experiments. The aim of this study is to investigate the performances of FPGAs, which promise fast decision-making within an affordable power budget. The performance will be tested by deploying a neural network for the classification of simulated $\pi^{\pm}$ and e^{+} shower events in the Epical-2 prototype calorimeter. By using a cost-effective FPGA with the Vitis AI software, our approach provides a low entry barrier while simultaneously allowing for scalability. The results suggest that the combination of machine learning algorithms and the custom hardware can significantly improve the latency performance, posing as a promising candidate for future particle physics experiments.

Contents

2 Introduction

The Big Bang theory on the beginning of the early universe supported by the Hubble-Lemaitre theory[1][2], gave humanity a scientific understanding of the origin of our cosmos. The expansion of the universe as observed by Hubble continues to this day, decreasing the temperature and density of the universe. Moments after the Big Bang in the early universe, the density of matter and temperature were so high that particles were not able to form, leaving quarks in a state of deconfinement. This dense hot plasma is what we call the quark-gluon plasma(QGP). In this hot primordial soup, quarks and gluons move freely, without being bound together. After a short period of time while the universe expanded, the density and temperature dropped below the so-called Hagedorn temperature $T_H = 1.66 \times 10^{12}$K[3], allowing for the strong interaction to hadronize the plasma constituents to particles. The ability to access the QGP, investigating its properties, can be very insightful in learning about the origin of matter and mass as we know it in the current cosmos. The interactions that occur in this phase of matter hold clues related to the Quantum-Chromo-Dynamics (QCD), the theory that deals with the strong interaction. QCD has been studied for decades and, as a well-established theory, its properties on the high temperature and/or density limit are yet to be fully understood. The matter that makes up the world around us is interacting in ways that are yet unknown. Experiments that recreate the high-energy density conditions of the early universe allow us to probe what happened right after the big bang. These experiments involve colliding heavy ions at relativistic velocities and subsequently investigating all the fragments produced. Recently, heavy-flavor probes have gained great attention in the characterization of QGP, being seen as a self-calibrating tomography tool. However, heavy-flavor particles are rare. To come up with a result on these rare-physics decay channels that is significant statistically, the experiment has to integrate a large statistic (i.e. record large numbers of collisions). Therefore, a large number of collisions have to be investigated, with each collision being very CPU and disk space demanding. This continuously increases the data output and through-output for analysis related to these types of experiments. With the end of Moore's law approaching and the Dennard Scaling failing[4], the computational power required in the investigation of the QGP will soon be unmanageable. Therefore, a reduction and compression of information is unavoidable and we would be interested in continuing this research feed. This thesis explores the use of neural networks in FPGA triggers to reduce the amount of data for analysis in a smart way, making an ultra-fast selection of events that are important for the experiment, while discarding all the others (which constitute the vast majority of all the events). This first-level trigger has the potential to significantly reduce the amount of computational power and data storage required in the experiment, both in a fast and energy-efficient manner, without reducing the significance of the statistics of the experiment.

2.1 Standard model

The Standard model is the theoretical framework that includes three of the four fundamental forces: strong, weak, and electromagnetic. Particles are divided into two categories: fermions and bosons. Fermions are half-integer spin particles that behave according to Fermi-Dirac statistics. Bosons are integer spin particles that obey the Bose-Einstein statistics. Fermions are classified into six quarks and six leptons. Leptons carry an integral electric charge. The electron e with unit negative charge being the most familiar to most people, the muon μ and the tauon τ are the heavier versions of the electron. Neutral leptons, called neutrinos ν, are paired with the flavor of the charged lepton. For particle decays into charged leptons, a paired neutrino of the same flavor is also created.
The quarks have a fractional charge of $-\frac{1}{3}e$ or $+\frac{2}{3}e$ and come in six different flavors: up, down, strange, charm, bottom, top quarks and their respective antiquarks. The flavors are grouped in pairs that have charges that differ by one unit of electric charge. Each quark flavor has its own unique configuration of spin, mass, charge, and color charge. Quarks do not come alone due to color-confinement as described by Quantum Chromo Dynamics; they come together forming hadrons, unlike leptons that can appear alone. The quarks interact via the strong interaction but also by electromagnetism and weak interaction, making it the only category of particles that are subject to all three forces of the standard model. Leptons, on the other hand, do not interact by the strong interaction.
The force carriers of the standard model belong to the bosons. There are four spin-1 gauge bosons: photons, electrically neutral, the weak boson Z^0, two electrically charged weak bosons $W^{\pm}$ and gluons that come in eight different color charges carrying the strong interaction. There is also a massive scalar gauge boson, called the Higgs boson, which has zero spin, even parity, and no electric or color charge. The Higgs particle is involved in the Higgs mechanism. The Higgs mechanism is triggered when electroweak symmetry is broken, generating masses for the weak gauge bosons[5]. The strong interaction binds the quarks together, forming neutrons and protons, for instance. Electromagnetic interaction that binds electrons to nuclei is mediated by the exchange of massless photons. The weak interaction that is conducted through the $W^{\pm}$ and Z^0 boson is involved in for instance nuclear β-decay, which is a relatively slow process where a radioactive nucleus emits an electron and neutrino. The mass of the weak interaction bosons is relatively high, being on the order of 100 times heavier than a proton mass.
The interaction of gluons is described by QCD, a field theory that is expressed through a Lagrangian that is invariant under the non-Abelian SU(3) symmetry. QCD has three important features:

- colorconfinement. The force between two color charges is constant at separation, building up until there is enough energy for a quark-antiquark to be born. This prevents the isolation of a color charge, making the global color charge neutral.
- asymptotic freedom. Causes strong interactions between particles to become asymptotically weaker at higher energy scales and shorter distances.
- chiral symmetry breaking. The spontaneous symmetry breaking of chiral symmetry as observed for neutrons favoring one chiral configuration over the other, breaking the macroscopic symmetry that particles should have the same laws of physics also for the mirror image of the particle.

The gluon interactions that are responsible for these three QCD features are the three and four gluon self-interactions. These are some of the interactions that are of interest for experiments to gain insight into the nature of QCD.

2.2 ALICE detector

The ALICE (A Large Ion Collider Experiment) experiment[6] focuses on the investigation of QGP properties and doing so on the investigation of QCD in extreme conditions. By smashing heavy ions together at relativistic energies, the QGP is produced. When the nuclei collide, the energy density is high enough in a region larger than the size of the hadron, deconfining the matter. From the remnants of the collisions, the detector can backtrack and investigate the event[7]. This experiment involves energies of the order of TeV on time scales of fs with particle counts of millions spread over 19 subdetectors in total. Over a ring of 27 km, located 100 m underground at the Large Hadron collider at CERN particle beams are accelerated to velocities near the speed of light. The partons travel in opposite directions to collide at center-of-mass energies up to 5.02 TeV, guided by superconducting electromagnets. At a frequency of $\sim$ 50 Khz, lead ions collide in the center of the inner barrel in detector systems. There are four detector experiment systems at CERN, one of which is ALICE, the others ATLAS, CMS, and LHCb that focus on exploring different questions in high-energy physics.

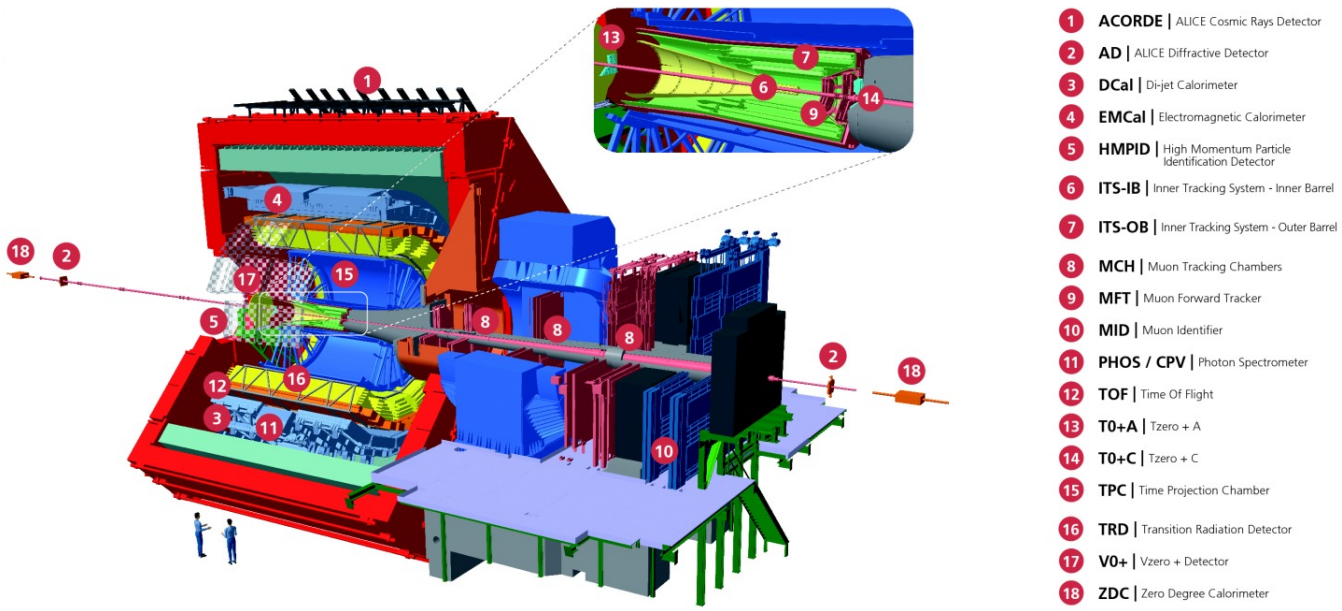


Figure 1: The ALICE detector[6]

The central part of the detector around the beam, the Inner-Tracking-System (ITS) together with the Time Projection Chamber (TPC) and the Transition Radiation Detector (TRD) make up the detectors for the reconstruction of the particle tracks. The ITS is closest radially to the particle beam, the innermost layer being 23 mm from the interaction point[8]. This tracker consists of six silicon layers and is used for low-momentum tracking, particle identification (PID), and primary and secondary vertexing. The short-lived heavy particles that pass through the detection layers allow for measurement of their properties and positions tracks. The ITS is surrounded by the TPC, extending $87 < r < 247$ cm outward from the interaction point, responsible for particle tracking and PID, together with determination of the decay vertices of the hadrons and leptons. The TPC barrel is filled with a gas composition, which is the detection medium. The choice of the gas composition determines the performance of the detector, since the charge transport in the drift volume and the amplification processes in the readout chamber depend on it. The identification of particles in the TPC is characterized by the particles energy loss in the detection medium. The liberated electrons in the medium drift to the end plates of the cylinders, where they are read out by Multi-Wire Proportional Chambers (MWPCs). Surrounding the TPC is the TRD arranged in 6 layers at radial distance from 2.90 m to 3.68 m from the beam axis, which is used for electron identification and tracking. The detection gas in the TRD is a mixture of xenon and carbon dioxide, where the energy deposits of X-ray are measured. The x-rays are emitted when electrons and positrons pass through the many thin detector layers of the TRD, due to the change in dielectric constants over the different materials. The TOF is wrapped around the TRD, allowing for charged hadron identification. The time of flight of particles from the interaction point to the TOF detector provides additional information on the PID process. The velocity of a charged particle is measured, which when combined with a momentum measurement can derive the particle mass. The TOF detector consists of 160000 multigap resistive plate chambers (MRPC) with a time resolution of about 100 ps. The MRPCs are constructed of two parallel window glass plates that create a narrow gas gap with an electric field. In figure 2 the performance of the particle identification process in the TPC and TOF is shown. Separation of particle species by combining information from both detectors is enabled in the momentum regions below 4 and 2.5 GeV/c, respectively. The ElectroMagnetic Calorimeter (EMCal), the PHOton Spectrometer (PHOS), and the High Momentum Particle Identification Detector (HMPID) are used for electron, jet, photon, and hadron identification. The EMCal is a lead scintillator sampling calorimeter responsible for detecting high-energy jets. The EMCAL is extended with the Di-jet Calorimeter (DCal) expanding the capabilities to measure di-jets, together forming a two-arm electromagnetic calorimeter. The PHOS is an electromagnetic calorimeter build-up of lead-thungstate crystals, measuring electromagnetic radiation. Photons and neutral mesons from radiative decays and strongly interacting matter can be detected in the range of 0.1 to 100 GeV/c. HMPID identifies high-momentum charged particles through the detection of Cherenkov radiation. The particles that pass through the liquid radiator medium C_6F_{14} produce Cherenkov light, which is captured by an MWPC. The HMPID is capable of separating pions and kaons up to $p_T = 3\text{GeV}/c$, and Kaons and (anti-)protons up to $p_T = 5\text{GeV}/c$, with separation of three sigma. In order to

identify muon particles, a wall is separating the muon tracker and the muon trigger. This wall denies all particles except the muon particles that penetrate everything. The distinction between muons and the other particles is made by checking which particle tracks are not stopped by the wall. This muon detector is in the forward region of the detector. Sub-Detectors that determine the power level of the collisions, the number of particles produced, and measurements of collision times are the T0, V0, FMD and Zero Degree Calorimeter (ZDC), placed at small angles on both sides of the interaction region.
The experiment relies on the identification of all the particles that are created in the collision and their respective charges. For this task, the detector is equipped with 19 different detectors to make use of the characteristics of the particles in the identification process with each subdetector dedicated for different particle measurements. The trajectories of particles resulting from the collision are reconstructed, including the decay products that form the daughter particles. The characteristics of the daughter particles can differ from the parent particles, leaving observable traces in the other sub-detectors. For example, consider the decay of a short-lived neutral pion π^0 into two photons γ. The photons, being electrically neutral, do not interact strongly with the detectors tracking system, unlike the neutral pion. Therefore the electromagnetic calorimeter is in place to measure electromagnetic particle energies. The photons deposit energy in the calorimeter, creating clusters or showers of particles. By combining these energy deposit data together with information from the other sub-detectors, the decay of the neutral pion can be reconstructed.

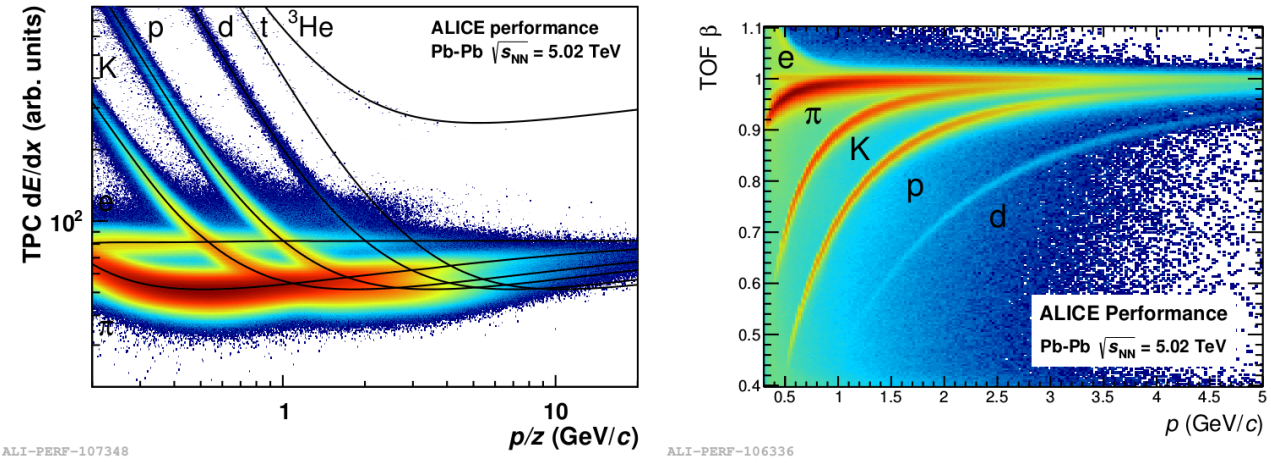


Figure 2: Particle identification by energy loss dE/dx in the TPC on the left, and by TOF on the right.[9]

In general, by consolidating data from all the subdetectors it becomes feasible to reconstruct the entire collision event. This enables the identification of each particle and its charge, from its tracks, contributing to the comprehensive visualization of the event as a whole.

2.3 ALICE upgrades

2.3.1 ALICE Run 4

To achieve the optimal experimental conditions to study the QGP in the LHC, upgrades to the ALICE detector are needed. High-statistics and high-precision measurements will give access to the rare-physics channels that are needed to understand the dynamics of the condensed phase of QCD. The ALICE collaboration is planning to upgrade the current detector by enhancing its low-momentum vertexing and tracking capability. Not only will the detector resolution be upgraded, the luminosity with Pb beams will be progressively upgraded, and the interaction rate will be increased to 50 kHz, resulting in luminosities of $L = 6\text{x}10^{27}$ cm$^{-2}s^{-1}$. This will drastically increase the statistics of the experiment but also substantially increase the data rates. To achieve a higher-precision measurement, the proposed upgrades for the Run 4 ALICE detector are as follows.

- A new high resolution, low-material-thickness Inner Tracking System (ITS)[10]. The inner layer of ITS3 will be at 18 mm from the collision point, closer than the current ITS2 that is at 23 mm. Additionally, the low material budget improves tracking by reducing the probability of particle absorption and multiple scattering by the detector material.

- The Foward Calorimeter (FoCal)[11] will be added as an upgrade to the ALICE experiment. A highly granular Si+W electromagnetic calorimeter combined with a hadronic calorimeter covering the range of pseudo-rapidities $3.4 < \eta < 5.8$.

Due to the increased collision rate of 50Khz at peak heavy-ion events, the readout electronics of the Transition-Radiation Detector, Time-Of-Flight detector, Photon Spectrometer, and muon spectrometer have to be improved. Not only the readout electronics, but also the forward trigger detectors need to be improved. The higher rate of operation calls for an improvement in the latency at which the triggers operate, in order to decide if the event is to be saved on the data server for further analysis. An upgrade to the offline data processing software is also required to cope with the reconstruction and analysis of the increased statistics.

ITS upgrade
High precision measurements of charm and beauty quarks in the low momentum region are the goal of the new ITS upgrade for Run 4. By reducing the radial distance down to 18mm between the interaction point and the first layer of the ITS, the measurement is improved. For the upgrade of the ITS the inner three detector layers will be replaced by silicon-only layers. The material budget is reduced by a factor of six to $0.05\%X_0$ per layer reducing the amount of materials that could impede the detection of particles. By positioning the initial detection layer closer to the interaction point at 18 mm and minimizing the material budget, the upgraded vertex detector will greatly improve tracking accuracy and efficiency for low transverse momentum p_T particles. This enhanced vertexing capability will improve the measurement of charm and beauty hadrons with low p_T, as well as the measurement of low-mass and low p_t dielectrons. The proposed ITS3 will consist of two half-barrels consisting of each three half-layers. Each half-layer consists of a single large pixel silicon chip which is curved to the cylindrical shape. The half barrels are supported with additional lightweight carbon structures. Making a single large-pixel silicon chip is a novel idea, enabled by stitching technology. Stitching technology is to fabricate an image sensor that is larger than the field of view of the lithographic equipment. Wafer scale sensors can be manufactured this way by placing reticles on the wafer with high precision, achieving a tiny well defined overlap. By plasma polishing the surface of the silicon the mechanical stress of the materials is released, allowing for the possibility to bend and operate ultrathin sensors. Therefore, the construction of a silicon-only cylindrical layer is made possible. Manipulation of silicon to achieve such extreme bending angles represents cutting-edge technology, involving thorough investigations into the effects on signal transfer capabilities. The heat dissipated by the sensors is removed by convection through forced air flow between the layers. Carbon foam rings in thermal contact with the sensors act as radiators, reducing the thermal gradient along the layer. To enhance the convective heat transfer, carbon foam is used. Due to the fact that the surface area of the volume of carbon foam is as large as 5000 to 50000 m^{-1}, where the air can pass through while dissipating heat. The air flow through the foam can be disrupted due to the internal structure of the foam, causing a high air pressure drop. In the design process, the optimization of air flow is crucial while avoiding airflow-induced vibrations that could cause mechanical instability. As part of the ALICE ITS upgrade, the ALPIDE sensor[12], specifically designed for this purpose, is introduced. This sensor is engineered to operate at an increased read-out rate of 100 kHz specifically for Pb-Pb collisions.
FoCal
The FoCal detector will allow the measurement of small-x-gluon distributions by prompt-photon production. The FoCal provides the unique capability to investigate Parton Distribution Functions by means of the high-precision measurement of direct photons and jets, as well as gamma-jet and jet-jet events. This will allow for the exploration of non-linear effects at small x that is yet unprecedented. These effects are part of the non-Abelian nature of QCD that contributes to the investigation of the strong interaction. The FoCal detector will be located outside the ALICE magnet at a distance of 7 m from the interaction point. The electromagnetic calorimeter will consist of 18 layers of tungsten absorbers and silicon sensors. 16 out of 18 layers are low-granularity silicon sensors with fast integration time for charge collection. The remaining 2 layers are placed at layers 5 and 10 where the electromagnetic shower reaches its maximum, are equipped with slower integration time high-granularity active pixels based on the ALPIDE sensor technology. These layers are responsible for the identification of overlapping particle showers.

