SAPIENZA
UNIVERSITÀ DI ROMA

Novel algorithms based on AI acceleration for the High Level Trigger of the ATLAS experiment

Facoltà di Scienze Matematiche Fisiche e Naturali

Corso di Laurea Magistrale in Particle and Astroparticle Physics

Candidate

Lucrezia Rambelli

ID number 1762610

Thesis Advisor

Prof. Stefano Giagu

Academic Year 2020/2021

CERN-THESIS-2021-379
//2021

Benedetti siano gli istanti
i millimetri
e le ombre delle piccole cose.

Fernando Pessoa

Contents

Introduction

The Large Hadron Collider (LHC) [17] at CERN is designed to achieve center of mass energy values of 14 TeV and is the largest and most powerful particle accelerator in the world. Along its circumference, 27km long, detectors are placed for the revelation and study of the events produced by the proton-proton interaction.

The ATLAS experiment [1] is one of them, it is a multi-purpose detector that has achieved many successful experimental results and that will now be updated to face with the new LHC experimental conditions during the high luminosity phase (HL-LHC) [25].

One of the purposes of the high luminosity phase, in which the number of events produced will be much higher than the current value, is the search for new physics, i.e. particles or events predicted by beyond Standard Model (bSM) models that can explain what SM does not.

As an even higher number of events are produced, one of the fundamental necessary upgrades to be made for the HL-LHC lies in new algorithms and methodologies for events selection in the software based trigger level, the High Level Trigger (HLT), which will have to be faster and implement very selective algorithms.

The point of this work is right here, in fact the goal of this thesis project is to propose a *feasibility test* for the implementation of HLT trigger algorithms for bSM physics selection which exploit the advantages of Deep Neural Networks (DNNs) on commercial FPGA accelerator devices, studying their physical and technical performances on simulated datasets in order to then test them on a trigger slice test of the ATLAS experiment during the end of the LHC-Run3 and for the HL-LHC phase.

In fact, FPGAs are programmable processors which offer high flexibility and which result to have high performances for DNN implemented algorithms. These hardware devices will then replace the current on-detector trigger system performed by the chambers placed in the muon spectrometer [27].

In the first chapter an overview of the theoretical context is presented, introducing the Standard Model, its predictions and its fallacies, then moving on to an analysis of the physics of Long Lived Particles (LLP), which are predicted by many of the bSM theories and that we would like go to select with our trigger algorithms. In the second chapter a description of the LHC apparatus and the ATLAS experiment is reported, with an in-depth analysis on today's and future trigger systems. The third chapter is dedicated to neural networks, their basic principles and possible structures and their training. Then in the fourth chapter the dataset and Deep Neural Network based trigger algorithms are introduced, and the algorithm performances in terms of efficiency and resolution are reported. The fifth chapter is dedicated to the implementation of DNN on hardware devices, it analyzes the Vitis AI library used for the model implementation on FPGAs, and then in the sixth chapter the implementation steps are showed and also performances in terms of memory occupancy and model prediction accuracy are computed. Finally in this chapter the processing times performances for FPGA devices are estimated, comparing their values for the CPU and GPU ones.

Chapter 1

Physics at ATLAS experiment: the theoretical context.

The ATLAS (A Toroidal LHC ApparatuS) experiment is a multi purpose detector that is a part of the Large Hadron Collider (LHC) complex at CERN in Ginevra. It analyzes the products of proton-proton interactions at high energies (up to $\sqrt{s} = 13$ TeV), reaching a coverage close to the solid angle around the collision point.
The ATLAS experiment follows several research lines, such as the confirmation of the Standard Model (SM) theory, Higgs boson meseurements and searching for physics beyond the Standard Model (bSM).