2.3.2 ALICE 3

The upgrades to the ALICE experiment for Run 4 will put a significant effort towards the understanding of the QCD, however the improvements will not be sufficient in answering several fundamental questions. The limited precision of the anticipated outcomes from Run 4 in the beauty sector hinders the comprehensive investigation of the transport and hadronization characteristics of beauty quarks. The combination of uncorrelated quarks in the

process of hadronization of the QGP is crucial for unraveling the dynamics that link hadronization and collective flow. The unified description of parton energy loss, collective flow, hadronization, and electromagnetic radiation will not be possible with the current experimental program. A novel experiment towards the understanding of electromagnetic radiation and heavy-flavor studies is needed. Systematically measuring heavy-flavor hadrons would allow for the study of effects due to the transport properties of QGP by the harmonization mechanisms. To get sufficient statistics on rare events that are key topics in the study of QGP requires even higher rate capabilities of the ALICE experiment.

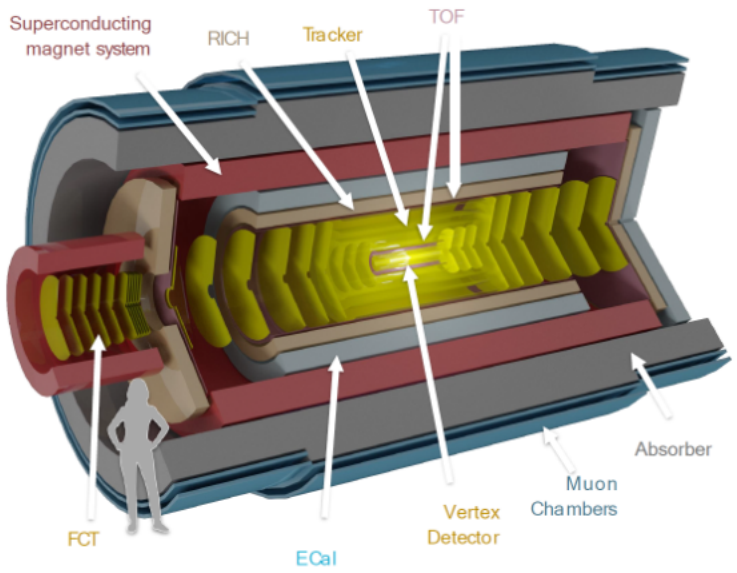


Figure 3: Proposed layout for the ALICE3 experiment.

The proposed detector that would allow for this incredible science feed is the ALICE 3 detector that would be installed during LHC Long Shutdown 3, and operational during LHC Run 5[13]. The main focus of the new design is an incredible pointing resolution $\sim 2\mu\text{m}/p[\text{GeV}/c]$, that is essential to reconstruct secondary vertices and decay chains, allowing for heavy-flavor identification. To achieve the required pointing resolution, the first detection layer must be as close to $\sim$ 5 mm at the top energy. A complete redesign of the detector makes it a compact detector with a full silicon tracking device with eleven barrel layers at its core. The inner three layers are the vertex detector with pseudo-rapidity acceptance of $|\eta| < 4$ and radially extending from 80 to 0.5 cm. To acquire the pointing resolution that is needed for the experiment, the inner core of the detector needs to be as close as possible to the collision inner barrel. The required positron resolution has to be 2.5 μm with a radiation length of 0.1% X/X_0. The Time-of-Flight detector consists of two cylindrical layers with a time resolution of 20 ps. The outer layer at 85 cm radial distance from the event point will make the separation of electrons from hadrons possible up to $P_T = 500\text{MeV}/c$ and π/K separation up to GeV/c. The inner layer will then account for the lower range of p_T.
Outside of the last TOF layer is a cylindrical Ring Imaging Cherenkov (RICH) detector, allowing e/π separation up to 2 GeV/c distinguishing protons from e, π and K up to 14 GeV/c. Outside of the RICH detector is the next layer of detector, an electromagnetic calorimeter enabling the measurement of photon-jet correlations and radiative decays of charmed mesons.
These barrel detectors are surrounded by a large superconducting magnet that generates the required magnetic field to curve the particles and measure the momenta from these track curvatures. The field produced by this magnet is $B = 2\text{T}$. For the detection of muons, a cylindrical steel absorber is $70cm$ thick surrounds the magnet with two layers of muon detectors. Lastly, a Forward Conversion Tracker (FCT) is placed in alignment with the barrel to measure photons down to tranverse momenta of 2 MeV/c with photon energies down to E_γ 50 MeV/c at pseudorapitity of $\eta = 4$.
The detector is proposed for 2035 but there are huge R&D challenges to be tackled for this futuristic detector to be realized. The radius of the innermost Vertex Tracker layer is 5 mm and is less than the transverse radius of the LHC beam during the injection phase of 10 mm. Therefore, the first 3 inner tracker layers have to be placed in

vacuum and have to be able to move in and out of position for measurement. During the injection phase, the layers must retract to a minimal safe position of 15 mm from the nominal interaction point. When the beam is stable, the layers have to move to the final position with the first layer at a distance of 5 mm from the nominal interaction point.
The position resolution that is desired of the order of 3-4 μ m will need a pixel pitch smaller than 10 μm. The area of the outside tracking layer is 67m^2, requiring the development of cost-effective sensors, with a low material budget of 1% X/X_0 per layer with strong but low-weight support structures and cabling. For the TOF system, the time resolution should be < 20 ps, implying that sensors and microelectronics must be significantly improved. If these challenges are overcome, the rate of data coming from the sensors will be on a scale that has not yet been seen before in a physics experiment. This implies the next challenge of coming up with a way to manage the data from the detector and reduce and compress it so that the analysis and also the storage are possible.

2.4 Data challenge

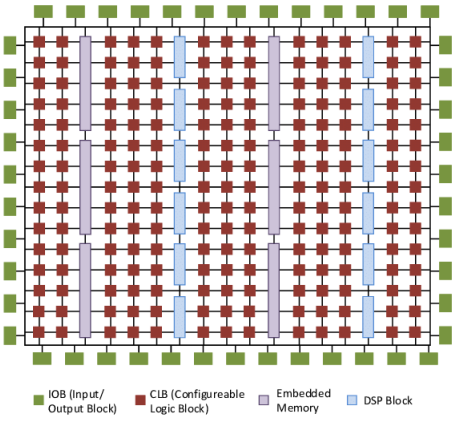
The successful ALICE Run 2 that is now finished hosted Pb-Pb collisions at an interaction rate of 10kHz, being limited by both the readout rate of the TPC $<$ 3.5 kHz and the bandwidth of the online data acquisition, the experiment only accounting for 80 GB/s.
Run 4 challenges
Alice Run 3 and 4 will account for 100x more Pb-Pb collisions at peak rates of 50 kHz[14]. This contributes to a total of 3.5 TB/s of data that requires extremely large data bandwidth to handle. The total amount of data, when compressed, that will be produced during Run 4 totaling 60 PB has to be compressed by a factor of 35 to be able to store the data. After only 7 months of data-taking of Run 3, a total of 70PB is already saved on the disk, which will increase until Long Shutdown 3 in December 2025. The current formatting of the data strategy would have to be drastically optimized. For Run 4 it requires a large reduction in data and a significant increase in data bandwidth due to the improvements of the sensors and the overall detector operating at higher interaction rates for the heavy ion collisions.
Data processing scheme
During the experiment, the data taken from the detector equipment are processed synchronously, keeping up with the bandwidth of the sensors. By chopping up the total data into 128-256 data packets called sub-Time-Frames containing data from only a subset of the experiment, the task at hand is divided into subtasks. Before the data goes onto the second processing level from Event reconstruction in the Event Processing Nodes (EPN), the data is reduced by a factor of < 6 to a total of 0.6 TB/s. In the EPN the chopped up bits of data are merged from the subframes to form one event. After this synchronous reconstruction, calibration and data compression results in Compressed Time Frames (CTFs) that are stored in a 60 PB disk buffer with a data rate of < 0.1 TB/s. Each processing step must continue to run synchronously with the collision rate of the experiment, preventing a bottleneck from forming that could potentially lose data from the experiment. CTFs mark the beginning of the asynchronous data process. CTFs are reduced data packets compressed by a factor of 35 to fit on the disk for event analysis on the large O^2 farm. In order to speed up the computations, heterogeneous computing is used with GPU computer card accelerator that is up to 30 times faster than a conventional CPU.
ALICE 3 challenges
If ALICE 3 where to be successful experimentally it would require statistics that are not seen before even compared to Run 4 which is already pushing the limits of what is possible with computational power. For ALICE 3 the total data through-output coming from the sensors of all the trackers combined is 20 TB/s for heavy-ion collisions. Then it is assumed that in the two-stage processing scheme, during the online processing phase, the data compression should be 35 fold. To estimate the computing power required for the GPU-based ITS tracker of ALICE 3, the requirements of the TPC tracker of run 3 are extrapolated since the current ITS tracker does not house comparable numbers to the proposed ALICE 3 ITS tracker. The TPC tracker requirements of run 3 are scaled linearly with the number of hits increase, and a scaling for the relative performance difference between the TPC and ITS tracker.
For ALICE run 3 the synchronous event reconstruction was benchmarked for 2000 AMD MI50 GPUs, which would account for an estimated total cost of 2MCF for the accelerator boards. This benchmark was only for the TPC detector that has the most computing intense reconstruction requirement. The GPUs are distributed over a total of 250 farms, each GPU requiring 300 W[15] totaling a power consumption of 1.20 Mwatt/h or the equivalent of the power consumption of one thousand households. The ITS tracking system is not yet complete compared to the ALICE 3 proposed tracker, it lacks reconstruction of looping tracks with low transverse momentum. Therefore, an additional 20% tracking time is added. It is then estimated that a total of 2000 GPUs are required from the year 2030, which are expected to be 4 times more powerful in the future than the ones available currently. Should this trend for the increase in computing power hold and for the amount of GPUs to be sufficient, there is still a memory

issue. During data taking, the amount of data produced per month for pp collisions is 180 PB/month, or about 38 times more than for Run 3. This requires an event-selection strategy to be implemented to reduce the data volume by two orders of magnitude. This skimming strategy is not yet developed.

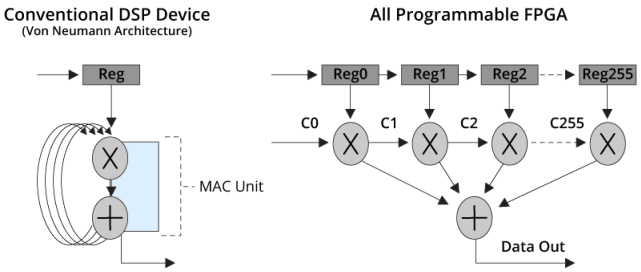
2.5 FPGA solution

The large data load and data processing challenges faced by the ALICE experiment are currently addressed with heterogeneous computing and hardware triggers. In heterogeneous computing, a combination of CPU and GPU computing units is used to utilize the advantages of both systems. For ALICE Run 3, GPU accelerator cards and hardware triggers are used to handle the data load at the synchronous phase where traditional CPUs are not able to cope with the huge data rates. Accelerator cards are good at parallel processing, performing arithmetic operations simultaneously. This allows for acceleration when the same workload is being performed many times in succession, as is the case for processing events in the ALICE experiment. Hardware triggers are used to reconstruct and compress events, significantly reducing the data load. The ALICE high-level trigger can handle event rates of 6 kHz at 49 GB/s for pp collisions[16].

One suitable type of hardware for heterogeneous computing and hardware triggers are Field-Programmable Gate Arrays(FPGAs). FPGAs are integrated circuits that provide a unique combination of flexibility and performance. Unlike GPUs, the circuitry and arrangement of the blocks can be tailored to meet specific computational requirements. Therefore, the chip allows for more flexible operations compared to other computing architectures. FPGAs consist of an array of configurable logic blocks and interconnections. The logic block on the FPGA can be configured for the implementation of simple logic functions such as AND and OR gates, but also more complex functions such as multipliers and memory units. The general layout of the FPGA architecture is shown in figure 4a, where the configureable blocks are displayed. By connecting the blocks together with the configureable interconnects, a complex system for the desired task can be programmed. The Digital Signal Processing (DSP) blocks are specialised to leverage the parallelism of the architecture taking data from multiple channels at once, as can be seen in 4b.



(a)



(b)

Figure 4: Typical Xilinx FPGA architecture with components labeled on the left, image taken from .[17]. Difference between conventional Neumann and FPGA architecture shown, image taken from [18].

The FPGA operates at high performance and low latency due to the highly optimized architecture for specific tasks, combined with parallel computation, allowing it to be faster than a general-purpose CPU or GPU. Low latency and high performance are desired for the ALICE experiment, where readout times are in the microsecond range and data rates are on the order of terabytes.
FPGAs generally have low power consumption, and the designer can fine-tune the hardware to meet the power efficiency requirements. By using portions of the fabric for designated tasks accommodating multiple functions, the energy efficiency is higher, as parts of the chip are left unused, thus conserving power usage. Due to the long product life and high flexibility of the FPGA, it is possible to reuse the chip for multiple purposes, thus removing the need for additional computing units. This is an excellent way to reduce the number of computing units required and cut the costs of the equipment. The tasks can be performed completely on the FPGA, reducing any communication overhead. Unlike the GPU that requires an additional CPU to manage to GPU, the FPGA can be designed

to not require the use of additional computing units. The tasks can be processed in parallel, executing multiple operations in parallel within one clock cycle. This is due to the array of programmable logic that resides on the chip rather than a sequence. Processing of a large amount of data in parallel is possible. Programmable FPGA logic allows parallel computations to occur in one clock cycle. The computations do not have to be in 2nd power bit fashion but can be any integer bit structure. This results in the possibility of low-latency applications to Run on the FPGA. In this thesis the possibility of using a neural network on the FPGA logic is explored, to hopefully in the end comply with the data requirements of the new Alice detector. The challenge with FPGA computing is the specialization required in order to program the fabric. Debugging and optimization is more complex than traditional programming on CPU and GPU.

2.6 Why neural networks

The reconstruction and event selection algorithms employed in the ALICE experiment leverage machine learning techniques, a branch of artificial intelligence. These algorithms are trained on simulated or previously analyzed data, to identify distinctive patterns and features in the incoming data that indicate noteworthy physics phenomena. Certain phenomena and infrequent events prove challenging to identify through traditional rule-based approaches, primarily due to the inherent complexity of the problem. Machine learning techniques excel at discerning underlying complex patterns and correlations from extensive datasets.
The algorithms in use can be approximated with ML solutions to minimize latency and maximize precision of the tasks, for example, in the case of jet tagging[19]. General purpose Processing units, such as CPUs and GPUs, have been predominantly used for the training of NN models. Especially large uncompressed NN models for which arithmetic operations are represented in floating-point accuracy. The training of these larger models occurs on GPUs. However with the use of DNN approximations it has offered the possibility to exploit custom hardware platforms, such as the FPGA and ASICs, with fixed-point quantization. This results in faster inference with possibly small accuracy reduction. The accuracy of ML models and techniques is largely dependent on the size and complexity of deep learning models. Larger models increase the amount of computations and increase latency, which is not desirable in the readout of the detectors. Several detectors (TRD, CPV, HMPID, EMCAL, DCAL, PHOS) require a fast trigger to comply with the high interaction rate (50 KhZ for Pb-Pb) of the ALICE experiment. Additionally, the hardware used for the trigger is constrained by the limited space in the cavern and, therefore, has limited computational capacity. The cooling equipment similarly has limited space, restricting the performance of the hardware. Meeting the computational demands of the experiment with resource-intensive models, while keeping latency and resources at a minimum, proves challenging. The improvement of the efficiency of the algorithm and its inference has improved significantly in recent years. Advancements in more compact network design[20], weight and filter pruning[21], and or quantization[22][23][24] to name a few. In post-training quantization, trained models are transformed into lower-precision equivalents, by definition losing model performance. Quantization-Aware training[25] is proposed to mitigate this issue, adopting a quantized representation of the model and incorporating constraints during weight optimization in the training process. Resulting in a model that is more robust for quantization and minimal loss in performance. By pruning the weights and filters, reducing the amounts of nodes, the model is compressed, making it more compact and less resource intensive. DNN pruning induces performance degradation, which again can be minimized by pruning the network during training rather than post-training. This minimizes tuning of parameters such as sparsity ratios.

A collaboration between CERN and Google was able to deploy a neural network on FPGA fabric with high accuracy and extremely fast inference[26]. The fully connected NN model was able to distinguish heavy-quark flavors at 70% accuracy and latency in the order of ns. By quantizing and pruning the model, it was possible to make an ultracompressed neural network detection trigger that meets the requirements of the CERN experiment. The QKeras model was deployed on the largest and most powerful FPGA hardware manufactured by AMD, the Virtex ultrascale+ VU9P, which is an acceleration card. The card is typically used for high-density data-center applications with high memory bandwidth, and has a clock frequency of 200 MHz. The QKeras model was synthesized using hlsm4l, a general library first published in 2018, to convert ML models into FPGA firmware[27]. The deployment of CNNs using hlsm4l was first made possible in 2021[28].
The deployment of CNNs on FPGAs can be achieved using several different software frameworks. A 2018 toolflow survey [29] provides a comparative analysis of hardware architectures and design software combinations. Since then the Xilinx Vitis-ai software framework, and the hls4ml library have been added to the list of possibilities.

3 Chapter 1 Preliminary study

The aim of this section is to validate a conceptual framework employing a basic but efficient neural network executed on the FPGA. This preliminary study examines the workflow from network training to deployment in the target architecture, as well as considering several networks and analyzing the consequential impact on latency and accuracy. This section will show certain limitations and potential enhancements of the Vitis AI software module. These insights will prove useful for the subsequent section of this thesis, which will focus more on the deployment of models on the FPGA in the context of experimental physics applications. The workflow that will be implemented is visualized in Figure 5. The framework side will consist of the training of several NN models, which then will be quantized and compiled on the host side using Vitis AI, after which it will be ready for deployment on the target side for testing of inference and accuracy. The Convolutional NN model trains on the MNIST dataset using Pytorch, a

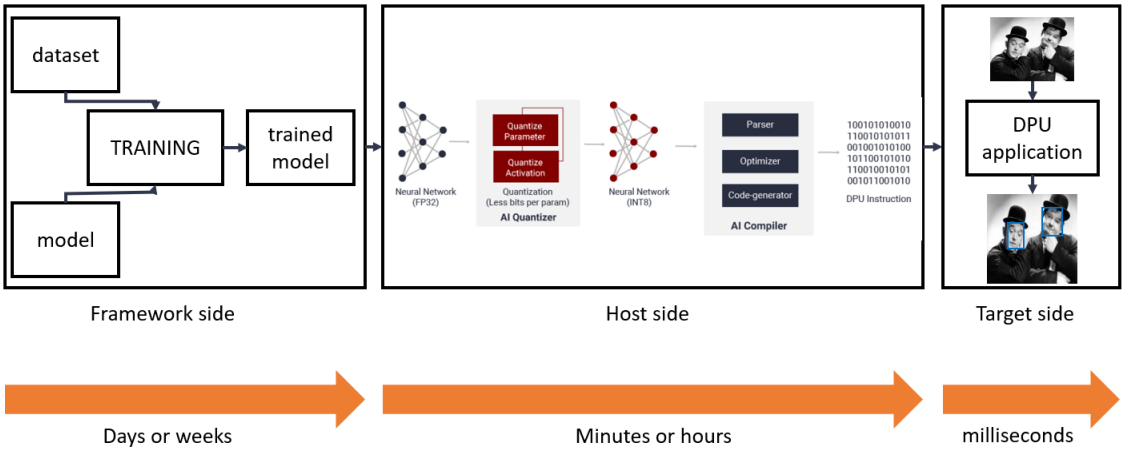


Figure 5: Workflow of deployment of NN on FPGA using Vitis-ai. Figure taken from [30]

machine learning library for Python. The trained model will then be quantized and compiled using Vitis-AI for the FPGA. Vitis-AI is an AMD Xilinx software framework, specialized in the development of ai inference applications for AMD accelerator boards. The Vitis-AI compiler transforms the NN into a model that is understandable by the FPGA architecture and can be deployed on the target hardware. The model is deployed and tested on the Kria kv260 development board, a Xilinx development platform to speed up inference testing of vision applications such as image classification.

3.1 MNIST dataset

The MNIST dataset is considered a fundamental benchmark for the testing of neural networks and, consequently, the tool pipeline. The MNIST handwritten digit database encompasses a collection of 70,000 individual grayscale images, each measuring 28x28 pixels, depicting handwritten numerical digits[31]. Each image is rescaled and centered to keep the required pre-processing to a minimum, putting the focus towards the classification problem. Identifying and classifying examples from the MINST database will serve as a good example of FPGA capabilities in vision applications. To illustrate the classification problem, a sample is taken from the MNIST data set and displayed in Figure 6. After preprocessing the images, the models training, evaluation, and quantization calibration can take place. For training, the dataset is split into 60 thousand images for training and 10 thousand images for testing. For the classification of the digits, a Convolutional Neural Network (CNN) will be used, as Le Cun et al. used in 1989[32]. After the training phase, the latency and accuracy of the models is measured for the varying amount of layers and trainable parameters of the model.
This dataset will serve as the first test for the study of neural network inference on the FPGA chip. Training on this problem will be relatively straightforward and fast, even on a local computing system. This will allow the focus of the study to be more on deploying the NN on the FPGA.

3.2 Neural networks

This section briefly gives an overview of the neural network used. In theory, linear regression basis functions can be sufficient to classify and recognize complicated patterns in data. In practice, these models are limited in applicabil-

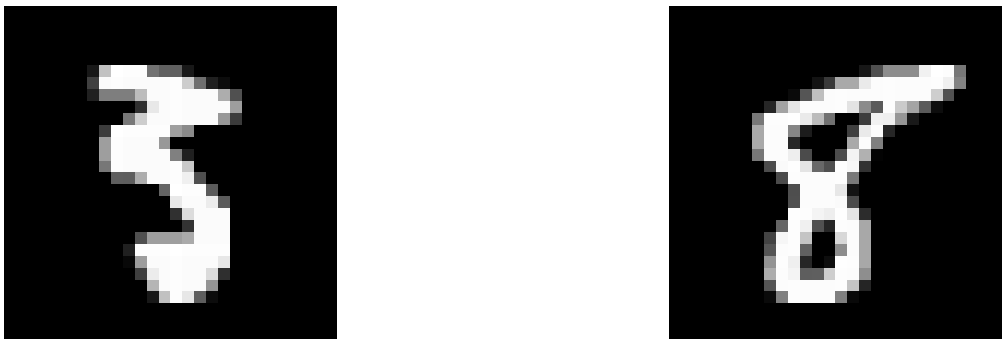


Figure 6: Handwritten 3 and 8 respectively in grayscale from the MNIST dataset.

ity due to the curse of dimensionality. This refers to the exponential increase in the volume of space that needs to be sampled as additional dimensions are added to the model. Consequently, the limitations imposed by the curse of dimensionality restrict the applicability of linear regression basis functions to handle high-dimensional data sets. In the context of two-dimensional images, each pixel in the image can be considered as a dimension.

- ***Convolutional Neural Networks***

The recognition of handwritten digits is accomplished by a Convolutional Neural Network (CNN), which is particularly useful for image recognition tasks. CNNs are made up of several layers of neurons with connections between the respective layers. The layers between the input and output layers are commonly referred to as hidden layers, connecting small regions of the input image with receptive fields in the layers. The convolutional layers in the model consist of a set of filters, the so-called kernels, which contain the training parameters. These filters extract the local features of the image that are later merged to detect higher-order features to ultimately extract the information about the image as a whole.
Given an unknown function $f : X \longrightarrow Y$ that best describes the data, the neural network decomposes the unknown function to multiple simpler functions, $f_i \circ f_j \circ ...f_M$ representing the M layers of the network. Each convolutional layer receives an image $A^{(m-1)}$ as input from the previous layer and outputs A^m for the m channels. The output of the layer is known as a feature map and is computed as

$$A^{(m)} = g_m \sum_k W_k^{(m)} \star A_k^{(m-1)} + b^{(m)}. \tag{1}$$

with b biases and W weights. The weights W or parameters use a spatial filter to parameterize features of the image, the weights are tailored by supervised learning of the network.

- ***Relu Activation***

The Rectified Linear Unit activation function (ReLU) is an activation function that is placed after the convolutional layers. This activation function is given by the positive part of the argument, in this case, the input to a neuron:

$$f(x) = x^+ = \left\{ \begin{array}{ll} x & \text{if } x \neq 0, \\ 0 & \text{otherwise.} \end{array} \right\} \tag{2}$$

The ReLU activation function was first introduced in 1969 by K. Fukushima[33] and is in place to enable improved neural network training. Activation functions in neural networks are crucial as they introduce nonlinearity to the output of a neuron. This non-linearity helps to capture and represent more complex patterns in the input data. For instance, the ReLU activation function outputs the input directly if it is positive, otherwise it outputs zero. During training, the network adjusts the weights and biases of each neuron in response to the input data and the error gradient calculated through backpropagation. As the network learns, certain neurons may activate (or not activate) in response to specific patterns in the input data. For instance, in an image recognition task, certain neurons may activate when the network encounters specific characteristics or features in the image, such as loops in handwritten digits. This activation and deactivation process is an outcome of the iterative learning process.

- ***Batch Normalization***

Batch Normalization was first introduced by Loffe and Szegedy in 2015[34] to make deep neural network training stable and faster. By normalizing the output of the layers for each batch, it prevents a phenomenon called *internal covariate shift*, which is the change of a layer's input distribution due to the adjustment of the parameters of previous layers. This process significantly slows training and requires lower learning rates. With the use of batch normalization, this shifting of input distributions is resolved. This normalization step is performed after each ReLU activation function of the convolutional layers.

- ***Soft Max activation***

To obtain a posterior distribution over the given set of digits, the output of the last layer is assigned to a soft max layer. This gives the output of the model and therefore the prediction of the network. The soft max activation function $\sigma(z)_i$, in the case of 10 classes, for class i is defined as:

$$\sigma(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{10} e^{z_j}}. \tag{3}$$

- ***Adam optimizer***

The optimization of the network occurs during training using the Adam optimizer algorithm [35]. Adam is an efficient stochastic optimization algorithm, requiring only first-order gradients. The method adapts the gradient estimates for different parameters and works with sparse gradients. In the context of optimizing a noisy scalar function $f(\theta)$, where θ represents the parameter, the objective is to minimize the expected error, indicated as $\mathbb{E}[f(\theta)]$, while ensuring the differentiability of f with respect to θ. The gradient of the function is denoted as $g_t = \nabla_\theta f_t \theta$ in the timestep t. The algorithm updates the exponential moving averages of the gradient m_t and the squared gradient v_t. The exponential decay rates of these moving averages are controlled by the hyperparameters $\beta_1, \beta_2 \in [0, 1)$. Updating occurs according to the rules:

$$\begin{aligned} m_t &= \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t, \\ v_t &= \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2. \end{aligned} \tag{4}$$

The moving averages are estimates of the mean and un-centered variance of the gradient. By initializing the moving averages at 0, the estimates are biased towards zero. Bias-corrected estimates $\hat{m}_t$ and $\hat{v}_t$ are computed therefore as:

$$\begin{aligned} \hat{m}_t &= \frac{m_t}{1 - \beta_1^t}, \\ \hat{v}_t &= \frac{m_t}{1 - \beta_2^t}. \end{aligned} \tag{5}$$

The updated parameter θ_t is then computed as:

$$\theta_t = \theta_{t-1} - \alpha \cdot \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}. \tag{6}$$

The iterative process of updating the parameters continues until the convergence of θ_t is achieved.

3.2.1 The Model

Three models undergo training and testing to evaluate their inference and performance. By systematically adjusting the number of nodes and layers in the models, a relationship between their performance and computation time is established. The models considered consist of three, four or five layers with 512, 16384, and 524288 trainable parameters, respectively. Each layer is made up of a combination of a convolutional layer followed by a batch normalization layer and a ReLu activation layer. The ReLu activation layer of the last layer is replaced with a flattening layer. Lastly, the Soft Max activation function maps the model output to the posterior distribution of the digits. For the convolutional layers, a kernel size is chosen with [5,5,2] for three layers, [5,5,4,2] for four layers, and [5,5,2,3] for the model of five layers. The size of the kernel indicates the length and height of the square kernel. To provide a visualization of the neural network, Figure 7 illustrates the architecture of the three-layer CNN.

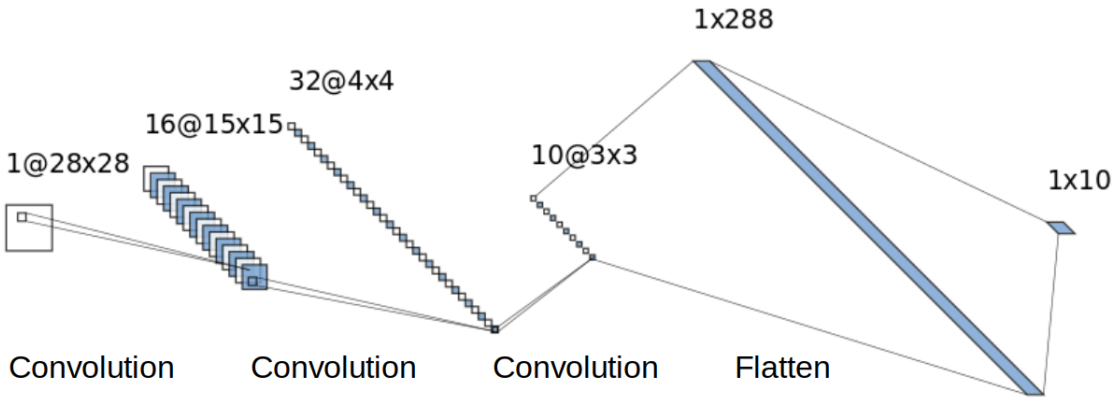


Figure 7: Illustration of the architecture of the three layer CNN model. $b@a \times a$ indicates b feature maps with sizes $a \times a$. After the flatten layer a Softmax layer activates the outputs to the posterior distribution corresponding to the 1×10 classes. The Figure is generated using NN-SVG[36].

3.2.2 Training

Training and testing of the model is performed on an Intel Core™ i7-8565U CPU with a clock speed of 1.80 GHz and 8 cores. The PyTorch model is trained for three epochs in the entire training data set using the CPU, employing a learning rate of 0.001. To mitigate the risk of overfitting and prevent the model from memorizing the image sequence, the training dataset is randomly shuffled before each epoch. After every training epoch, the model is validated on the validation data set, which is kept separate from the training set. The results of the training process are shown in Figure 8b. Model weights are uniformly initialized before training, as seen in 8a. The accuracy of the initial model without training is 10%, because, as expected, every output of the model is equally likely. When the model is trained over three epochs, the weights are redistributed relatively quickly, and the model can learn the underlying patterns of the data. After training, the weight distribution shows resemblance to a Gaussian distribution with zero mean, as seen in Figure 8c.

3.2.3 Testing

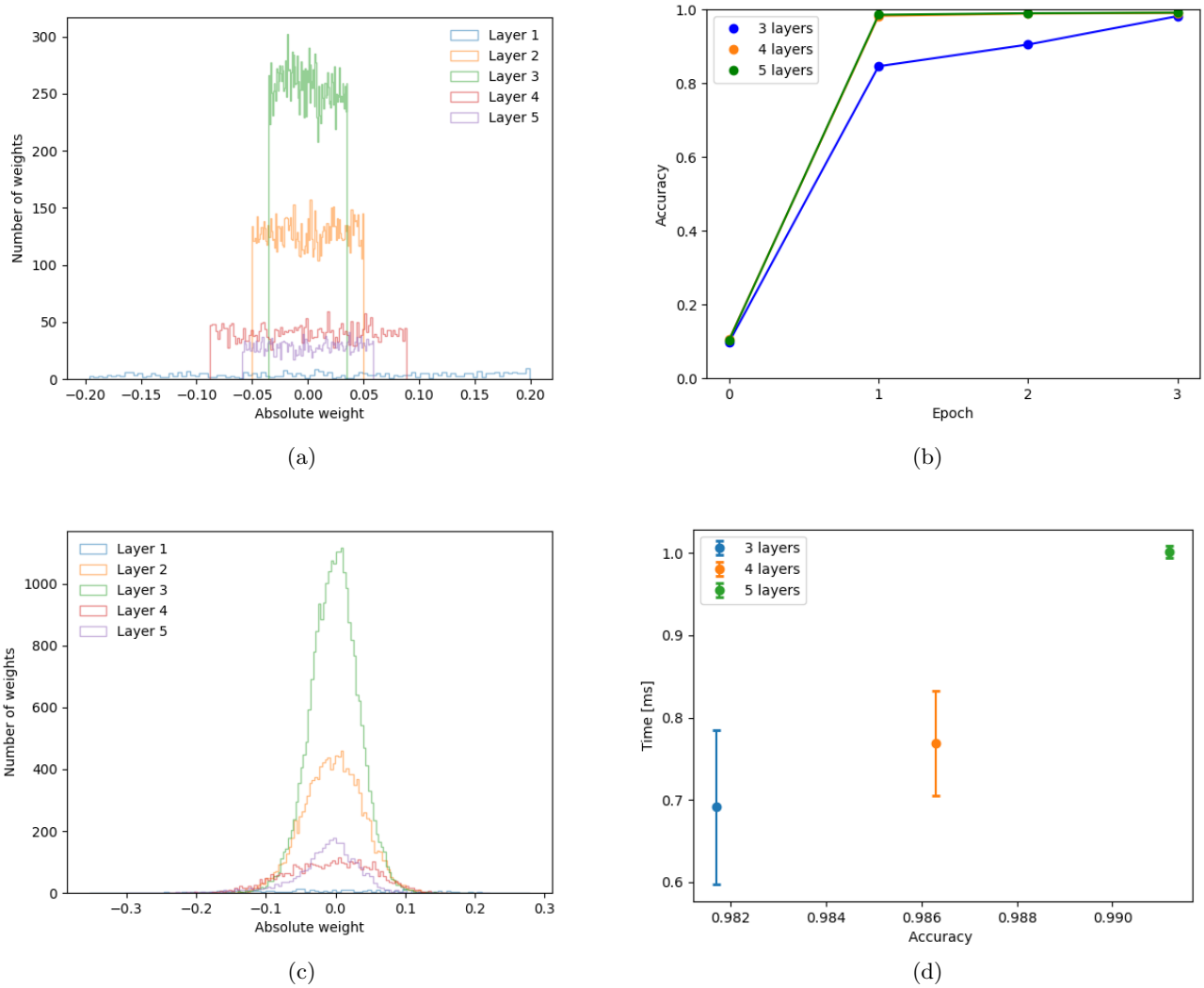
After training the models for three, four, and five layers, the inference of the model is measured on the CPU. The timing of the model is measured in inference mode over a total of 100 iterations of each image in the testing data set.

Code block for the inference measurement:

```
import torch.utils.benchmark as benchmark

@torch.inference_mode()

def run_inference(model: nn.Module,
    data: torch.Tensor) -> torch.Tensor:
    return model.forward(data)


t0 = benchmark.Timer(stmt="run_inference(model, data)",
                     setup="from __main__ import run_inference",
                     globals={"model": model, "data": data},
                     num_threads=1,
                     label="Latency Measurement",
                     sub_label="torch.utils.benchmark.")
```

The code running under the mode of *@torch.inference_mode()* ensures no gradient is calculated and backward propagation of the model does not take place. Inference mode improves the model's performance and allows for the comparison of the inference measurements with the FPGA. The testing code is run on a single thread with batchsize

1, again to make the testing compareable to the FPGA which does not allow for multithreading. If the FPGA is placed inside the detector for trigger applications, data processing is desired on an one-event basis to prevent data buffers. Therefore, the images are classified one by one with a batch size of one to replicate the trigger application for event processing.



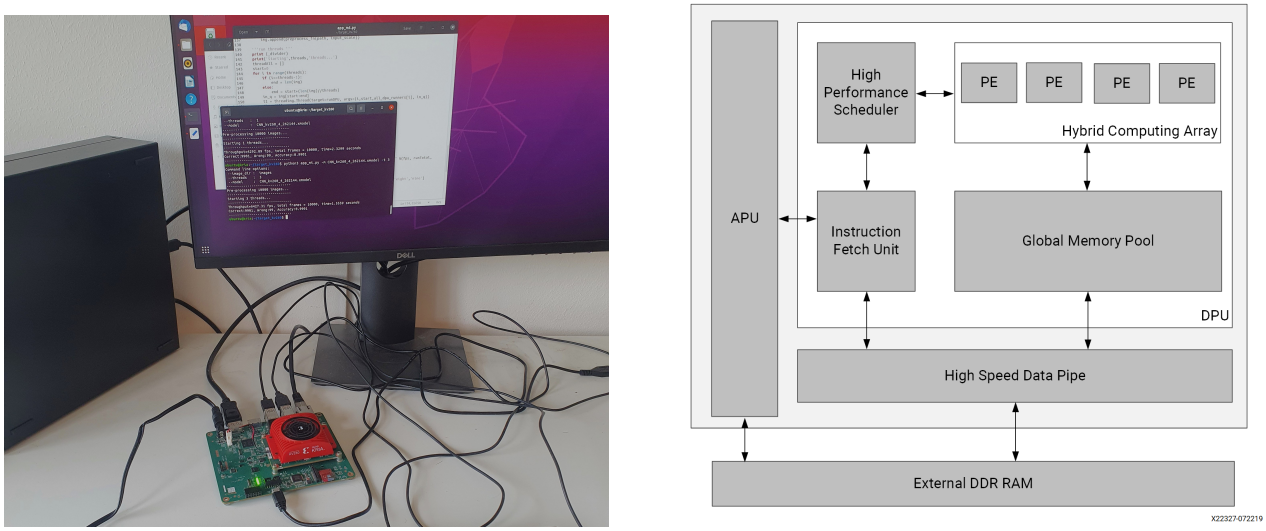
(a) (b)

(c) (d)

Figure 8: Training of the CNN on CPU hardware. Weights are uniformly initialized in the model before training, as displayed for the 5-layer CNN in 8a. The training progress of the models is displayed in 8b. The weights distribution after the training process result in the Figure 8c for the 5-layer CNN. The Latency of the model on the CPU was measured and is displayed in Figure 8d.

The increased amount of layers improves the accuracy of the model but increases the latency of the model as seen in Figure 8d. Similarly, the amount of weights of the model impacts the model's latency, due to the increased amount of computations. This is due to the CPU architecture, which does not have effective parallelism to compute it in one clock cycle. Every calculation requires one additional clock cycle for the CPU. The best performing model has 5 layers and achieves an accuracy of 99.12%, with a latency of 1.001 ± 0.007 ms per image for the test data set. This result will be used as a comparison to the performance of the FPGA architecture.

3.3 Kria kv260



(a) (b)

Figure 9: (Left) Experimental setup with the Kria kv260 FPGA board connected to an external monitor for testing purposes. (Right) Top-level layout of the DPU core consisting of the distinct components Application Processing Unit (APU), Processing Engine (PE) and Deep Learning Processing Unit (DPU). Figure from [37]

The NN is tested for inference on the Zynq Ultrascale+ MPSoc XCK26 FPGA chip. This chip runs on the Kria k26 SOM as part of the Kria kv260 Ai vision starter kit by Xilinx[38]. It is a low end FPGA application device to get started with creating FGPA AI applications. The chip has a total of 256K programmeable logic cells, 1248 DSPs, and 26.6 Mb on-chip memory. The Kria SoM k26 contains a DPU core, specifically DPUCZDX8G. Kria k26 can be configured with different configurations for the deep learning processing unit (DPU). The DPU configuration B4096 at 300 MHz offers a peak performance of 1.4 TOPS, and B3196 at 300 MHz offers a peak performance of 0.94 TOPS. The power consumption of the B4096 configuration is benchmarked at 10 watts.

DPUCZDX8G Architecture	LUT	Register	DSP	Block Ram	Channel Parallelism	Peak Operations
B3196	46714	79710	566	208	14	3136
B4096	52161	98249	710	255	16	4096

Table 1: Resources of the different DPU configurations.

For this example, the B3196 DPU configuration is used, since that was the configuration with the highest-peak operations available at the time. Subsequently, the support for the B4096 configuration for the kria kv260 was made available[39], which is used later in the study. To demonstrate the capabilities of the FPGA architecture, the model is loaded onto the kv260 for inference testing. The software to deploy the model on the target device is Vitis-AI developed by Xilinx. The version of Vitis-ai used for this particular test is Vitis-AI version 2.0.

3.3.1 DPU

The DPU is Xilinx's Intellectual Property (IP), that is, a reusable logic unit designed specifically to accelerate DNN models. The internal architecture of the DPU is illustrated in Figure 9b, which consists of four main components: the scheduler module, Processing Engines (PE), instruction fetch unit block, and global memory pool module. The Application Processing Unit (APU) is an ARM processor responsible for running the application, handling data transfer from and to the DPU, and handling any interrupts. The instruction fetch unit reads and executes DPU instructions that are associated with the CNN models. These instructions for the DPU are then passed on to the high-performance scheduler to manage the transfer of instructions between the PEs and the memory. The PEs execute the DPU instructions with input data from the Global Memory Pool which serves as

a buffer for input and output data. Weight, bias, and intermediate feature maps are stored in on-chip memory consisting of block RAM. The DPU can be utilized by converting CNN models from Tensorflow or Pytorch into a compatible format through the Vitis AI framework. Vitis AI can facilitate model quantization, pruning, and model compilation into DPU-supported instructions. One can generate a configuration file *arch.json* in the Vivado or Vitis suite that is read by the Vitis AI compiler. This configuration file contains the instructions and options for the configuration of the DPU core. For this example, the standard configuration file provided is used with the fingerprint of the model architecture *0x1000020f6014406*, and can be found in the folder of the Vitis-AI software */opt/vitis_ai/compiler/arch/DPUCZDX8G/KV260/arch.json*. The layout and placement of the computations on the DPU is determined by this configuration file. To make a custom architecture configuration, the Vitis unified platform can be used. In this study, the precompiled configuration is used.