1.1 The Standard Model

The Standard Model is actually the most successful theory of fundamental interactions we have. It has been developed from the 1960s and it contains a description of all the fundamental forces, except for gravity. In 2012, after various experimental confirmations, we discovered its last missing piece: an Higgs-like boson, compatible with the SM prediction. It is based on the non abelian, semi-simple gauge group $SU(3)_C \times SU(2)_I \times U(1)_Y$ whose gauge bosons are respectively the gluons $g^{\mu}_{\alpha=1,..,8}$, $W^{\mu}_{i=1,2,3}$ and B^{μ}, which linear combinations are the mediators of weak and electromagnetic interactions (W^+, W^-, Z^0, A_μ).

The color symmetry $SU(3)_C$ is the group of Quantum Chromo Dynamics (QCD), and describes the strong interactions between quarks and gluons. The symmetry group $SU(2)_I \times U(1)_Y$ embeds the weak and the electromagnetic interactions in the electroweak ones (where I is the isospin and Y is the hypercharge).
The SM matter fields are introduced in the theory as irreducibile representation of the global symmetry group, and their quantum numbers are listed in the following table

Field	$SU(3)_C$	$SU(2)_I$	$U(1)_Y$
Q^i_L	3	2	1/6
u^i_R	3	1	2/3
d^i_R	3	1	-1/3
L^i_L	1	2	-1/2
e^i_R	1	1	-1

where the index i goes from 1 to 3 and indicates that there are three different generations of

fermions with the same quantum numbers.

A Higgs scalar field is added to the theory, necessary to give mass to fermions through Yukawa coupling and to gaugue bosons through Higgs Mechanism. Under the SM symmetry group the Higgs field can be described as

$$\phi \simeq (1, 2, +1/2) \simeq \begin{pmatrix} \phi^{\dagger} \\ \phi^{0} \end{pmatrix} \tag{1.1}$$

and it is associated with an invariant gauge potential defined as

$$V(\phi) = \mu^2(\phi^{\dagger}\phi) + \lambda(\phi^{\dagger}\phi)^2 \tag{1.2}$$

where the constant λ must be positive in order to guarantee a ground state, and according to the sign of μ^2 the potential shape changes and the ground state can be unique or degenerate (figure 1.1). The Higgs mechanism which creates the symmetry breaking arise for $\mu^2 < 0$, and

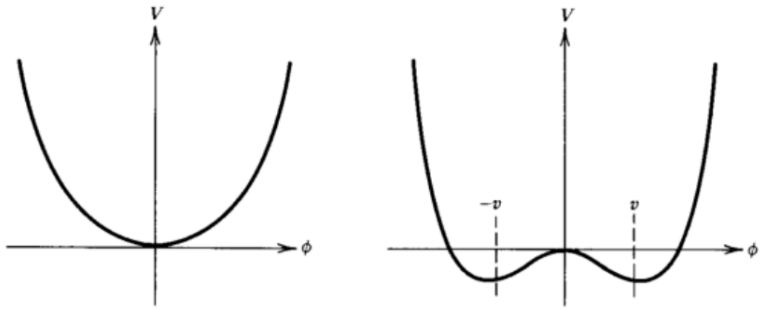

Figure 1.1: Behavior of the Higgs potential $V(\phi)$ for $\mu^2 > 0$ on the left and for $\mu^2 < 0$ on the right.

the potential then acquires a non-zero expectation value (VEV). In particular, the Higgs boson is found to acquire non-zero vev of v = 247GeV.

The Higgs gives a mass to the fermions (quarks and leptons) through Yukawa coupling terms with their fields in the fermionic Lagrangian term. In it, when the symmetry is broken and the Higgs field acquires the vev value, mass terms for the fermions dependent on the Yukawa coupling constants come out, and by diagonalizing these terms with unitary transformations, the mass eigenstates come out with the values reported in the Particle Data Group (PDG).

In particular leptons are grouped in three families, once for each flavour: electron, muon and tau, which are ordered by mass and composed by a charged lepton, which interact under electromagnetic and weak forces, and a neutral one called neutrino, which only interacts weakly.