3.3.2 Quantization

The resources of the FPGA are limited compared to traditional CPU and GPU units; therefore, hardware implementations must be highly efficient. To achieve this, the floating-point accuracy weights and biases are converted to lower precision fixed-bit precision. This significantly reduces the use of hardware memory and computations, but can reduce the predictive accuracy of the model. To deploy the model on the Kria kv260, the weight and biases are quantized from 32 bit floating point accuracy to 8 bit accuracy. The reduction of abundant bits of the weights is at a minimal loss of overal accuracy of the model. The impact of weight rounding, resulting in loss of precision, becomes evident in the overall accuracy of the model as a result of the cumulative effect of altered computations. The magnitude of this effect varies depending on the robustness of the model. Notably, in the case of larger models consisting of more weights, the error introduced by weight rounding tends to be relatively smaller compared to smaller models with a limited number of weights. The technique of quantization-aware training can be used to maintain the accuracy of the model after quantization. The strategy that is used for post-training quantization as implemented by the Vitis-Ai software is the power-of-2 scale quantization. The quantization strategy as mentioned in [22] is introduced in the section below. Considering a NN with L layers and W_i parameters, during supervised learning the loss function that is optimized for is given as:

$$L(\theta) = \frac{1}{N}\sum_{i=1}^{N} l(x_i, y_i; \theta), \tag{7}$$

for input data with its label (x, y) and the loss function $l(x, y; \theta)$. During post-training quantization the parameters that are stored in floating point precision are quantized after the training period. The mapping function that is the quantization operator $Q(r)$ is described as:

$$Q(r) = Int(r/S) - Z, \tag{8}$$

with r the real-valued input, S the real valued scaling factor, and Z an integer zero point. The *Int* function maps a real value to an integer value through a rounding operation of choice. Rounding operations are the cause of loss of accuracy. The method employed quantizes the parameters in an uniform manner, each parameter in real space is uniformly mapped to a quantized space. On the contrary, a nonuniform quantization strategy can be employed to more effectively represent specific datasets by selectively allocating more precision to regions characterized by a high-density data distribution. The choice of the real valued scaling factor S used for the calculation of the quantization operator $Q(r)$ is given by:

$$S = \frac{\beta - \alpha}{2^b - 1}, \tag{9}$$

where α, β is the clipping range and b the quantization bit width. Determining the clipping range is referred to as calibration, and a common choice is that of the minimum and maximum values in the data set. However, this is an asymmetric quantization scheme because the minimum is not necessarily symmetric with the maximum $\alpha \neq \beta$. The symmetric quantization scheme can be chosen by, for example, taking the maximum absolute value of the signal and setting $-\alpha = \beta$. The argument for a symmetric clipping range is the choice of the zero point then being $Z = 0$ in the quantization function. This results in the option to zero out the zero point in the model, ultimately lowering the amount of computations. Non-uniform quantization involves quantization steps and quantization levels. The real values are mapped to a quantized step:

$$Q(r) = X_i, \text{ if } r \in [\Delta_i, \Delta_{i+1}). \tag{10}$$

That is, the value of a real number r falls in-between the quantization step given, it is set to the quantization level X_i. This quantization scheme can achieve higher accuracy for fixed bit width due to the possibility of calibrating

for the important value regions in the dataset. In the case of power-of-2 scale quantization scaling, the quantization levels increase exponentially instead of linearly, thus being a nonuniform quantization strategy. The choice for the power-of-2 scale is motivated first, because logaritmic representation can encode data with large range in fewer bits than fixed-point representation. Second, the log domain is native to the encoding that digital hardware uses. Lastly, it is shown that logaritmic quantization obtains higher classification accuracies than linear quantization[24]. The motivation for the choice of the 2-base logaritmic scale becomes clear when considering the operation for the multiplication that is used for the convolutional layers. The operation in the quantized form becomes a simple bitshift as shown in the equation:

$$\begin{aligned} w^T x &\approx \sum_{i=1}^{n} w_i \times 2^{x_i} \\ &= \sum_{i=1}^{n} \text{Bitshift}(w_i, x_i), \end{aligned} \tag{11}$$

where $x_i = \text{Quantize}(\log_2(x_i))$. The operation Quantize$(x)$, quantizes x from floating precision to an integer, and the Bitshift(a, b) operation shifts a value by b bits. This simplifies the computational requirements of the target device, because computationally expensive multipliers are redundant. The quantization of a model post-training is implemented using Vitis-Ai in Tensorflow with the following block of code:

```
from tensorflow_model_optimization.quantization.keras import vitis_quantize

quantizer = vitis_quantize.VitisQuantizer(model, quantize_strategy='pof2s')
quantized_model = quantizer.quantize_model(calib_dataset=calib_dataset,
                                           calib_step=100,
                                           calib_batch_size=10,
                                           **kwargs)
```

A range of quantization strategies can be used by changing the parameter *quantize_strategy*. A list of options is *pof2s*(default), *pof2s_tqt*, *fs*, *fsx*, and also the option to implement a custom quantization strategy by *quantizer.set_quantize_strategy(new_quantize_strategy='custom_strategy.json')*. The default power-of-2 scale quantization strategy is used for this example.

3.4 Results

Table 2: Models performance for different model sizes.

Model Size	Accuracy Float (%)	Accuracy Quantized (%)	Parameters	Latency CPU (μs)	Latency FPGA (μs)	Latency FPGA 3 threads (μs)
3 Layers	98.18	98.04	16,080	691.22±94.22	140.94	113.92
4 Layers	99.05	99.01	25,296	768.91±64.21	145.18	110.12
5 Layers	99.12	99.09	45,776	1000.14±7.03	154.61	100.06

Table 2 presents the performance metrics for different model sizes in terms of accuracy (both for the float and quantized model), the number of model parameters, and latency on both the CPU and FPGA platforms. In terms of model size, measured by the number of parameters, the 3-layer model was the smallest with 16,080 parameters, while the 5-layer model was the largest with 45,776 parameters. The 4-layer model had an intermediate size with 25,296 parameters. In terms of accuracy, all three models performed remarkably well. The 3-layer float model achieved an accuracy of 98.18%. As we increased the complexity of the model, a noticeable improvement in accuracy was observed. The 4-layer model achieved an accuracy of 99.05%, while the 5-layer model exhibited the best performance, scoring an accuracy of 99.12%. After post-training quantization, the model performance dropped only slightly for all models. The 3-layer model achieved accuracy of 98. 04%, the 4-layer model 99.01% and the 5-layer 99.09%. That is a decrease in the accuracy of only 0.03% to 0.14%, which results in 14 more images being misclassified due to the quantization of the model. As for the latency, the measurements were taken on a CPU and an FPGA platform, with the latter measured both with single thread and with 3 threads for the CPU control part of the FPGA. The latency of the CPU was the highest for the 5-layer model at 1000.14μs with a standard deviation

of 7.03μs. The 3-layer model and the 4-layer model had latencies of 691.22μs ($\pm$94.22μs) and 768.91μs ($\pm$64.21μs), respectively.
Despite the linear architecture of the CPU, the latency did not increase linearly with the amount of trainable parameters. For a 2.8 increase of parameters, with a completely linear system one would expect a latency of about $\approx 1900\mu$s for the 5 layer model. This is far from the measured latency of 1000.14 ± 7.03 on the CPU for the largest model. Some of the latency of the models is not from calculating the matrix multiplications with the parameters, there is some bias time involved with the framework of the model prediction. The latency on the FPGA platform was lower compared to the CPU across all models. The 5-layer model, despite being the most complex, showed the highest latency of 154.61μs, which is still significantly lower than its CPU counterpart. The 3-layer model showed the lowest latency of 140.94μs, followed closely by the 4-layer model with a latency of 145.18μs. When the number of threads of the Kria SOM increased to three, that is, the CPU control part of the FPGA, the latencies decreased further for all models. The 5-layer model latency was reduced to 100.06μs, which was the lowest among the models, followed by the 4-layer model with 110.12μs and the 3-layer model with 113.92μs. The decrease of latency for larger models in the multi-threaded case is not explained easily. As graphically shown in Figure 10, the FPGA significantly outperforms the CPU, the FPGA was 10 times faster than the CPU for the largest 5-layer model.

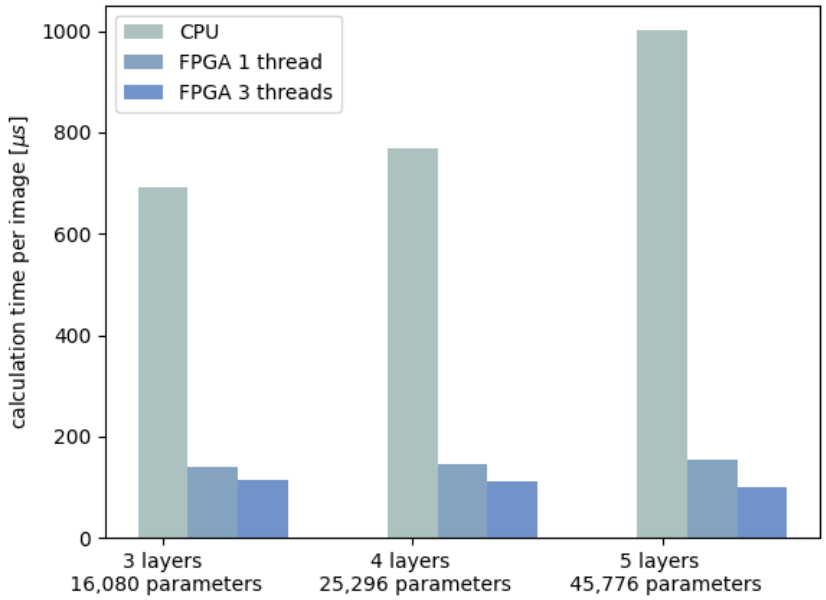


Figure 10: Comparing FPGA and CPU latency of CNN models for the MNIST dataset.

3.5 Conclusion

The comparative analysis of the models latency between the CPU and FPGA signifies a clear advantage for the FPGA. As the model architecture is scaled by adding more layers, and the amount of trainable parameters increases, 16,080 to 25,296 and 45,776, the calculation time of the CPU increases from $691 \pm 94.22\mu$s to $1000.14 \pm 7.03\mu$s. However, in case of the FPGA the latency remains relatively steady, independent of the number of layers in the model, from 140.94μs to 154.61μs. This distinction can be primarily attributed to the FPGA's capability to distribute the computations over the chips architecture, making use of the parallel computations of the architecture. The CPU's linear computational framework clearly increases the latency with the number of computations of the model. Since the model used is not optimized for the FPGA architecture, an improvement in the models performance is to be expected after optimization. This would further improve the relative performance compared to the CPU. For this relatively simple study, however, it is shown that it is possible to deploy a CNN on the FPGA hardware leveraging the parallel architecture of the chip to outperform the CPU. In conclusion, the FPGA's chip architecture allows for parallel computation of the parameters in a convolutional layer, with the largest model still able to fit in one clock cycle of the FPGA. For even larger models, the latency of the FPGA will likely increase due to the limited

space of the DPU. For later studies, it is interesting to quantify this result and see how much of the operations are actually used by the model and how much of the model's parallelism is utilized. The decrease in latency by the FPGA for this small model shows that it is feasible to deploy neural networks on the FPGA to speed up computations. The full potential of the FPGA has not yet been fully capitalized, but will be explored further in the next chapter.

4 Chapter 2 Particle Shower identification

4.1 Introduction to particle showers

The calorimeter is a fundamental sub-system of the ALICE detector for studying the behavior of subatomic particles. The calorimeter measures the energy deposited by particles as they pass through the detector layers, allowing the identification of particles through their energy and tracks. As the particles pass through the layers, parts of their energy are absorbed, and a cascade of secondary particles is again produced, detected by the sensors. When high-energy particles collide during a Pb-Pb event inside the particle accelerator, a large amount of energy is released in the form of new particles, leaving a cascade of energy deposits inside the calorimeter detector. These so-called particle showers (also referred to as jets) in the calorimeter can be used to characterize quark-gluon plasma on the final stage of the data analysis and modeling. Current particle shower identification algorithms rely on low-level taggers, a mixture of simple manually optimized algorithms, and machine learning models. In this study, a different approach is used, a single trained neural network without the use of low-level taggers as a trigger. First, a convolutional neural network is used to distinguish different particle showers, with the use of the projection of the pixel hits in the calorimeter. After the binary classification of particle showers, the network performance is evaluated, and the missclassifications are studied to find out what makes the structure difficult to classify. Lastly, overlapping particle showers are classified up to close separations between the centers, to test the potential resolution of the calorimeter.

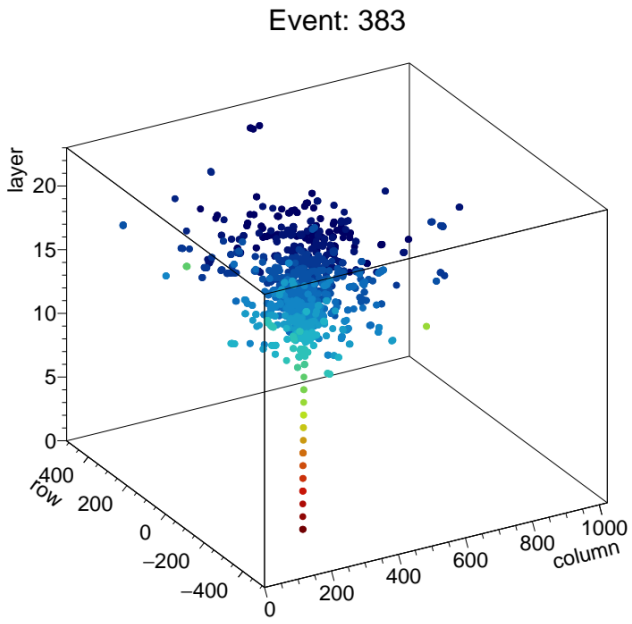


Figure 11: Simulation of a $\pi^{\pm}$ shower in a calorimeter detector. The particle traveled from in the positive layer axis direction leaving a trail of hits in the detector as it cascades.

4.2 Theory of particle showers in detectors

Electrons that move through a medium, lose energy through the ionization of the material. This leaves a trail of ionized atoms liberated e^- in the medium. In tracking detectors, this trail of ionization is detected with a common detector material of doped silicon wafers. e^--hole pairs are created in the ionization process which, when a potential difference is applied across the silicon, drift in the direction of the electric field. At the edge of the wafer there is a p-n junction that can collect the charge, which is the signal that is then amplified, giving a pixel accuracy signal.

Photon and electron interactions in matter

Electrons can interact and lose energy in various ways when moving through a medium. At low energies e^- and e^+ mainly lose energy by ionization. There are other processes contributing as well, (Møller scatering, Bhabha

scattering, and e^+ scattering), but they are relatively small. The ionization loss rate increases logarithmically with energy. Above a critical energy (few tens of MeV for most materials), the energy loss is dominated by the Bremsstrahlung[40] which rises nearly linearly. Charged particles moving through matter interact with the e^- and atoms in the material, exciting and ionizing the atoms. This results in an energy loss of the traveling charged particle. The Bethe-Block formula with corrections best describes this process.

$$-\langle\frac{dE}{dx}\rangle = 2\pi N_a r_e^2 m_e c^2 \rho \frac{Zz^2}{A\beta^2}\left[\ln\left(\frac{2m_e c^2\beta^2\gamma^2 E_{\max}}{I^2}\right) - 2\beta^2 - \delta - \frac{C}{Z}\right], \tag{12}$$

with E_{max} the maximum energy transfer to an e^- in a single collision,

$$E_{max} = \frac{2m_e c^2\kappa^2}{1 + 2s\sqrt{1+\kappa^2} + s^2} \tag{13}$$

where, $\kappa = \beta\gamma; s = m_e/M$. I is the mean ionization potential of the atom. The shell correction is denoted as C, δ is the density correction. For particles at low energy, the corrections are large and become smaller at higher energy particles.

The stopping power differs for e^- and e^+, and both differ from heavy particles due to the kinematics, spin, charge, and identity of the incident e^-. For e^- the Møller cross section describes large energy transfers to atomic e^-. e^--e^+ scattering is described by the Bhabha cross section. High-energy e^- predominantly lose energy in matter through bremsstrahlung, while high-energy photons lose energy mainly by e^+e^--pair production. The mean distance over which a high-energy e^- loses all but $1/e$ of its energy is called the radiation length X_0 given by:

$$\frac{1}{X_0} = 4\alpha r_e^2 \frac{N_A}{A}\{Z^2[L_{rad} - f(Z)] + ZL'_{rad}\}, \tag{14}$$

where $4\alpha r_e^2 \frac{N_A}{A} = (716.408\text{g cm}^{-2})^{-1}$. L_{rad} and L'_{rad} are tabulated parameters [41]. L_{rad} is the radiation logarithm for the material, while L'_{rad} is the radiation logarithm for the inelastic form factor. The function $f(Z)$ is an infinite sum but can be approximated for elements up to uranium as:

$$f(Z) = a^2[(1+a^2)^{-1} + 0.20206 - 0.0369a^2 + 0.0083a^4 - 0.002a^6], \tag{15}$$

where $a = \alpha Z$.

Hadron interactions in matter

The electronic interactions of charged particles heavier than e^- are now discussed. Fast charged particles electronically interact with matter, leading to ionization and atomic or collective excitation. Most often the energy losses are small, with 90% of the collisions resulting in energy losses of less than 100 eV. This infers that in thin absorbers only a few collisions will take place with a large variance in the energy loss. The Rutherford differential cross section describes the scattering of heavy particles from free e^-[42]:

$$\frac{d\sigma_R(W;\beta)}{dW} = \frac{2\pi r_e^2 m_e c^2 z^2}{\beta^2}\frac{(1-\beta^2 W/W_{max})}{W^2}, \tag{16}$$

where W_{max} is the maximum energy transfer in a single collision. The maximum energy transfer in a single collision for a particle with mass M is given as:

$$W_{max} = \frac{2m_e c^2\kappa^2}{1 + 2\gamma s + s^2}. \tag{17}$$

In matter e^- are not free, therefore a more complicated differential cross section was obtained by Bethe using Born theorie[43].

$$\frac{d\sigma_R(W;\beta)}{dW} = \frac{d\sigma_R(W;\beta)}{dW}B(W) \tag{18}$$

with a correction factor $B(W)$ that accounts for electronic binding interactions.

The average energy loss rate for relativistic charged heavy particles in the energy region $0.1 \leq \beta\gamma \leq 1000$ is best described by the Bethe equation12, where E_{max} is then replaced by W_{max}.

Electromagnetic cascade

When a high-energy e^- or photon strikes a dense absorber, it triggers an electromagnetic cascade. This cascade is set off by pair production and bremsstrahlung, which in turn produces more e^- and photons, albeit with reduced energy. The longitudinal development is dominated by the high-energy part of the cascade particle. This is related

to the radiation length of the incident particle until the critical energy of the e^- energies is reached, at which point it is not able to generate more shower particles. The mean longitudinal profile of the energy deposition in an electromagnetic cascade is reasonably well described by a gamma distribution:

$$\frac{dE}{dt} = E_0 b \frac{(bt)^{a-1} e^{-bt}}{\Gamma(a)}, \tag{19}$$

with the scaling variables $t = x/X_0$ and $y = E/E_c$ to measure the distance in units of radiation lengths, and energies in units of critical energy. This can also be seen in Figure 13, where a gamma function is fitted to the fractional energy deposition from an e^--induced cascade EGS4 simulation. The transverse development of electromagnetic showers in different materials scales fairly accurately with the Molière radius R_M which is defined as 90% of the showers energy:

$$R_M = X_0 E_s / E_c \tag{20}$$

where $E_s \approx 21 MeV$ and the Rossi definition of E_c is used.

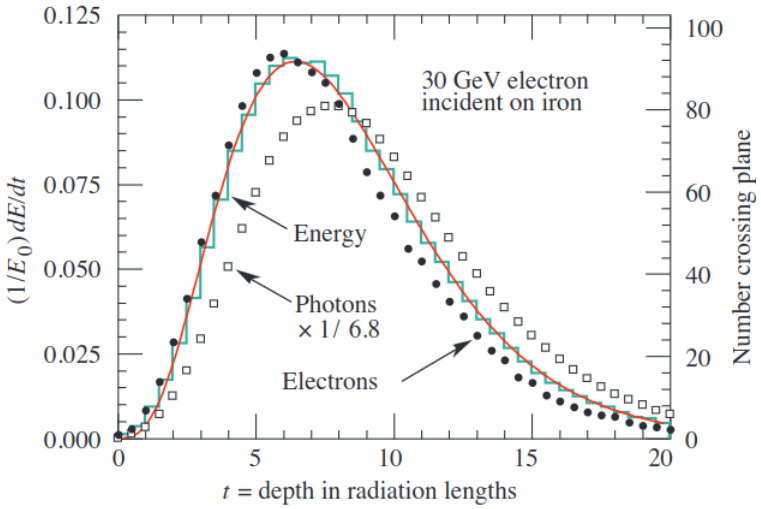


Figure 12

Figure 13: EGS4 simulation of 30GeV e^--induced cascade of iron. The histogram shows fractional energy deposition per radiation length, with the gamma-function fit to the distribution. Histogram taken from particle data group review 2022[44].

4.3 Distinguishing e^+ and $\pi^\pm$ events

4.3.1 Epical-2 prototype detector

This study investigates the applications of neural networks on FPGAs as a hardware trigger for the ALICE forward electromagnetic and hadronic calorimeter (FoCal). The FoCal is a proposed electromagnetic calorimeter combined with a sampling hadronic calorimeter used for photon isolation and jet measurements. The FoCal covers pseudorapidities of $3.4 < \eta < 5.8$ and is an upgrade to the ALICE experiment that is to be installed during Long Shutdown 3 (LS3) for the LHC run 3. It provides unique capabilities to measure small-x gluon distributions via prompt photon production and will enhance the resolution of the ALICE experiment.
A potential candidate for the FoCal detector is the Epical-2 prototype that will be used for this study of a NN FPFA trigger. The 25x10^6 pixel detector consists of 24 layers of ALPIDE CMOS MAPS sensors. Each ALPIDE chip has 1024x512 pixels of size 28x28μm that measure the electromagnetic shower response, energy resolution and linearity at a frequency of 40Mhz. For the evalation of the prototype, a simulation model using GEANT4 and Allpix2[45] is compared with the stochastic energy resolution. The simulations represent the detector prototype well and show that the resolution is comparable to state-of-the-art resolution for Si-W calorimeters.

4.3.2 Simulated Data Sample

In this section we briefly describe the simulation data of the Epical-2 detector that will be used for the classification of particle showers with the use of neural network models on the FPGA. The data is produced with a Monte Carlo simulation of the Epical-2 detector response using Allpix2[45], a pixel detector simulation framework based on GEANT4[46] and ROOT.

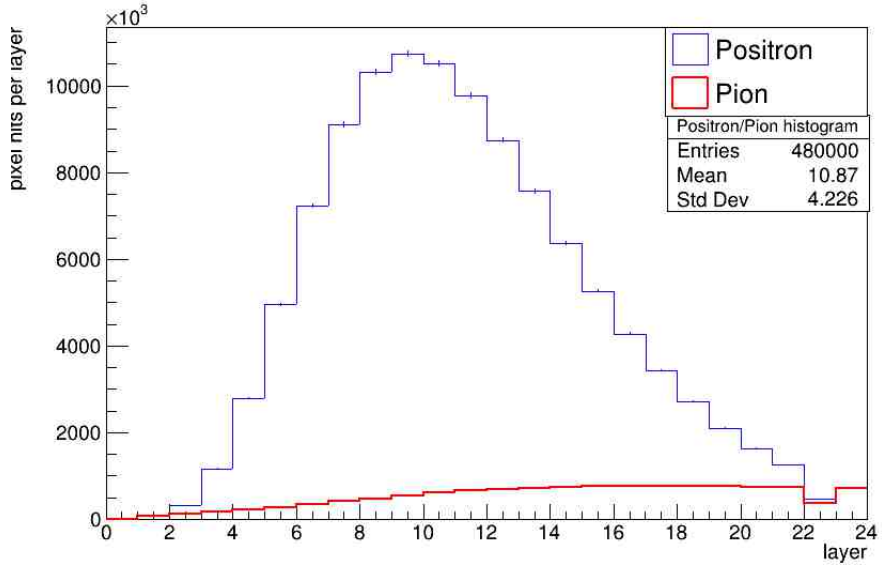


Figure 14: Histogram showing the distribution of particle hits per layer for $\pi^{\pm}$ and e^{+} events in the Epical-2 prototype detector Monte-Carlo simulation.

The simulation framework GEANT4 has several lists of physics models to choose from for the simulation. The most accurate simulation of electromagnetic physics requires the FTFP_BERT_EMZ[47] configuration and is used to produce the data for this study. The name of the physics model, FTFP_BERT_EMZ, follows a systematic convention. 'FTF' represents the Fritiof string model, which is applicable for energies above 5 GeV for high-energy hadronic interactions. The 'P' in the name signifies the incorporation of the G4Precompound model, employed for the de-excitation process. 'BERT' corresponds to the Bertini cascade model[48], utilized for energies below 10 GeV for lower-energy hadronic interactions. Finally, 'EMZ' is the designated option for the most precise electromagnetic physics, including single scattering physics and the multiple scattering Goudsmit-Saunderson model[49].
The Fritiof model is used in high-energy hadron collisions. It describes the hadronic interactions for particles whose energy is sufficiently high to probe the inner structure of the hadrons. In this energy region, it is necessary to consider the structure of the hadrons and the behavior of the individual quarks. The conceptualization of the interactions is the collision between strings, which are field configurations between quarks and antiquarks, rather than collisions between point particles. Upon collision the high-energy hadron can be excited into a string state, subsequently disintegrating into new hadrons which the particle detector detects. In the Fritiof model both the initial excitation of the string and the subsequent breakup into new particles are included. Together with the Fritiof model, the pre-compound model is used to simulate the de-excitation of an excited nucleus after a high energy interaction. Excitation of nucleus can occur due to the ejection of one or more nucleons after a high-energy collision. The process by which this excess energy is released is simulated by the pre-compound model, by emission of nucleons or light nuclear fragments. Each type of emission is probabilistically calculated on the basis of the energy and angular momentum of the excited nucleus, together with the binding energies of the possible emitted particles.
The Bertini cascade model simulates the interaction of high-energy particles with a nucleus, where one or more nucleons are knocked out of the nucleus. This creates a cascade of secondary particles as the secondary particles each potentially interact with other nucleons, creating further secondary particles. The cascade continues until all the particles have either escaped from the nucleus or have been absorbed. In this Bertini cascade model, the motion of nucleons inside the nucleus, as well as the potential for $\pi^{\pm}$ production and the effects of nuclear binding energy are taken into account.

The multiple scattering of e^- is simulated with the Goudsmit-Saunderson model. The model takes into account not only the average deviation of the particle's path but also the distribution of deviations around the average. For large deflections, especially, this distribution is non-Gaussian. This model assumes that the individual scattering events cause only small deviations of a particle's path, the small-angle theory. Several parameters can be chosen in the simulation, of which the choice of the discrete stepsize, which affects the mean energy loss during a step. The actual energy loss is then calculated as a summation of all the models with an added fluctuation using the GLANDZ model.
The simulation data consist of the set of pixels that are hit in an event, which are stored as layer, column, and row per hit. A 3D reconstruction is then made, resulting in a matrix of size 24x1024x1024 for all pixels. By doing so, the particle shower are visualized as in Figure 11. The dataset totals 20k $\pi^{\pm}$ and 20k e^+ simulated events. To make the data feasible for machine learning, a projection onto a 2D plane of choice can be made, creating an image that can be processed quickly by the neural network. By quantizing the image to a lower size, the image resolution is lowered, but the processing speed is increased. Quantization of the original image to a new image of varying sizes is considered, ranging from 256x256 down to 16x16. The processing of the image by the model is drastically faster for smaller image sizes. This decrease in latency comes at the price of decreased accuracy as a result of the decrease in resolution. The effect the quantization has on the speed and accuracy is extensively studied.

4.3.3 Analysis

The neural networks performance is analyzed using TMVA[50] in combination with ROOT. The performance of the model is measured for signal efficiencies of the model at low background efficiency. In the context of machine learning, background efficiency refers to the ability of the model to accurately identify and account for background events in the detection of particles. When distinguishing between two or more types of particles, particle type A is treated as signal events, and all the other types as background events.
TMVA calculates the signal efficiency at a given background efficiency (like B=0.01) on the basis of the Receiver Operating Characteristic ROC curve. The ROC curve is a graph that illustrates the performance of the classification model at different classification thresholds. When plotting the True positive rate against the False positive rate for several cutoff points, the ROC can be plotted. The trade-off between true and false positive rate is thus represented by the curve. To find the signal efficiency at a given background efficiency of B=0.01, the point of the ROC curve that corresponds to the given background efficiency is taken. The resulting signal efficiency is then the true positive rate for a given false positive rate.
The performance of a particle classifier is determined by the number of false classifications by the model. A wrong prediction about the event can occur if the model confuses it with another particle. This is best displayed with the use of a confusion matrix. A confusion matrix for multiple classes shows a metric of how often a certain class is mistaken for another class. Distinguishing a pattern from the confusion matrix allows for insight into the power of the model and its failures. TMVA calculates a confusion matrix by considering the signal efficiency of class A compared to all the other classes, at a given background efficiency on the diagonal, and the class A signal compared to the class B background on the off-diagonal.

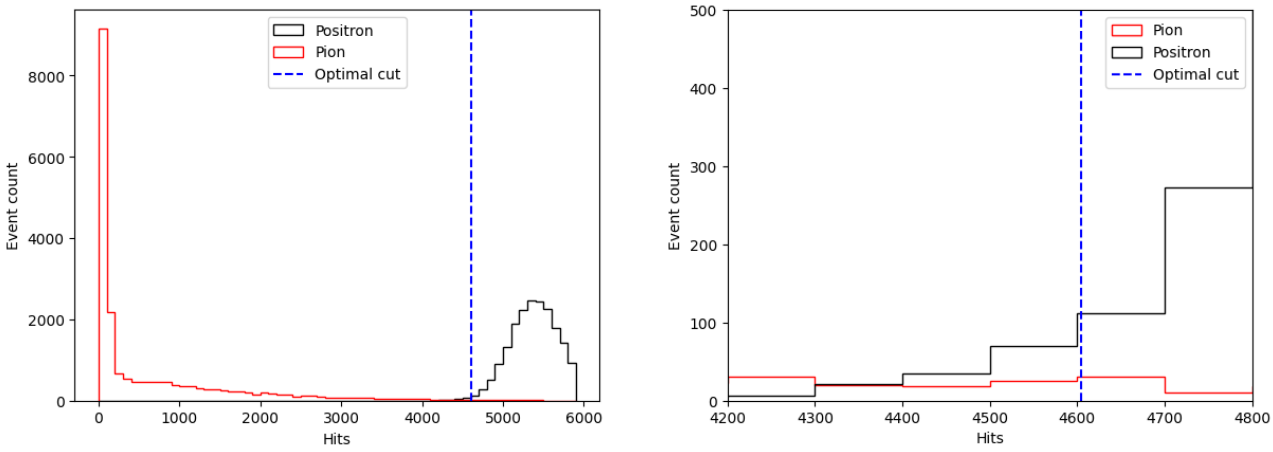
4.4 Binary Classification

The first investigation is on the distinguishing of the two classes of showers that are $\pi^{\pm}$ or e^+ showers. From the distribution of hits per layer in Figure 14 a clear distinction can be made between $\pi^{\pm}$ and e^+ showers in the detector. The e^+ jets decay early on in the detector every time due to the larger cross section of the smaller momentum of the particle. Therefore, the particle interacts much more likely with the silicon detection pads in the detector, which causes the particle to decay into an electromagnetic particle shower. This produces a larger total amount of hits for the first several layers of the detector. The hadronic $\pi^{\pm}$ has a larger momentum with a smaller cross section than the e^+ resulting in a lower chance of decaying within the length of the detector. This results in a lower total number of hits in the detector, and the first layers detect relatively fewer hits. The distinction between the particles becomes even more clear when looking at separate particle events in the detector. e^+ generally start to shower in the first few layers of the detector, whereas $\pi^{\pm}$ do not shower at all, leaving just a single track of hits through the layers. Other about 20% of the $\pi^{\pm}$ do produce a shower in the detector, but at varying layers. Some start showering at the end of the detector after traveling through most of the layers, while some shower very early, similar to e^+. Most of the events can be separated with great precision just by looking at the 3d image of the event as displayed in 11, however, the problem arises when the $\pi^{\pm}$ showers early on in the detector. This produces a shower in the detector that looks very similar to that of the e^+. Distinguishing $\pi^{\pm}$ from e^+ that show a similar decay profile is important for the analysis of the ALICE experiment. The missing or incorrect classification of $\pi^{\pm}$

will result in higher uncertainties in the analysis later on. Therefore, we require to classify the particles with high precision in all cases. The remainder of this section is focused on separating $\pi^{\pm}$ from e^{+} events at high precision. The $\pi^{\pm}$ is referred to as a signal, and the e^{+} as a background.

Cut value

The most simple classification is first tested by using a simple cutoff value on the total amount of pixel hits for one



(a) (b)

Figure 15: (Left) Total hits in in Epical-2 detector in simulated $\pi^{\pm}$ and e^{+} shower event. Optimal cutoff value displayed to distinguish the two classes based on total hits. (Right) Zoomed in axis range to display the overlap between the two classes at the optimal cut value.

event. The e^{+} that reach the detector start an electromagnetic cascade from the first layer, creating on average 5602 hits in the detector. The $\pi^{\pm}$ are less likely to cascade early on in the detector, and on average create less hits in the detector with an average amount of hits of 657. The simplest cutoff value to make is based on the total amount of hits. The best possible cutoff value of 4604 has an accuracy of 99.55% accuracy for binary classification. This simple first-order calculation results in an extremely fast trigger. However, it has 136 false negative events, where a $\pi^{\pm}$ cascade got miss-classified as e^{+} event. These are the early cascade $\pi^{\pm}$ that results in irreducible contamination of the e^{+} sample once this simple algorithm is used. By making a first rough cut at 3000 total hits in the detector, 48.05% of the total data set is correctly classified as $\pi^{\pm}$. This leaves 820 $\pi^{\pm}$ events and 19999 e^{+} events, which is roughly 96% e^{+} events. Only one outlier e^{+} is lost in this case. To accurately classify the remaining events, we need to look more carefully at the structure of the particle showers to be able to identify early showering $\pi^{\pm}$. This requires a machine learning algorithm to accurately distinguish $\pi^{\pm}$ from e^{+}.

XY projection

The detector consists of 25x10^6 pixels in total, which produces a 3d that produces a large amount of computations if it were fed through a convolutional neural network. Therefore, we reduce the dimensionality of the problem by projecting the data onto the xy-plane. This reduces the 24 layers to 1 layer, reducing the amount of computations exponentially. Information on the structure of the particle event in the detector is now partially lost at the expense of computation time. To further reduce the size of the event image, the image is mapped to a smaller image size with fewer pixels. To quantify the effect of this quantization step, we vary and test for several quantization steps. The input image is mapped to $n \times n$ pixels by summing the $1024/n$ neighboring pixels in both the x and y directions. For example, one pixel in the 128x128 mapped image of the 1024x1024 original image, consists of the summation of a 8x8 kernel. Now the total data points of the event is reduced to the order of 1e4, which can be handled by a convolutional neural network. The effects of the choice of the xy plane and the mapping size are investigated by comparing different mappings and comparing to an xz-plane projection.

XZ projection

The information that is lost in the xy plane projection, namely the structure of the development of the particle shower over the layer, can be seen in the xz plane projection. Specifically, at what layer the electromagnetic cascade starts in the detector, can be determined precisely in this plane. This allows one to classify the particles accurately due to the characteristic decay of the $\pi^{\pm}$. The events that look very similar to each other in the xy-plane are the e^{+} events start to shower in the early layers of the detector and have over 4000 total hits in the detector. As shown in Figure 16, both the showers of $\pi^{\pm}$ and e^{+} can show very similar patterns in the detector. This makes it

difficult to distinguish the particles. By projecting the data onto the xz plane, the total data points of one event is reduced from $L * R * C$ to $L * C$ which, for quantization down to 128 columns amounts to just $24 * 128 = 3072$ datapoints. With the number of layers fixed, the number of total data points after the mapping scales linearly with the quantization size instead of quadratic as with the xy-plane projection.
The xz-plane projection case is not considered for deployment in the FPGA and, therefore, is not quantized. This reason for this is that the jet-tagging of double particle jets will not be generalizable in this case. The xz-plane projection loses the information of the relative distance between the jets unless the angle of both centers of the jets and the z-direction is 90 degrees. For the other angles, the relative distance of the jets in the projection is not determinable, unless the center of both jets positions is also known.

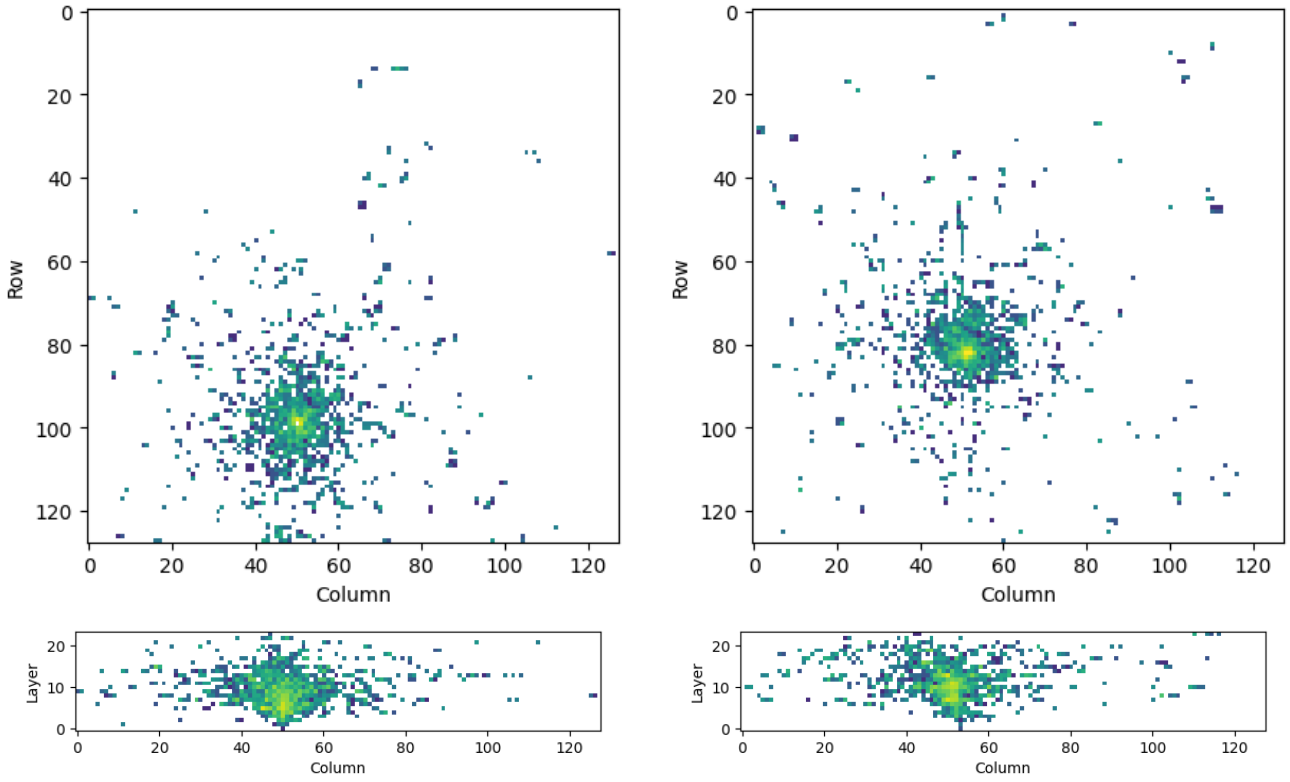


Figure 16: (left) e^+ and (right) $\pi^\pm$ cascade events for more than 3000 total hits in detector. The XY-projection is displayed on the top row. The XZ-projection is displayed on the bottom row.

4.4.1 Model

The model used for the binary classification, is a convolutional neural networks with three convolutional layers. Latency and accuracy are determined by the size of the model, the amount of weights, connections and layers. The design of the model is selected with the preliminary study on the FPGA in mind with a hard requirement for the accuracy to be greater than the simple cut off value. A Keras model in Tensorflow2 is the architecture of choice due to the support by the Vitis AI quantizer package and the compilation tool, at the time of the study. Vitis AI is constantly improving and developing the software with support for features being added and removed constantly. The Tensorflow2 software seemed to be the most reliable option, due to its extended period of support, and the additional options for quantization strategies. The vitis-quantizer module supports, but not limited to, the conv2d, maxpooling2d, ReLU activation, reshape, flatten and dense layers for deployment on the FPGA fabric. The first reshape layer introduces minimal latency but is required to transform the single bitstream of data coming from the detector, to a 2d image that can be handled by the convolutional network. After each convolutional layer, followed by a ReLU activation, a maxpooling2d layer is introduced that would be fused with the conv2d layer according to the documentation. This we will see later will introduce additional CPU computation time and should therefore be left out for faster inference. The final softmax activation function is performed as post-processing on the CPU to determine the classification outcome. The documentation mentions the softmax layer can run on the standalone

Softmax IP for acceleration, however the compiler puts it on the CPU. The weights of the model ar initialized with a truncated normal distribution. The model layout is displayed in Figure 17 and the total amount of trainable parameters per model configuration can be found in the tables with the respective results of the models.

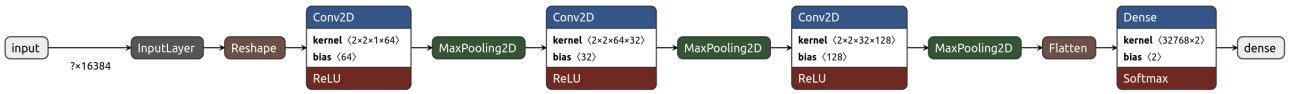


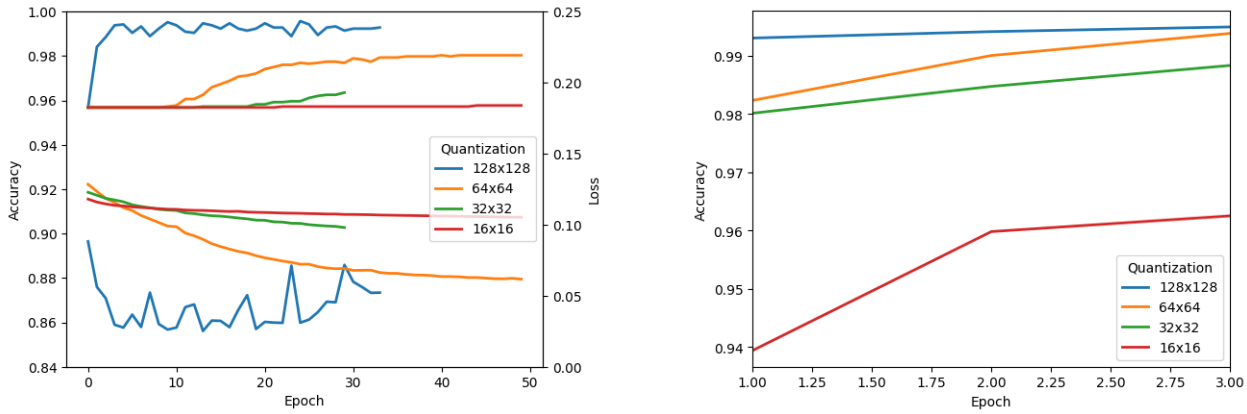
Figure 17: Convolutional neural network for input of 64x64 mapped pixels for binary classification.