Then in the SM there are three neutrinos, one for each lepton family, and their characteristic is that their flavour and mass eigenstates doesn't correspond, so neutrinos flavour oscillations find out.

Gauge bosons acquires mass through the Higgs mechanism, where, by expanding the Higgs field around its value in vacuum as

$$\phi = \frac{1}{\sqrt{2}} \begin{pmatrix} \pi_1 + i\pi_2 \\ v + h + i\pi_3 \end{pmatrix} \tag{1.3}$$

where v is the vev, h the Higgs radial physical field and $\phi_{i=1,2,3}$ are the Goldstone bosons. The generators of $SU(2) \times U(1)$ include in their definition the Pauli matrices and the Identity (for

U(1)), and through their linear combinations it's possible to obtain a field with respect the vacuum higgs is invariant and three massive fields (connected to the three Goldstone bosons). The physical states thus obtained are described by the following fields:

$$\begin{aligned} W^{\mu}_{\pm} &= \frac{W^{\mu}_{1} \mp iW^{\mu}_{2}}{\sqrt{2}} \\ Z^{\mu}_{0} &= cos\theta_{w}W^{\mu}_{3} - sin\theta_{w}B^{\mu} \\ A^{\mu} &= cos\theta_{w}B^{mu} + sin\theta_{w}W^{\mu}_{3} \end{aligned} \tag{1.4}$$

where θ_w is the Weinberg angle, the field A^{μ} is the one corresponding to the massless field associated to U(1) and commonly denoted as γ, and finally the fields $W^{\pm}$ and Z^0 are the massive fields responsible for the weak interactions, which, having mass through a spontaneous symmetry breaking, make the weak interaction a short-range force.

For their fields structures the electroweak $W^{\pm}$ bosons result to interact only with left-handed fermions or with right-handed anti-fermions, while the Z^0 boson interacts with all type of fermions/anti-fermions. Finally, the photon γ is the massless mediator of the electromagnetic force and interacts only with electrically charged particles.

The phenomenological property of *color confinement* is associated with particles that are not singlet under SU(3), therefore with color-charged particles (quarks and gluons), and it does not allow them to exist as free or asymptotic states but only as color-less bounded states.

The reason for this confinement is connected to the high value of the strong coupling constant at low energies, which does not allow to observe quarks and gluons as free particles but only as color-less states called hadrons and divided into mesons (composed of quarks and anti-quarks) and baryons (bound states of three quarks).

The particles theorized by the Standard Model are therefore summarized in figure 1.2

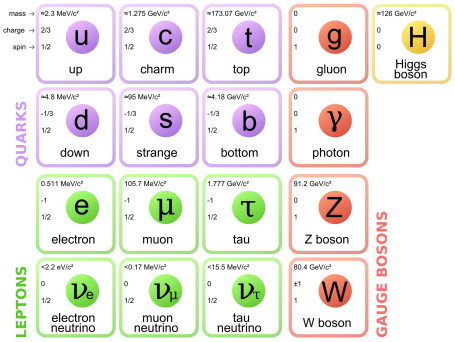

Figure 1.2: Standard Model Particles

Despite the Standard Model describes a large number of important results founded in the last decades, it is far from being complete and physicists are looking for Beyond the Standard Model (bSM) models which are able to explain the experimental evidence not foreseen by SM, to solve theoretical problems related to it (such as the hierarchy problem) and to define models that include observed experimental anomalies.

For example in the SM the Yukawa fermionic term gives mass to quarks and leptons through coupling with the Higgs field, but does not confer mass to the neutrinos. However, experimental evidence shows that at least two of the three neutrinos of the SM must have a mass (with different values), and therefore effective theories which introduce mechanisms capable of explaining the experimental evidence are required (in this case the See-Saw mechanism is one of the answers).

Another example of experimental evidence that suggests there is a need to investigate bSM is related to Dark Matter (DM), which includes 85% of the universe and which we are not yet able to explain what it is. In particular, experimental evidences such as the tangential velocity [1] and the gravitational lensing [2] show how it is necessary to look for candidates who can play the dark matter role, with the aim of being able to explain these two phenomena.

There are therefore a series of theories that includes different types of DM candidates, such as axions, right-handed neutrino, WIMP particles and others, and therefore one of the aims is to design models in which there is a portal that connects the dark sector to the one described by the Standard Model.

There are also a series of theories and models that aim to extend the SM considering it as the manifestation of physical phenomena only for certain energy scales (which define the values of the fundamental parameters that define it), such as the theory of the great unification (GUT) or SuperSymmetric theories (SUSY), able to give a theoretical explanation to the discrepancy between the radiative corrections to the mass of the Higgs boson (naturally they come out in the order of 10^17) and their experimental value.

One of the research fields for bSM physics, not by signature but which starts from the theorization of new particles and researches them experimentally, is about the Long Lived Particles [3] (LLP), that are particles with and observable lifetime which are present in most of the fundamental bSM theories, such as the SuperSymmetric or the Hidden Sector ones.

1.2 Long Lived Particles physics

The lifetime τ of a given particle is defined as the inverse of its decay width Γ, and is described as a function of the mass M of the particle itself and of the matrix element M which describes its decay integrated over the phase space $d\Pi$.

$$d\tau \simeq \frac{1}{d\Gamma} \simeq (\frac{1}{M}|\mathcal{M}|^2 d\Pi)^{-1} \tag{1.5}$$

So, a particle with long lifetime will have a very small phase space or a small matrix element.

Small values for the matrix element are given by suppressions that can arise from approximate symmetries which, if exact, would prohibit the decay. Another reason for small matrix element is the presence of coupling constant with small values, which arises from the presence of scales larger than the mass M containing operators of higher dimensions that mediates the decay (for example the μ decay is mediated by the Fermi constant G_F introduced by the presence of the W boson).

Small phase face values are instead associated to given processes where the candidate particle is produced in a decay with other two particles with similar mass values.

The SM particles have lifetime spread over a very wide range of values (figure 1.3), and SM itself contain many particles with long lifetime, such as the proton, the muon or the Higgs boson; and for this reason is reasonable that long lived particles can also appear in bSM theories.

[1] The tangential velocity of a star within a galaxy for classical mechanics should go as $v^2 = GM(r)\ dotr$, with r distance from the center of the galaxy and M (r) amount of matter within r, but experimentally we observe a trend in disagreement with the predictions showing the presence of "unseen" matter.

[2] The problem of gravitational lensing is connected to the phenomenon theorized by general relativity for which matter 'bends' the space that surrounds it. Going to measure the paths followed by photons in the universe it's therefore possible, by measuring the lensing, that is the variation of their trajectory, to measure the amount of matter of a given object. However, measurements of these paths are experimentally observed that are not in agreement with the presence of only visible matter, and could be explained by introducing Dark Matter as a contribution to the observed gravitational lensing

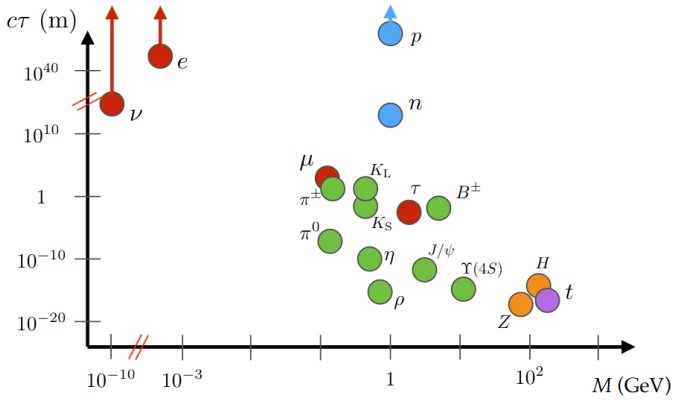

Figure 1.3: SM Particles lifetime.