4.4.2 Training

The model is trained for up to 50 epochs with early stopping of 20 epochs if no improvements in the validation accuracy is seen. After each epoch the dataset is shuffled, to prevent over fitting of the sequence of input images. Optimizing the weights and biases of the trainable parameters are updated with the Adam optimizer algorithm [35]. The Adam algorithm is better at generalizing performance is most cases over the conventional SGD algorithm, and is often proposed as the most efficient stochastic optimization. The learning rate of the optimizer is set to 0.001, with a weights decay of 0.5 for a stable training process. The weights decay rate, the rate at which the weights are decaying to zero if no update is performed after each epoch, is set to 0.1. The weights decay minimizes the over fitting of the model, at the cost of making the model less power full. To speed up the training process, the training occurs in batches of 100 input images. Finally the Binary cross entropy loss function is used for the training process. The models is trained for three different cases of which two different projections of the data. One xz projection case before any cut is made to the dataset. Two xy projection cases, one before the cut and one after. After the training process has converged to an optimum, the model is saved and tested. The testing procedure to measure the inference time of the model, is described before in section 3.2.3, but now adapted for the Keras model instead of the Pytorch model, with the *timeit* module. The training progress of the xy-plane projection after the cut of the data is displayed in Figure 18. It shows the two out of the four models where stopped before reaching the end of the training period. Although the smallest model 16×16 did train for the entire 50 epochs, it shows little improvement in the validation accuracy. The models suffer from overfitting relatively early on in the training process, resulting in the halting of the training or the minimal improvement.
The XY-plane projection case, the models weights and biases are quantized with the quantization aware training strategy as discussed in section 4.6.1, by retraining the weights for 3 epochs. The model is recompiled before QAT with a learning rate of 0.0005 with no weights decay, and sparse categorical cross entropy loss function that is the general case of the binary cross entropy loss function. This fine-tuning step is necessary for the smaller models, that are not durable enough for post training quantization. This strategy minimizes the accuracy loss to within one percent for the larger models, and the smallest model suffers more from the quantization, losing more than two percent accuracy. The results of the training procedure are displayed in Figure 18. The accuracy of the models is close to 99.0 percent at the first epoch, and slight adjustments are made after that. For the QAT process the accuracy of the larger models rises slightly over the 3 epochs and ensures the models performance on the FPGA. Post-training quantization reduces the models accuracy to 50% thus the QAT procedure is preferred.

4.4.3 Results

XY-projection
First the results from evaluating the xy-plane model are discussed. Training the model on the complete dataset resulted in the performance presented in table 3, and training on the reduced dataset in table 4. The models performance in general slightly decreases with the decreases size of the input image, as details of the particle showers structure are lost in the mapping process. In order to outperform the simple cut strategy an accuracy higher than 99.55% should be achieved. Training the model on the entire dataset, produced an accuracy of 99.77% for the largest input size model. This model had a total of 92 miss-classifications, in comparison the cut strategy produces 180 miss-classifications of which 136 are false negatives of the pion. The smaller models are performing slightly worse than the cut strategy with the smallest model losing a significant amount of accuracy. The smallest model has a signal efficiency of 98.6% at background efficiency of 1%. In this analysis however, we are interested in performance at background efficiencies below 1% for the entire dataset. Testing the inference of the models reveals a relation between model size and inference time. The smallest model is approximately four times faster than the largest model, with the total amount of parameters only halved. In order for this method to be applicable in a trigger in the ALICE detector, the latency has to be in the order of μs, or three orders of magnitude faster.



(a) (b)

Figure 18: (left) Training history on the reduced dataset after the cut, of xy-projected models for different quantizations of input data. The models accuracy is shown on the left y-axis, and is displayed above in the plot and the loss below with the y-axis on the right. (right) Quantization aware training history of xy-projected model on full dataset for three epochs.

The largest neural network is able to distinguish pion from e^+, and makes twice as few mistakes as the simple cut strategy. This improves the total statistics of the pion events in the ALICE detector, and will improve the significance of experimental findings. To improve the selection power of the model around 4000 total hits, and distinguish e^+ from early showering $\pi^\pm$, the model was also trained on a reduced dataset. This dataset consists of only events above 3000 hits to expose the neural network more to these types of events. However the total amount of $\pi^\pm$ that cascade early, relative to the e^+ is small, 808 to 19999. This has an effect on the training on the models, which often where halted early due to no improvements. Especially the smaller models where outperformed significantly by the models trained on the entire dataset. Nonetheless the largest model had an accuracy of 99.47% or 110 miss-classifications, which is compare-ble to the previously trained model.

Table 3: Models performance for different quantization of xy-plane projections.

Input Pixels	Accuracy Float (%)	Accuracy Quantized (%)	Parameters	Sig. Eff B=0.01	Latency CPU (ms)
16x16	98.95	96.25	26,082	0.986	2.32±0.11
32x32	99.35	98.83	29,154	0.988	3.65±0.04
64x64	99.52	99.38	41,442	1.000	5.20±0.17
128x128	99.77	99.49	53,762	1.000	9.82±1.89

Table 4: After cut Models performance for different quantization of xy-plane projections

Input Pixels	Accuracy Float (%)	ROC	Sig. Eff. B=0.01	Sig. Eff. B=0.10	Sig. Eff. B=0.30
16x16	95.77	0.932	0.730	0.830	0.920
32x32	96.35	0.957	0.762	0.898	0.953
64x64	98.03	0.976	0.840	0.935	0.976
128x128	99.47	0.994	0.947	0.971	0.980

XZ-projection

The xz-plane projection shows promising results in binary classification for all model sizes, compared to the xy-plane projection. Performance of the xz-plane projection is significantly higher for the smaller models with the 16×24 mapping, compared to the 32×32 xy-plane version. Comparing to the 16×16 model is an unequal comparison due to the reduced total pixel amount of 256 compared to 384 for the xz-plane. The smallest model for

the xz-plane projection has a validation accuracy of 99.62 % which is higher than that of the simple cut strategy. However the largest model only performs slightly better than the smallest model. This suggests is not possible to distinguish between early showering $\pi^{\pm}$ and e^{+} based on the information available in the image, or the training has to be improved. A larger data-set would allow for longer training periods without over-fitting. This would allow the model to generalize better and in turn improve the models accuracy. One could also try different model architectures, that are more focused towards capturing the structure in the first few layers of the cascade. A more extensive model with additional layers and more parameters can capture the detailed structure of the shower better. This however increases the inference of the models significantly, which is not desired in this study.

Table 5: Models performance for xz-plane projection data.

Input Pixels	Accuracy Float (%)	Parameters	ROC	Sig. Eff. B=0.01
16x24	99.62	16,658	0.999	1.000
32x24	99.67	20,898	0.999	1.000
64x24	99.67	29,378	1.000	1.000
128x24	99.70	54,274	1.000	1.000

4.4.4 Conclusion

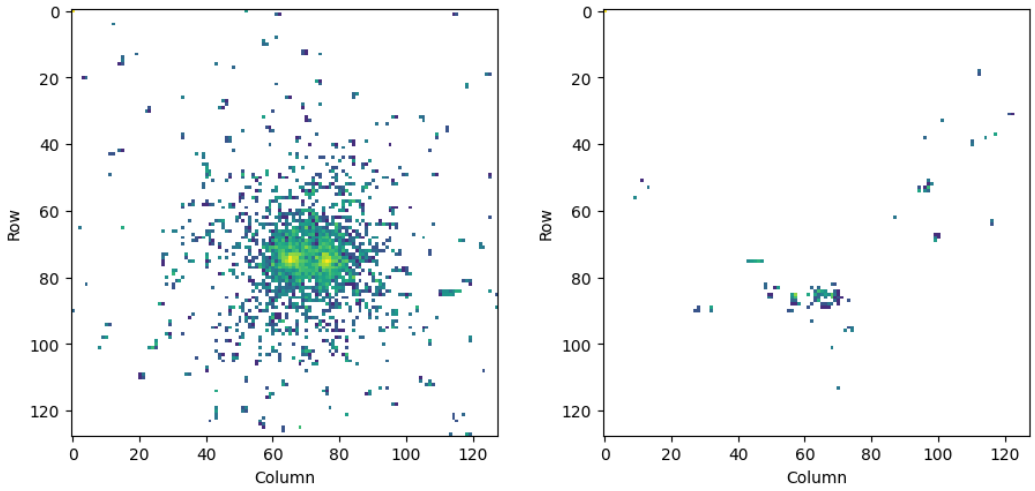
Pion and e^{+} showers can be classified better with larger networks that have many layers and parameters. However, with limited computational resources and timing constraints, as is the case for a trigger, a compromise is to be made. The best strategy to binary classify the pion and e^{+}, is to first introduce a simple cut off value, of 3000 hits total, below which all events are considered to be $\pi^{\pm}$. This reduces the pion events by 96%, without misclassifying e^{+}. After the cutoff, applying a convolutional neural network that is trained on the entire dataset. The larger the model the better the performance. To outperform the optimal cut value an accuracy above 99.55 % is required, this is possible for the largest model of size $128x128$ input pixels. This strategy can be improved upon by acquiring a larger dataset with more pion events than 3000 total hits. This would make the generalization of the model better and improve performance on the entire dataset.

When we are only interested in single pion and single e^{+} events, the xz-plane projections also show interest. These models outperform for each mapping size, the cutoff strategy. Although the largest model did not significantly outperform the xy largest model, with a larger dataset, the chance of overfitting becomes smaller, possibly increasing the performance of the model. Now the model slightly underperforms due to the overfitting that takes place relatively quickly, halting the training process.
In practice single particle decay events are hard to come by, overlapping showers of e^{+} and or $\pi^{\pm}$ are more common. In the next section, we are interested in distinguishing the different overlapping particle showers.

4.5 Overlapping jets

The ALICE experiments main goal is to study the QGP. There are several types of particle decays that can provide information about the QGP and its properties. Two particular decays of interest are the dilepton and dihadron decays. Dileptons are clean probes that interact only electromagnetically with the surroundings, allowing them to carry information from the production point to the detector. This allows for the study of the early stages of heavy-ion collisions[51], through the invariant mass spectra of dileptons.
Dihadron correlations are used to study the anisotropic flow of particles in the experiment[52][53]. Anisotropic flow refers to the preferred direction that particles take after being emitted from an anisotropic source, for instance, the QGP. This allows for the detailed study of properties of the QGP; for example, the temperature and viscosity are probed using the measurement of the anisotropic flow.
Detecting dilepton and dihadron decays involves distinguishing particle showers that occur in close proximity to each other, resulting in overlapping cascades of particles. Additionally, due to the high multiplicity of the ALICE experiment, many particle decays occur close together, resulting in jets overlapping in the detector. This presents a challenge to distinguish between the different particles involved in decays. Using the calorimeter detector data, a reconstruction of the jets allows the identification of the particle. The overlapping of the jets makes it more difficult to distinguish from which jet the particle originated. Reconstruction of jets is an intense computing process that

commonly involves backpropagation of the particle tracks to its origin. Using convolutional neural networks, we aim to distinguish jets with high accuracy while keeping latency low. The study of overlapping jets is performed to determine the minimum distance between particle decays required to distinguish two particle decays. This in turn determines the resolution of the experiment and the precision with which we can track and analyze the particles. The distance between the centers of the jets that are of interest is varied from 1 cm down to 1 mm.



(a) (b)

Figure 19: (Left) Two combined e^+ events and (right) two combined $\pi^\pm$ events at a separation distance of $\Delta r =$ 25mm apart

4.5.1 Data preparation

Studying overlapping particle showers requires a labeled data set with showers of $\pi^\pm$ and e^+ decays. A xy-plane projection with two overlapping particle showers is generated, as explained in the following section. Each event is projected onto the xy-plane and mapped to 128×128 pixel image as before. Hereafter, the origin of the particles decay shower is determined. This is determined by finding the first layer of the detector that has several hits. This usually coincides with being the pixel with the most hits of that layer. The position of this pixel is marked as the origin of the particle shower. Then, different particle showers are combined by translating all hits of one shower relative to the origin of the other shower. This results in two particle showers that overlap with a distance Δr between each origin. Any hits that are moved outside of the detector range are discarded. This strategy does not consider any relative angle between the particles decay tracks and assumes the incident particle tracks are parallel. The relative distances Δr are given in Table 6. Hits that are translated outside of the image are discarded. A total of 20k events are generated for each class, randomly selecting events of $\pi^\pm$ or e^+ decays. Various cases are considered for analysis, namely a three-, four-, and five-class case. The three-class classification consists of overlapping $\pi^\pm$-$\pi^\pm$, $\pi^\pm$-e^+ and e^+-e^+ events. The four classes classification is made up of single $\pi^\pm$ and single e^+ events, and overlapping $\pi^\pm$-$\pi^\pm$ or e^+-e^+ events. The five classes consist of all the possible cases, single $\pi^\pm$ and e^+ events, overlapping $\pi^\pm$-$\pi^\pm$ and e^+-e^+ and lastly $\pi^\pm$-e^+ events. The simple cut strategy that worked in binary classification cannot distinguish between a $\pi^\pm$-e^+ event and a single e^+ event. To make the distinction between the overlapping particle showers, the structure of the cascade has to be studied. To study the limitations of the models resolution and performance, for each case the model is trained independently. The most general case with the five classes is the ultimate benchmark of the models performance.

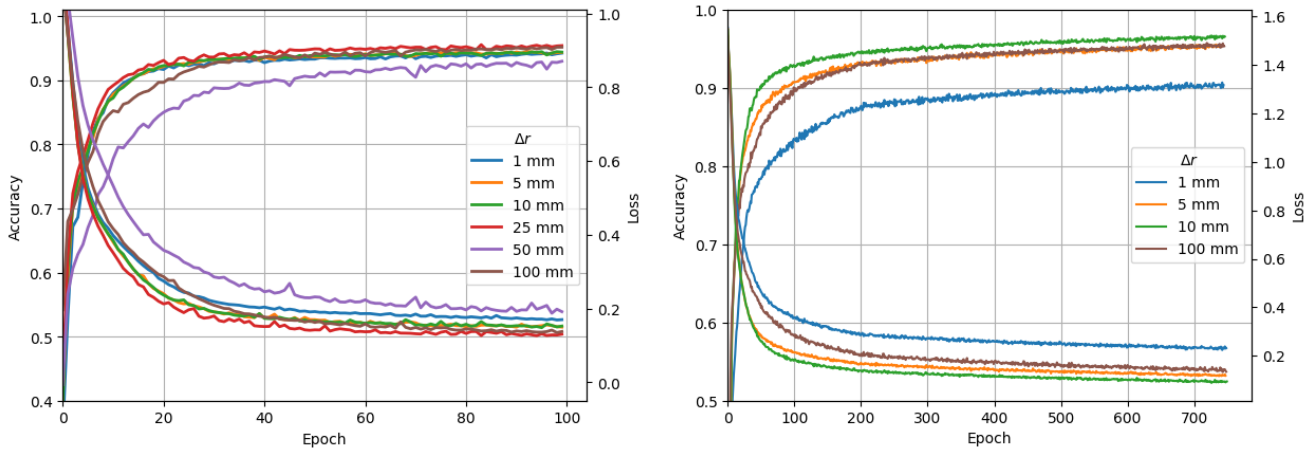
4.5.2 Model

For the model, a convolutional neural network is used that consists of 4 convolutional layers and has an input size for the image of 128×128. Each layer consists of a convolutional layer followed by a relu activation layer and a batch normalization layer. The last batch normalization layer is followed by a dense layer and a softmax activation

layer. The model consists of a total of 84,868 trainable parameters, and is compiled with a catagorical cross entropy loss function and the Adam optimizer, with a learning rate of 10^{-4}. No decay rate for the weights is set.

4.5.3 Training

Each case required a different combination of training parameters; the five-class case has to be trained more extensively than the four-class case. The four-class model is trained over a total of 100 epochs, after which the improvement in validation accuracy improved still. However, early stopping was set for 20 epochs. The five-class case is trained for up to 1000 epochs, with early stopping after 50 epochs of no improvement in the validation accuracy. The learning rate is adjusted after 250 epochs to half the learning rate and fine-tune the model.
The training history of the four-class case is shown in Figure 20a, and the five-class case in Figure 20b. Training of the model for over 700 epochs takes a lot of computing power; therefore, the training of this model was performed on an Imac Studio with Apple M1 Ultra silicon chip, with 20 CPU cores. This significantly speed up the training of the model. After roughly 250 epochs the models performance improvement slows down; now the weights are fine-tuned for minor improvements. Classifying five classes proves to be more difficult than four classes, with the overall accuracy of the trained model being lower at small separations. This change in accuracy is completely due to the added class of overlapping $\pi^{\pm}$-e^{+}. The model cannot distinguish properly between $\pi^{\pm}$-$\pi^{\pm}$ or single e^{+}. This effect gets worse as the distance between the jets is lowered.



(a) (b)

Figure 20: (Left) Training history of 4 class model.(Right) Training history of 5 class model.

4.5.4 Results

The models accuracy after the training period is shown in Table 6. The testing of the model resulted in confusion matrices that are graphically displayed in Figure 21 for the four class case, and in 23 for the five class case.
In the four-class case, the predictive power of the models decreases slightly as the distance between the centers of the showers decreases. For the largest distance tested, the accuracy was the highest, but not as high as the binary classification. The distinction between single $\pi^{\pm}$ events and double $\pi^{\pm}$ events remains difficult using this projection, and the model is not able to generalize in all cases. The model trained for jets separated $\Delta r = 50mm$ performed the worse but still withing reasonable standard deviations. The variations in accuracy of the models are partly due to the variance of the datasets. A data set at Δr can have more than average $\pi^{\pm}$-$\pi^{\pm}$ events that have more than 3000 hits. As seen before in the binary classification of particle showers, it is difficult to distinguish $\pi^{\pm}$ decay showers from e^{+}, for $\pi^{\pm}$ events that shower in the first layers of the detector.
Nonetheless, as displayed in the confusion matrix, the single e^{+} events are separated from the double e^{+} events with an accuracy of 93.4% for background efficiency $B = 0.01\%$ $\delta r = 1$mm. At this distance there are only 3 pixels in-between the two centers of the showers. The shower tagging resolution of the detector is therefore higher than 93% for two probe e^{+}. The accuracy decreases, however, for larger separations. The center shower has the highest amount of hits, with two centers so close to each other, the maximum of the centers increases significantly. The model picked up the increase of maximum at double e^{+} events, and is worse at distinguishing the e^{+} larger

separation, since the maximum of the two overlapping showers is decreased. Even at the smallest separation distance $\Delta r = 1$mm, the model is able to distinguish 61% of the single $\pi^{\pm}$ events from the double $\pi^{\pm}$ events at a background efficiency of $B = 0.01\%$.

Table 6: Classifiers distinguishing particle showers for varying separations, trained on multiple classes.

Δr (mm)	Accuracy 4-class	Accuracy 5-class
100	95.16	95.36
50	92.96	–
25	95.95	–
10	94.39	96.60
5	94.33	95.54
1	94.24	90.38

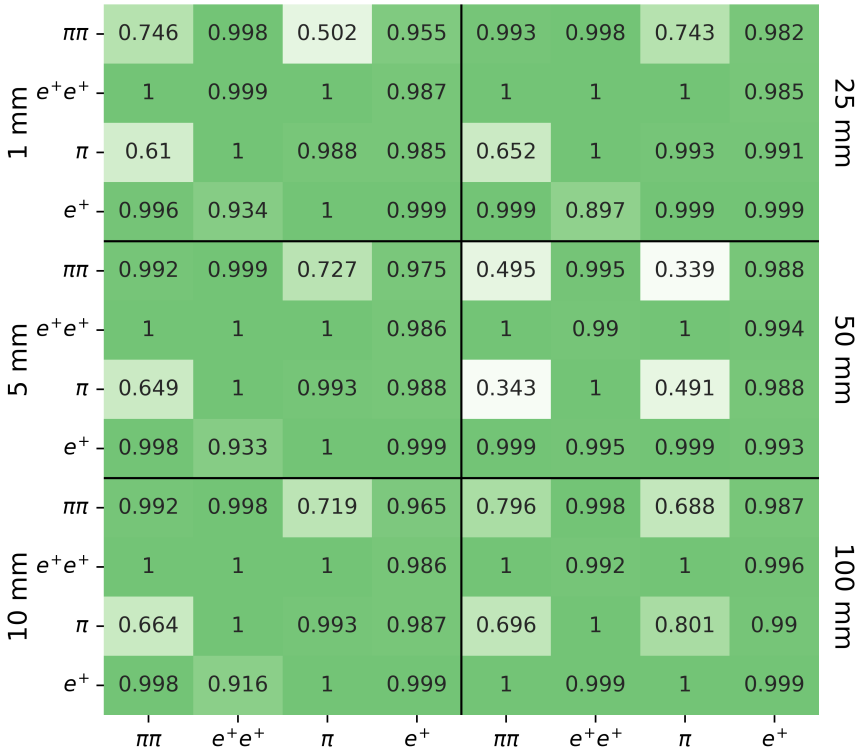
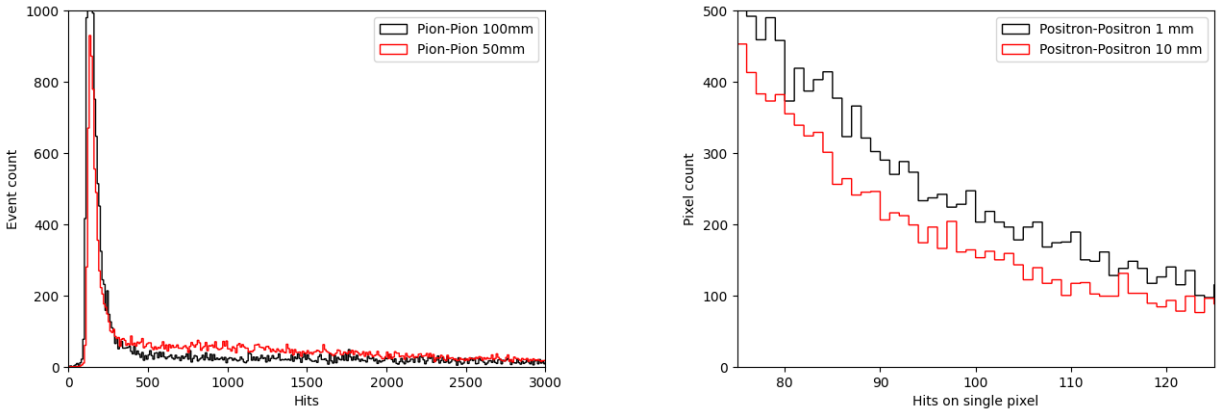


Figure 21: 4-Class Confusion matrix of Signal efficiency at background efficiency $B = 0.01\%$. The class on the y-axis is the true class, and the class on the x-axis the predicted class. On the diagonal of the matrix is the signal efficiency with all the other classes considered as background.