Most of the LHC research for new physics assumes that it is a promptly one (with short lifetimes which decay near the interaction vertex), and a specific sector is then needed for the research of the so-called Long Lived Particles (LLP) in future LHC phases: the LHC LLP Community.

For our purposes we'll define LLPs as bSM particles with a long lifetime that give up their energy or decay into SM particles into the acceptance region of the detector. In particular, we will discuss below the search for LLPs in the ATLAS experiment.

The experimental signatures of LLPs in the detector are many and generally vary greatly from the signals coming from SM processes, allowing an analysis of the events that contain them with almost free-background, and can be seen schematically in figure 1.4.

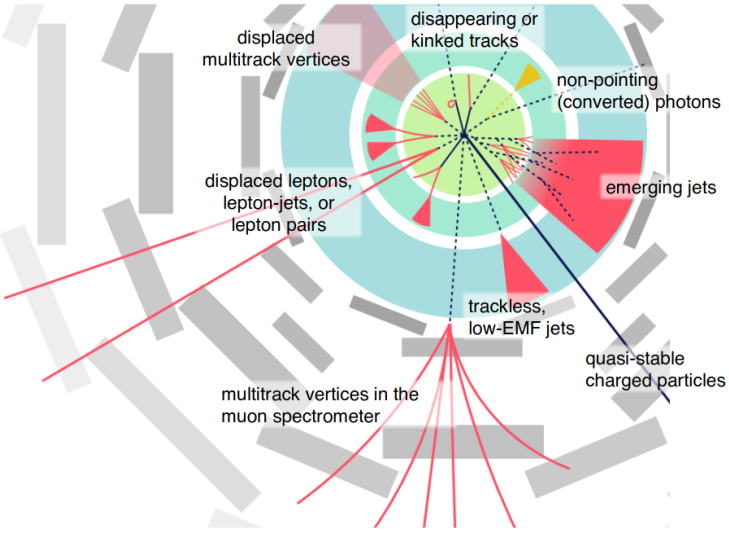

Figure 1.4: Overview of the different atypical experimental signatures that can result from bSM LLPs in the detectors at LHC [19].

For example, relevant signatures of the presence of LLPs can be localized deposits of energy released in the calorimeters, particles that decay out of time with collisions, displaced vertices in the inner tracker or in the muon spectrometer, disappearing tracks, etc..

Therefore the possibility of opening a new branch of new physics research following the LLPs research is convenient as these have well distinguishable signatures (little background) and also they are supported by many theoretical models.

In general, the experimental research of LLPs is distinguished according to the type of particle that is going to be detected.

The LLPs color-charged can be experimentally detected directly in their interactions with the detector, which form jets, the neutral ones instead, being these particles usually either too heavy or too weakly interacting, can be detected only using their decays (in a region from the Inner Tracker to the Muon Spectrometer).

The new phase planned for the LHC collider, the HL-LHC, will have many effects on the search for LLPs: for example powerful deep learning resources will be used for the implementation of event selection algorithms containing the LLPs.

Below we will explore, in detail, the research of LLPs at LHC, showing some of the simplified models that theorize LLPs by offering signatures accessible to research at LHC, making a summary of the events that are the background to the interesting ones.

Simplified Models

The importance of having simplified models that describe a given theory lies in the fact that they offer a mapping between theoretical physical quantities and experimental signatures.

The LHC LLP Community is looking for approaches to simplified model sets that ensure *powerful* experimental results, i.e. that cover almost the totality of the results of the model itself, *efficient*, i.e. that reduce possible overturns obtained from multiple searches, *flexible*, that is applicable to several different models in case the extension of a given theory is required, and *durable*, that is rarely to be reused in the future for new models .

Since the production and decay of the LLPs are in physically distinct positions, we will divide them, and the simplified models associated with them, by channels where the free parameter is the lifetime.