The model is trained on data that consists of six different sets of overlapping showers, with varying relative distances between the centers of the showers. The resulting signal efficiencies per class are shown in the confusion matrix in figure 21. The confusion matrix displays the classes that are difficult to distinguish by the model. As expected, the signal efficiency of the model for double $\pi^{\pm}$ and single $\pi^{\pm}$ events is lower. The model confuses the single event for the double event and vice versa, independent of the distance between the showers. The 50 mm dataset performs considerably worse than the other distances, due to the low signal efficiency with the single and double $\pi^{\pm}$ events. To better understand why the accuracy for this data set is relatively lower than that of the other distances, the double $\pi^{\pm}$ data set is shown in figure 22a. The histogram suggests that the 50 mm dataset has relatively more double $\pi^{\pm}$ events that have large amounts of hits in the detector. The simulation events are selected at random before merging. With low statistics, it is not unlikely that a significant amount of the selected events will have more events, although this has a lower probability. With a higher amount of events and a larger data set, the probability

of this happening decays exponentially as the standard deviation decreases.
At a separation of 1mm between the centers of two $\pi^{\pm}$ showers, the distinction between individual showers becomes challenging, often appearing analogous to a singular $\pi^{\pm}$ shower event. This results in a lower signal efficiency of 0.746 for the double $\pi^{\pm}$ shower at a background efficiency of B=0.01. The shortfall is attributed to the misinterpretation of the double $\pi^{\pm}$ shower by the models as a single $\pi^{\pm}$ shower event. The true double $\pi^{\pm}$ class predicted as the $\pi^{\pm}$ class is found to have a signal efficiency of 0.502, similarly the true single $\pi^{\pm}$ class predicted as the double $\pi^{\pm}$ class has a signal efficiency of 0.61. With increased distances, the signal efficiency improves noticeably due to the emergence of two clear centers in the $\pi^{\pm}$ track pattern within the detector. The predictive accuracy of the model decreases when $\pi^{\pm}$ registers a higher hit count in the detector. This is explainable by the pattern of the single $\pi^{\pm}$ that produces a large number of hits, bearing similarity to that of two $\pi^{\pm}$ with fewer hits.
Nonetheless the model's performance at all the distances show great promise in being able to function as a $\pi^{\pm}$ or e^{+} tagging algorithm. Even when the cascades produced by the particles overlap inside the detector, it is possible to distinguish each particle and therefore detect probes in heavy ion collisions that can have several particle cascade happening in close proximity. The ability to detect these particles and distinguish between each class is valuable in improving the overall accuracy of the experiments with improved statistics. Now to simulate the most general case inside the detector, a five class model is considered as well. The overlapping of $\pi^{\pm}$ and e^{+} cascades had been left out before in the four class case, as the accuracy of the model was expected to be low. Now that the baseline is established with the four class case, the model is trained for the five-class case.
The resulting confusion matrix is displayed in figure 23. The addition of the overlapping e^{+} $\pi^{\pm}$ shower has a siginificant effect on the overall predictive power of the model, due to the difficulty of distinguishing a $\pi^{\pm}$ shower that overlaps with a e^{+} shower. The signal efficiency of the $\pi^{\pm}$-e^{+} class is decreasing as the distance between the showers decreases. Even when there are only 4 pixels between the center of the $\pi^{\pm}$ shower and the e^{+} shower, a signal efficiency of 0.674 is achieved with background efficiency B = 0.01 is achieved. The signal efficiency of the e^{+} class at background efficiency B=0.01 is 0.638 for this seperation. This is a significant decrease in predictive power compared to the four class model and is completely attributed to the addition of the possibility of overlapping $\pi^{\pm}$ e^{+} showers. The quantization down to 128by128 pixels sums over eight neighbouring pixels resulting in the centers of the showers being merged.



(a) (b)

Figure 22: (Left) Event count versus total hits in detector at two different distances between centers of $\pi^{\pm}$ showers. The simulation data from the double $\pi^{\pm}$ showers at 50mm separation shows relatively more events with higher amount of hits compared to the 100mm data set. (Right) Pixel count histogram versus total hits on single pixel for double e^{+} events at 1 mm and 10 mm separation. At smaller relative distances the centers of the showers overlap resulting in high amount of hits in a single pixel.

Nonetheless the information is not completely lost as the model to make a classification between the overlapping showers, due to the increased number of pixels hits. The model's misinterpretation of a e^{+} as an overlapping e^{+} $\pi^{\pm}$ event, and vice versa is as expected. This shortfall in the predictive power of the models is reflected in the low signal efficiency for the signal e^{+} and background $\pi^{\pm}$ and vice versa. This effect gets worse as the separation between the showers centers becomes smaller.
In conclusion the model is able to accurately classify the five classes of particle showers, even down to small relative distances of the showers centers. Detecting and identifying particles based on the xy projection of the showers pattern

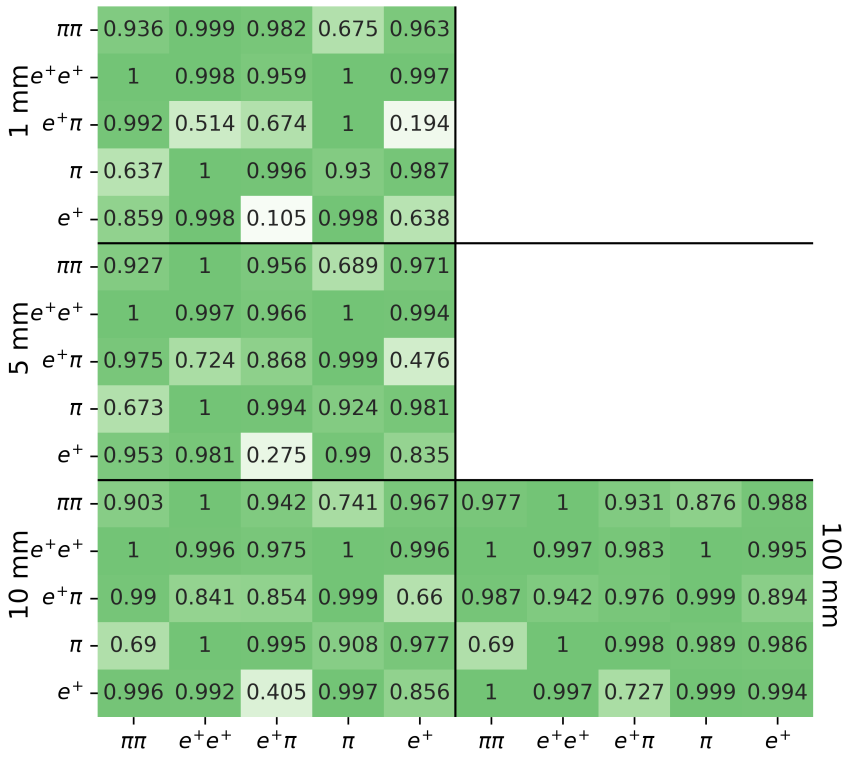


Figure 23: 5-Class Confusion matrix of Signal efficiency at background efficiency $B = 0.01\%$. The class on the y-axis is the true class, and the class on the x-axis the predicted class. On the diagonal of the matrix is the signal efficiency with all the other classes considered as background.

in the calorimeter detector is made possible by the distinct patterns 2dimensional patterns that are produced in the detector. The structure of the shower holds information on the particles class. This information would be lost if a selection would be made solely on the total amount of hits in the detector as was done for the optimal cut in the binary classification. Probing the structure of the particles showers inside the detector is necessary to preserve as much particle events in the experiment, and improve the statistics of the overall experiment.

4.6 Shower Classification on FPGA

In the following section the discussion of the deployment of the model on the FPGA takes place. The binary classification of particle showers was performed using CPU trained convolutional neural networks. In order to deploy the model onto the FPGA, the parameters of the model are quantized to 8-bit accuracy from the floating point accuracy. This process is of quantization is performed using quantization aware training as described in section 4.6.1 and the resulting training accuracy displayed in figure 18b.

4.6.1 Quantization Aware Training

Post-training quantization-aware training leads to significant accuracy drops for small models. Therefore, quantized learning is introduced by simulating quantization effects in the forward pass of the learning process. Quantization-Aware Training (QAT) is implemented to minimize accuracy drops due to quantization of small model. QAT can be implemented in Tensorflow 2 with the following block of code:

```
from tensorflow_model_optimization.quantization.keras import vitis_quantize
quantizer = vitis_quantize.VitisQuantizer(model)
qat_model = quantizer.get_qat_model(
    init_quant=True,
    calib_dataset=calib_dataset)

///compiling and training
```

```
quantized_model = vitis_quantizer.get_deploy_model(qat_model)
```

As indicated by the comments, the compiling and training takes place before a call to get the deployable model is made. This converts the quantized model to a model that is understood by the compiler. The quantization strategy that is implemented in QAT is described in technical terms, taken from [25]. The approach simulates quantization effects in the forward pass of training. In backpropagation, nothing changes compared to normal training of a model. In the forward pass, the quantization inference is simulated by implementing in floating-point arithmetic the rounding effects that occur in the quantization scheme, which is stated below. First, the weights are quantized before any convolution with the input data. For the use of batch normalization, the corresponding parameters are "folded into" the weights before quantization. These are later to be folded back for deployment. Second, activations are quantized at point of inference, for instance after applying the convolutional layer then the ReLU activation function is quantized. For each layer, the quantization is point-wise applied with the function defined as:

$$\begin{aligned} clamp(r; a, b) &:= min(max(x, a), b), \\ s(a, b, n) &:= \frac{b-a}{n-1}, \\ q(r; a, b, n) &:= [\frac{clamp(r; a, b) - a}{s(a, b, n)}]s(a, b, n)) + a, \end{aligned} \tag{21}$$

Where r is a real-valued number to be quantized, with $[a; b]$ the quantization range, n the number of quantizaion levels, i.e. $2^8 = 256$ for 8-bit quantization strategy, and [.] denoted the rounding to the nearest integer.
The QAT algorithm is then given as follow:

1. Create floating-point model
2. Insert fake quantization operations according to equation 21.
3. Train the model
4. Create and optimize the model
5. Test inference

quantization scheme
The scheme with which the quantization takes place is introduced:

$$r = S(q - Z), \tag{22}$$

for some constants S and Z that are the quantization parameters. Each activation array and weights array have its own quantization parameters and thus scheme. In the case of 8-bit quantization, for instance, of weights in the layers, the float value is reduced to an 8-bit representation. The constant S is an arbitrary positive real number represented as a floating point on the cpu architecture. The constant Z is of the same type as the quantized value q.
Integer-arithmetic-only matrix multiplication
The translation from the computation of real numbers to the computation of quantized values is performed using a matrix multiplication that involves only integer arithmetic. The scaling value S is in floating point precision. Consider the multiplication of two square matrices $N \times N$ of real numbers r_1 and r_2. The product of the multiplications is represented by $r_3 = r_1 r_2$.
The matrix consists of the entries $r_\alpha^{(i,j))}$ with $1 \leq i, j \leq N$ for matrix ($\alpha = 1, 2or3$). Equation 22 then is written as:

$$r_\alpha^{(i,j)} = S_\alpha(q_\alpha^{(i,j)} - Z_\alpha). \tag{23}$$

The corresponding quantization parameters with which they are quantized are depicted as (S_α, Z_α). Quantized entries are denoted as $q_\alpha^{(i,j)}$. This matrix multiplication implies that we have:

$$S_3(q_3^{(i,j)} - Z_3) = \sum_{j=1}^{N} S_1(q_1^{(i,j)} - Z_1)S_1(q_2^{(i,j)} - Z_2), \tag{24}$$

This then rewritten as

$$q_3^{(i,k)} = Z_3 + M \sum_{j=1}^{N} (q_1^{(i,j)} - Z_1)(q_2^{(i,j)} - Z_2), \tag{25}$$

Where we define the multiplier M as

$$M = \frac{S_1 S_2}{S_3}. \tag{26}$$

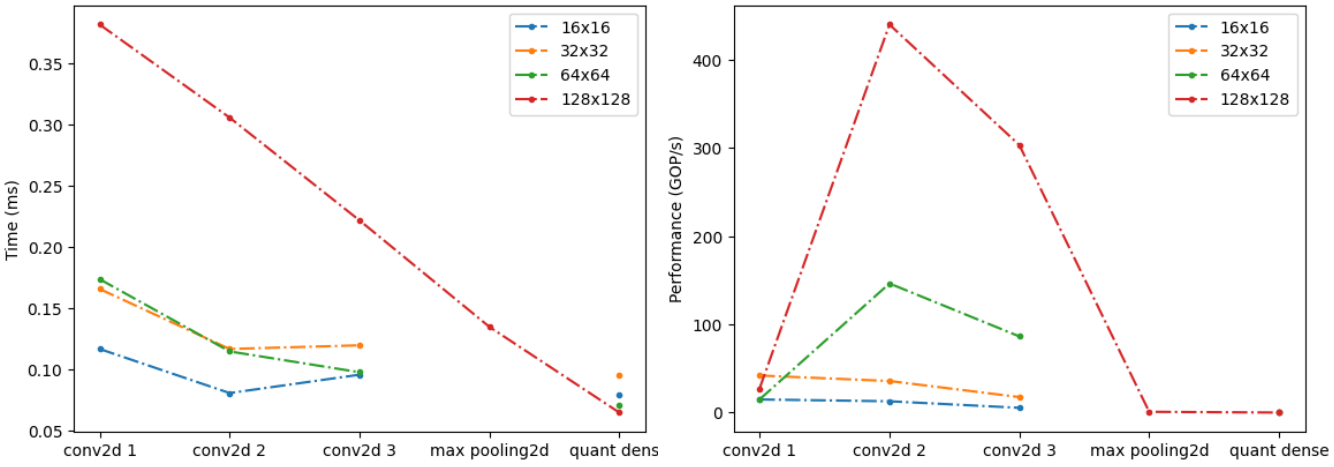
Now the matrix multiplication involves only integers, except for the multiplier M that depends only on the constant S_α. The multiplier M is empirically found to only to be in the interval $(0, 1)$, and therefore can be expressed in the normalized form of:

$$M = 2^{-n} M_0, \tag{27}$$

where M_0 is in the interval $[0.5, 1)$ and n is a nonnegative integer. This normalized multiplier can now be expressed as a fixed point multiplier in a 2n-bit integer representation. For hardware 8Int, M_0 is represented by the Int8 integer closest to the value of $2^7 M_0$. Due to the constraint that $M_0 \geq 0.5$ the value always is at least 2^6, ensuring a 6-bit precision relative to the floating point value of M. Now, for the multiplication by 2^{-n} can be implemented by means of an efficient bit shift, that is, to be correctly implemented such that the round-to-nearest behavior is correct. The implementation in assembly code of how to achieve this on the arm architecture goes beyond the scope of this project.

4.6.2 Results

For smaller models that consist of fewer weights, the accuracy during the QAT is slightly lower than for the larger models with more weights. This is due to the robustness of the model which can be attributed to the absolute number of parameters of the model. After the quantization of the models, they are ready to be compiled for the target device by the vitis ai compiler. This translates the .h5 keras model to an binary .xmodel file that the DPU can interpret. This xmodel holds the information of where calculations take place on the DPU model, and what connections are made in order to perform the calculations of the model. The Vitis AI compiler takes care of this process but does not directly synthesize the best possible model layout for the DPU. A custom layout of the model on the DPU is possible with the use of Vivado HLS. To get a clearer understanding of how the model performs on the FPGA and how each individual layer of the network performs, the Vitis AI analyzer is used. This software allows for the analysis of the models resources on the DPU. The latency for each layer is displayed in figure 24a for all four models sizes. The timed latency per layer does not add up to the total latency of the model since the timing operation of the analyzer inflicts a penalty on the latency. This does, however, give a comparative indication of each layers performance. The performance of each layer as operations per second is displayed in figure 24.



(a) (b)

Figure 24: Performance of the model on the DPU. (Left) Latency of each layer timed using the Vitis ai tracing library. (Right) Performance of each layer in GOP/s, measured using the Vitis ai tracing library.

The latency of the individual layers varies depending on the size of the model and which consecutive layer it is. The first layer of the network always seems to take the longest amount of time to compute, this is due to the

operations per second of the first layer being low compared to the others. The largest model with input images of 128x128 pixels takes the longest to compute but does show the largest performance in operations per second, utilizing the theoretical 1.4 TOPS of the Kria board the most of all the models. Still this theoretical performance is never seen, as most layers are atleast 10 times slower in operations per second than the indicated 1.4 TOPs. Several factors are potentially limiting the performance of the FPGA, resource utilization, design optimization, clock frequency, and compiler tool limitations. Resource utilization not all the resources of the FPGA are utilized. The architecture consists of numerous LUTs and DSP blocks in the DPU; if not all of these blocks are used, theoretical maximum performance can never be reached. Design optimization can also limit performance if the design is not fully optimized for parallel processing or pipelining. This could limit the FPGAs performance. The clock frequency might not be as high as the theoretical maximum frequency, which limits the overal operations per second that can be achieved. The compiler tool used might not be able to fully optimize the algorithm for the FPGA fabric, this goes together with the resource utilization in the sense that the compiler tool does not place the operations as efficiently on the FPGA fabric, thus not utilizing the resources to its maximum potential. The max-pooling layer for the largest model induces latency as can be seen in figure 24a. The maxpooling layer should be merged with the convolutional layer as according to the documentation of Vitis AI, this nonetheless did not happen. This extra layer adds unnecessary latency to the execution time and should be removed for inference testing. After the model compilation, it can be plotted in a schematic with an indication of what device each layer

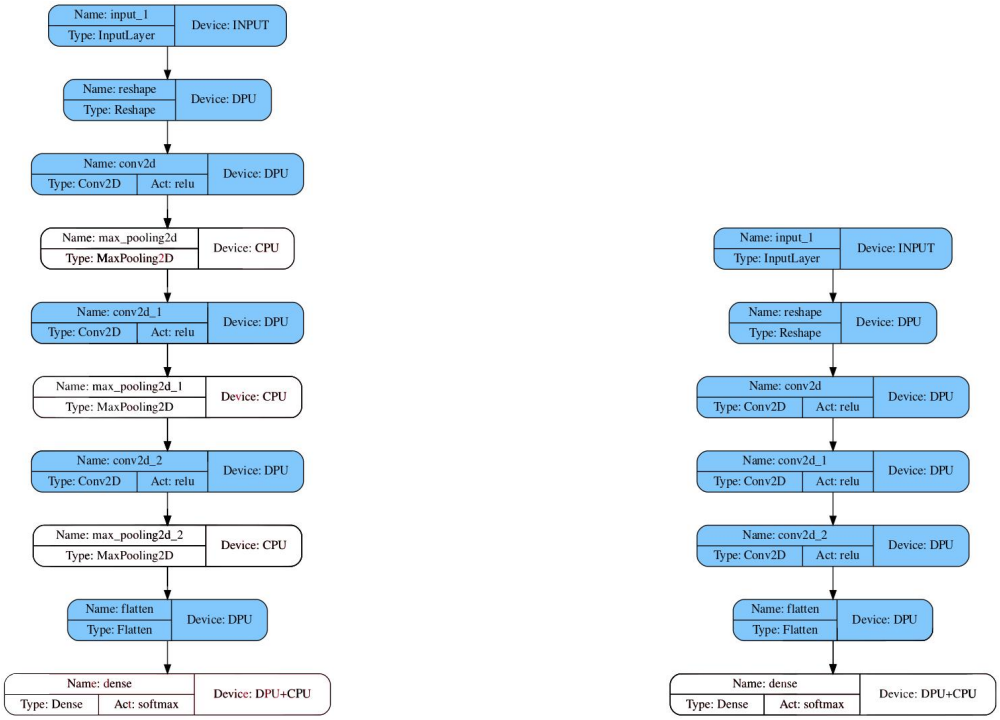
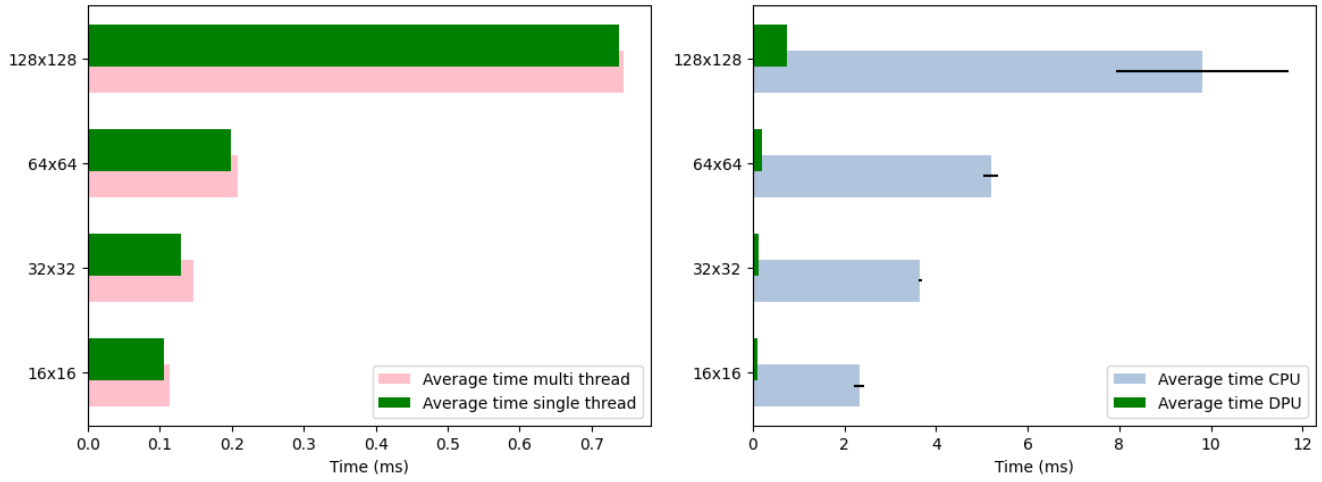


Figure 25: Model as compiled by the Vitis Ai compilation software. (Left) The max pooling layers are not compiled on the DPU architecture but rather on the CPU. (Right) The Max pooling layers are removed, hence there are no intermediate layers compiled for the CPU, removing potential latency reduction.

is compiled on. Surprisingly the model is split between CPU and DPU architecture due to the lack of DPU support for some types of layers. As seen in 25 the max pooling layers are not compiled correctly for the DPU architecture and therefore are computed on the CPU. This indicates that not only for the larger model the Maxpooling layers compilation process failed, but for all models. This forms an issue and increases the latency of the model, therefore the maxpooling layers are removed from the model, significantly decreasing the latency of the model. The smallest models time to classify 4000 events, improved by $\approx 33\%$ after removing the maxpooling layers.
The final comparison of latency for the model between CPU and DPU is displayed in figure 26. The average time of the computation on the CPU is about 30 times longer than on the DPU for the same model. For the smallest model of 16x16 the total time of the computation is $100\mu s$. But the latency barely increases up to the 64x64 model, boasting a higher accuracy.



(a) (b)

Figure 26: (Left) Average time of multi and single threaded computation of one single image on DPU. (Right) Comparison between FPGA and CPU times of model.

The effect of multithreading and single threading of the computation is measured by timing the individual time it takes for the image to be classified. This does not improve the total time of the classification, as the speed up of multithreading occurs only because multiple images can be classified simultaneously. This can have an positive effect on the total time it takes for x images to be classified. The time it takes for a single image to be classified using single or multithreading is shown in figure 26a, and is just slightly longer for the multi-threading task than for the single-threading task. The latency of the model on the DPU is then compared to the CPU in figure 26b with the FPGA being significantly faster. The smallest 16 by 16 image size is approximately 100μs for one image to be classified. As the model size increases, the latency on the FPGA slightly increases up to the largest model which is significantly slower with about 700μs latency per image. The results show that on average the FPGA performs 30 times faster than the CPU.

Table 7: Timing results of the FPGA and the CPU for all the different model sizes, measured in milliseconds.

Model	CPU (ms)	DPU (ms)
128x128	9.82±1.89	0.738
64x64	5.20±0.17	0.199
32x32	3.65±0.04	0.129
16x16	2.32±0.11	0.105

4.7 Conclusion

Simulated particle showers in the Epical-2 prototype calorimeter were successfully classified with the use of convolutional neural networks. To determine whether CNN classification was successful, the performance of the model is compared to a simple first-order cut on the total number of hits. By making the optimal cut of 4604 hits in the detector, it is possible to classify 99.55% of the events successfully. However, in doing so, the cut loses efficiency for $\pi^{\pm}$ that shower early in the detector and have relatively many hits for a $\pi^{\pm}$ shower. $\pi^{\pm}$ events with over 3000 total hits are difficult to distinguish based on a visualization of the shower in the detector, as it looks similar to the e^{+} events. For the CNN to outperform the optimal cut, it has to learn a pattern in the early showering $\pi^{\pm}$ structures on which it can distinguish it from the e^{+} showers. To tackle this issue, different strategies were adopted, training the CNN after making the cut and considering both an xz-projection and an xy-projection.
Training the CNN on the reduced data set was unsuccessful as the model overfitted the data due to the limited number of $\pi^{\pm}$ events after the cut. This can be addressed by using a larger dataset. The xz-projection at first seemed promising, as the smaller models achieved much higher accuracies than for the xy-projection; however, the

largest model with the highest accuracy did not outperform the xy-projection. With the use of a CNN on the xy-projected input images of quantization 128×128 an accuracy of 99.70% was achieved, outperforming the optimal cut strategy. The optimal cut although faster, due to the first-order calculation being performed of the total particle hits, loses potential $\pi^{\pm}$ events. These events are valuable for further analysis of the ALICE experiment in order to improve the overall statistics of the experiment.

In the high luminosity ALICE experiment, it is common for particle showers to reside in close proximity to each other. Showers can start to overlap in the calorimeter, which makes event reconstruction more difficult. Therefore, a more advanced study of classifying overlapping particle showers using neural networks was performed, to determine the minimal distance required between the centers of the showers. The data set for this analysis consisted of combined $\pi^{\pm}$ and e^{+} events. Two different studies of four and five classes were conducted, with the latter being the most realistic. Up to 10mm of relative distance the model was even able to accurately distinguish a overlapping $\pi^{\pm}$ and e^{+} shower from a single e^{+} shower. The signal efficiency for the e^{+}-$\pi^{\pm}$ class was 0.868 with a background efficiency of $B = 0.01$ with a distance of only 5mm between the shower centers. The accuracy of the model at this distance was 95.54% for the validation set. It is concluded that CNN is capable of accurately detecting a $\pi^{\pm}$ that is only 10mm apart from a e^{+}. The other cases of overlapping particles are able to be distinguished up to the close separation considered of 1mm.

After training and testing the CNN on the CPU, the binary classification network was deployed on the FPGA for inference measurements and comparison with the CPU. Using QAT and by optimizing the model slightly for the FPGA a latency in the order of $O(100\mu s)$ was achieved for the smaller models (16by16 and 32by32). The smallest model achieved on average an inference of 100μs on the DPU. Together with the 32by32 model, the respective inference was a factor of 30 faster compared to the CPU. These models did not fully utilize the theoretical performance of the FPGA of 1.4TOP/s, with the smallest model managing only 10 GOP/s. This indicates the model can be optimized more for the FPGA fabric by pruning the model and paying attention to the resource utilization of the FPGA. Using Vitis AI the model is deployed onto a general DPU layout for the KRIA kv260, but a custom DPU layout can be produced with the use of Vivado HLS, improving the performance of the model. In conclusion, the classification of binary $\pi^{\pm}$ e^{+} particle showers was successfully deployed on the FPGA hardware with a speedup of a factor 30. The potential performance of the FPGA was not utilized and further improvements are tenable. Using the FPGA as a level-1 hardware trigger in the ALICE experiment requires the latency of the model to be in the order of $O(1\mu s)$, this is two orders of magnitude faster than the CNN we were able to deploy on the FPGA. Nevertheless, the study showed the potential performance improvement of the FPGA that could be used in the online processing stage of the experiment to classify and reconstruct the particle events. By deploying a larger NN on several FPGA's that work concurrently, classification and identification of complex reconstruction events would be possible in the online processing stage with a 30 times speed up compared to CPUs.

5 Outlook

Employing machine learning on the FPGA for classification problems shows great potential to reduce latency. The Kria kv260 used for this study has limited resources and performance compared to more powerful but expensive FPGA's. A more resourceful FPGA would allow us to use a more extensive and powerful model with higher predictive power. Furthermore, the additional resources allow for more concurrent calculations, and thus a reduction in latency. However, the resources of the Kria board were not fully utilized, as the measured performance did not come close to the theoretical value. By testing different custom DPU layouts, the optimal configuration can be found, ultimately maximizing the performance of the chip.
A further optimization of the hardware of FPGAs on a board with more resources has the potential to significantly reduce the computation time during the offline processing of the experiment. Level-1 hardware triggers are required to operate in the microsecond range, therefore, using a neural network on the FPGA is not a viable candidate for the trigger as it operates two orders of magnitude too slow. Nevertheless, the speedup of the custom FPGA architecture compared to CPUs can be used in the offline processing stage. In this stage of the data processing, it is possible to use more complex analysis, as the latency requirements are less severe and the data rates do not have to keep up with the experiment. For instance, the classification of overlapping particle showers can be carried out in this stage by an FPGA using a large neural network. Currently, the offline processing in the ALICE experiment is done using GPUs. Our study, having demonstrated that the FPGA is about 30 times faster than the CPU, assures that this device would also be several times faster than a GPU. Current and future particle colliders have complex detector geometry together with large amounts of hits and events. It is difficult to use simple track reconstruction algorithms for jet tagging, with the high luminosity of the experiment, as many events build up in the detector before readout. The machine learning approach used in this study could efficiently classify events from the complex data provided, and promises to be the new standard for event processing.
Already, this study showed that it is possible to classify overlapping particle showers up to close separation with precision, using a compact lightweight neural network. To take full advantage of the parallel computations of the FPGA fabric, a larger neural network with even more layers can be used to classify these overlapping showers. Suggestions for further reducing latency include pruning the network and maximizing hardware utilization using an optimized custom hardware layout. Using hls4ml together with qkeras allows for tuning the amount of parallelization of the model. Also, the amount of bits used in the quantization scheme can be n-bit instead of the 8-bit of the Vitis AI quantizer. In this study only the Vitis AI software was considered, as it allows for a relatively simple workflow and eventually deployment of the model on the FPGA.

References

[1] Georges Lemaitre. A Homogeneous Universe of Constant Mass and Growing Radius Accounting for the Radial Velocity of Extragalactic Nebulae. *Annales Soc. Sci. Bruxelles A*, 47:49–59, 1927. doi: 10.1007/s10714-013-1548-3.

[2] Edwin Hubble. A relation between distance and radial velocity among extra-galactic nebulae. *Proceedings of the National Academy of Sciences*, 15(3):168–173, 1929. doi: 10.1073/pnas.15.3.168. URL https://www.pnas.org/doi/abs/10.1073/pnas.15.3.168.

[3] Rolf Hagedorn. Statistical thermodynamics of strong interactions at high energies. *Nuovo Cimento, Suppl.*, 3: 147–186, 1965. URL http://cds.cern.ch/record/346206.

[4] Karl Rupp. 50 years of microprocessor trend data. https://www.karlrupp.net/2018/02/42-years-of-microprocessor-trend-data/, 2018.

[5] Peter W. Higgs. Broken symmetries and the masses of gauge bosons. *Phys. Rev. Lett.*, 13:508–509, Oct 1964. doi: 10.1103/PhysRevLett.13.508. URL https://link.aps.org/doi/10.1103/PhysRevLett.13.508.

[6] K. Aamodt et al. The ALICE experiment at the CERN LHC. *JINST*, 3:S08002, 2008. doi: 10.1088/1748-0221/3/08/S08002.

[7] Johann Rafelski. Melting hadrons, boiling quarks. *The European Physical Journal A*, 51(9), sep 2015. doi: 10.1140/epja/i2015-15114-0. URL https://doi.org/10.1140%2Fepja%2Fi2015-15114-0.

[8] F. Reidt. Upgrade of the ALICE ITS detector. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 1032:166632, jun 2022. doi: 10.1016/j.nima.2022.166632. URL https://doi.org/10.1016%2Fj.nima.2022.166632.

[9] Rainer Schicker. Overview of ALICE results in pp, pA and AA collisions. *EPJ Web of Conferences*, 138:01021, 2017. doi: 10.1051/epjconf/201713801021. URL https://doi.org/10.1051%2Fepjconf%2F201713801021.

[10] Luciano Musa. Letter of Intent for an ALICE ITS Upgrade in LS3. Technical report, CERN, Geneva, 2019. URL https://cds.cern.ch/record/2703140.

[11] Letter of Intent: A Forward Calorimeter (FoCal) in the ALICE experiment. Technical report, CERN, Geneva, 2020. URL https://cds.cern.ch/record/2719928.

[12] M. Mager. Alpide, the monolithic active pixel sensor for the alice its upgrade. *Nuclear Instruments and Methods in Physics Research Section A Accelerators Spectrometers Detectors and Associated Equipment*, 824:434–438, 07 2016. doi: 10.1016/j.nima.2015.09.057.

[13] Letter of intent for ALICE 3: A next generation heavy-ion experiment at the LHC. Technical report, CERN, Geneva, 2022. URL https://cds.cern.ch/record/2803563. 202 pages, 103 captioned figures, 19 tables.

[14] Chiara Zampolli. Alice data processing for run 3 and run 4 at the lhc. 12 2020.

[15] Rohr, David. Usage of gpus in alice online and offline processing during lhc run 3. *EPJ Web Conf.*, 251:04026, 2021. doi: 10.1051/epjconf/202125104026. URL https://doi.org/10.1051/epjconf/202125104026.

[16] Real-time data processing in the alice high level trigger at the lhc. *Computer Physics Communications*, 242:25–48, 2019. ISSN 0010-4655. doi: https://doi.org/10.1016/j.cpc.2019.04.011. URL https://www.sciencedirect.com/science/article/pii/S0010465519301250.

[17] Ronak Bajaj and Suhaib Fahmy. Mapping for maximum performance on fpga dsp blocks. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 35:1–1, 01 2015. doi: 10.1109/TCAD.2015.2474363.

[18] T.K. Jappe, S.A. Mussa, and Richard Rosendo. Synchronous state machine inner fpga controlling pfc boost converter. pages 1097 – 1102, 08 2010. doi: 10.1109/ISIE.2010.5636865.

[19] Huilin Qu, Congqiao Li, and Sitian Qian. Particle transformer for jet tagging, 2022.

[20] Forrest N. Iandola, Song Han, Matthew W. Moskewicz, Khalid Ashraf, William J. Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and ¡0.5mb model size, 2016.

[21] Xiaohan Ding, guiguang ding, Xiangxin Zhou, Yuchen Guo, Jungong Han, and Ji Liu. Global sparse momentum sgd for pruning very deep neural networks. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/f34185c4ca5d58e781d4f14173d41e5d-Paper.pdf.

[22] Amir Gholami, Sehoon Kim, Zhen Dong, Zhewei Yao, Michael W. Mahoney, and Kurt Keutzer. A survey of quantization methods for efficient neural network inference. *CoRR*, abs/2103.13630, 2021. URL https://arxiv.org/abs/2103.13630.

[23] Ron Banner, Yury Nahshan, and Daniel Soudry. Post training 4-bit quantization of convolutional networks for rapid-deployment. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/c0a62e133894cdce435bcb4a5df1db2d-Paper.pdf.

[24] Daisuke Miyashita, Edward H. Lee, and Boris Murmann. Convolutional neural networks using logarithmic data representation. *CoRR*, abs/1603.01025, 2016. URL http://arxiv.org/abs/1603.01025.

[25] Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. 2017. doi: 10.48550/ARXIV.1712.05877. URL https://arxiv.org/abs/1712.05877.

[26] Claudionor N. Coelho, Aki Kuusela, Shan Li, Hao Zhuang, Jennifer Ngadiuba, Thea Klaeboe Aarrestad, Vladimir Loncar, Maurizio Pierini, Adrian Alan Pol, and Sioni Summers. Automatic heterogeneous quantization of deep neural networks for low-latency inference on the edge for particle detectors. *Nature Machine Intelligence*, 3(8):675–686, Aug 2021. ISSN 2522-5839. doi: 10.1038/s42256-021-00356-5. URL https://doi.org/10.1038/s42256-021-00356-5.

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

[28] Thea Aarrestad et al. Fast convolutional neural networks on FPGAs with hls4ml. *Mach. Learn. Sci. Tech.*, 2 (4):045015, 2021. doi: 10.1088/2632-2153/ac0ea1.

[29] Stylianos I. Venieris, Alexandros Kouris, and Christos-Savvas Bouganis. Toolflows for mapping convolutional neural networks on fpgas: A survey and future directions, 2018.

[30] Giovanni Guasti. Speed up the deployment path with vitis ai 2.5. https://support.xilinx.com/s/article/Speed-up-the-deployment-path-with-Vitis-AI-2-5?language=en_US, 2023. Accessed: 2023-06-23.

[31] Yann LeCun and Corinna Cortes. The mnist database of handwritten digits. 2005.

[32] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, 1989. doi: 10.1162/neco.1989.1.4.541.

[33] Kunihiko Fukushima. Visual feature extraction by a multilayered network of analog threshold elements. *IEEE Transactions on Systems Science and Cybernetics*, 5(4):322–333, 1969. doi: 10.1109/TSSC.1969.300225.

[34] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift, 2015.

[35] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[36] Alexander LeNail. Nn-svg: Publication-ready neural network architecture schematics. *Journal of Open Source Software*, 4(33):747, 2019. doi: 10.21105/joss.00747. URL https://doi.org/10.21105/joss.00747.

[37] Xilinx. *DPUCZDX8G for Zynq UltraScale+ MPSoCs Product Guide (PG338) version 4.1*. Xilinx, San Jose, CA, Year.

[38] Xilinx. Kv260 starter kit documentation, 2023. URL https://docs.xilinx.com/r/en-US/ds986-kv260-starter-kit. Accessed: 2023-07-10.

[39] Xilinx. Fix DPU of benchmark-b4096. https://github.com/Xilinx/kria-apps-firmware/commit/7dd3ea53446102080e3affa2149c2b5bf752012d, 22 September 2022.

[40] D. H. Jakubassa-Amundsen. Advances in bremsstrahlung: a review, 2021.

[41] Yung-Su Tsai. Erratum: Pair production and bremsstrahlung of charged leptons. *Rev. Mod. Phys.*, 49:421–423, Apr 1977. doi: 10.1103/RevModPhys.49.421. URL https://link.aps.org/doi/10.1103/RevModPhys.49.421.

[42] Bruno Rossi and William B. Fretter. High-Energy Particles. *American Journal of Physics*, 21(3):236–236, 03 1953. ISSN 0002-9505. doi: 10.1119/1.1933408. URL https://doi.org/10.1119/1.1933408.

[43] H. Bethe. Zur theorie des durchgangs schneller korpuskularstrahlen durch materie. *Annalen der Physik*, 397 (3):325–400, 1930. doi: https://doi.org/10.1002/andp.19303970303. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/andp.19303970303.

[44] R. L. Workman et al. Review of Particle Physics. *PTEP*, 2022:083C01, 2022. doi: 10.1093/ptep/ptac097.

[45] S. Spannagel, K. Wolters, D. Hynds, N. Alipour Tehrani, M. Benoit, D. Dannheim, N. Gauvin, A. Nürnberg, P. Schütze, and M. Vicente. Allpix2: A modular simulation framework for silicon detectors. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 901:164–172, sep 2018. doi: 10.1016/j.nima.2018.06.020. URL https://doi.org/10.1016%2Fj.nima.2018.06.020.

[46] S. Agostinelli et al. Geant4—a simulation toolkit. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 506(3):250–303, 2003. ISSN 0168-9002. doi: https://doi.org/10.1016/S0168-9002(03)01368-8. URL https://www.sciencedirect.com/science/article/pii/S0168900203013688.

[47] A. Bagulya et al. Recent progress of geant4 electromagnetic physics for lhc and other applications. *Journal of Physics: Conference Series*, 898(4):042032, oct 2017. doi: 10.1088/1742-6596/898/4/042032. URL https://dx.doi.org/10.1088/1742-6596/898/4/042032.

[48] D. Wright and M. Kelsey. The geant4 bertini cascade. *Nuclear Instruments and Methods in Physics Research Section A Accelerators Spectrometers Detectors and Associated Equipment*, 804:175–188, 12 2015. doi: 10.1016/j.nima.2015.09.058.

[49] Omrane Kadri, V. Ivanchenko, F. Gharbi, and Adel Trabelsi. Incorporation of the goudsmit–saunderson electron transport theory in the geant4 monte carlo code. *Nuclear Instruments and Methods in Physics Research Section B Beam Interactions with Materials and Atoms*, pages 3624–3632, 03 2013. doi: 10.1016/j.nimb.2009.09.015.

[50] Andreas Hoecker, Peter Speckmayer, Joerg Stelzer, Jan Therhaag, Eckhard von Toerne, and Helge Voss. TMVA: Toolkit for Multivariate Data Analysis. *PoS*, ACAT:040, 2007.

[51] Maurice Coquet, Xiaojian Du, Jean-Yves Ollitrault, Sören Schlichting, and Michael Winn. Intermediate mass dileptons as pre-equilibrium probes in heavy ion collisions. *Physics Letters B*, 821:136626, oct 2021. doi: 10.1016/j.physletb.2021.136626. URL https://doi.org/10.1016%2Fj.physletb.2021.136626.

[52] ALICE Collaboration. Anisotropic flow and flow fluctuations of identified hadrons in pb−pb collisions at $\sqrt{s_{\mathrm{NN}}}$ = 5.02 tev, 2022.

[53] Jun Xu and Che Ming Ko. Higher-order anisotropic flows and dihadron correlations in pb-pb collisions at $\sqrt{s_{NN}}$ = 2.76 tev in a multiphase transport model. *Phys. Rev. C*, 84:044907, Oct 2011. doi: 10.1103/PhysRevC.84.044907. URL https://link.aps.org/doi/10.1103/PhysRevC.84.044907.