The existence of LLPs is predicted by several bSM theories. For example in SUSY theories they arise from the decay of a supersymmetric particle χ that, decaying in its lighter candidate χ_{LSP} (the Lightest Supersymmetric Particle, which is stable due to the R-parity conservation) and a SM particle, if the mass difference between the supersymmetric particles is small the SM particle comes out as a weakly interacting one.

LLPs are also included in Hidden Sector theories, which consider that, given the "SM sector" invariant under its well-known symmetry group, a SM particles can become, under transformations which arise from the mixing term of the SM fields with the new ones (which must be included if they result to be invariant), a "dark" particle, i.e. belonging to another sector which is described by an unknown symmetry group.

Therefore this dark particle will do, in its sector, a series of processes and will be able to transform itself again (according to a given coupling given by the mixing term) into an SM particle, then observed experimentally in the detectors. So, if the coupling is small, the transition probability will also be small and LLPs particles will arise.

LLPs can be produced in different several processes, each associated with a specific experimental signature useful for its selection. As can be seen in figure 1.5 they can be produced in Direct Pair Production processes (DPP); Heavy Parent processes (HP), where LLPs are decay products of heavier particles produced in the primary pp vertex and are produced with a SM particle; in Higgs processes (HIG), where they arise from the coupling with the SM higgs and can be detected searching for jets or leptons characteristic of the VBF and VH productions;

Heavy Resonance processes (RES), when LLPs are produced by on-shell resonance decays arising from the primary vertex (the off-shell decays are similar to the DPP production processes), and finally in Charge Current processes (CC), where, in right-handed neutrino theories of the electroweak scale, the LLPs are produced by the lepton decay of a W resulting in the detector as prompt charged leptons.

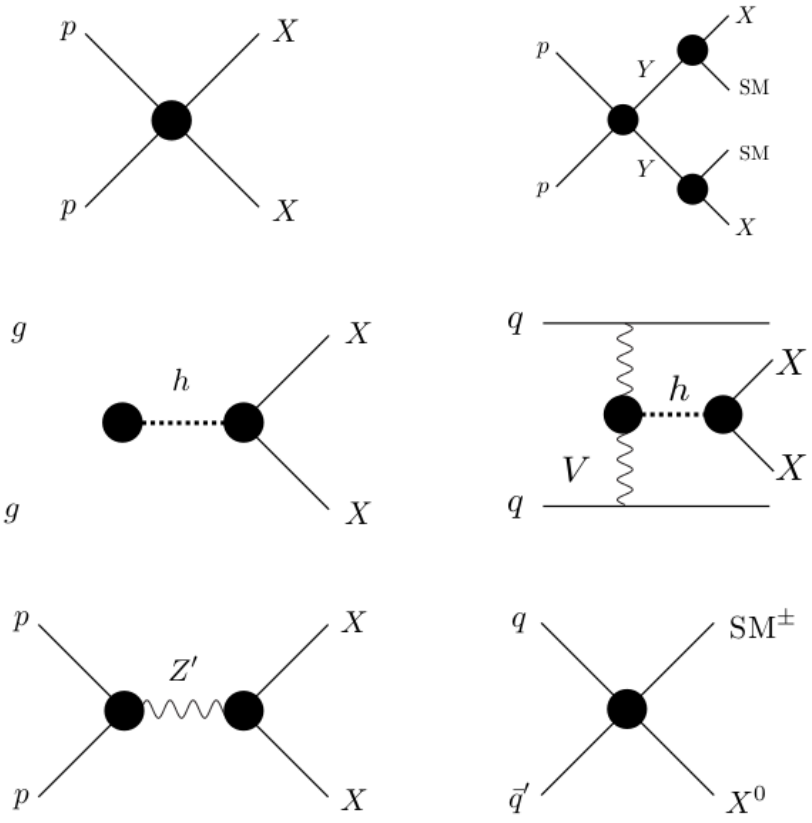

Figure 1.5: LLPs production mechanisms. Starting from the first line and continuing to the right we have: direct pair production (DPP) heavy parent (HP), Higgs modes (HIG) (including production for VBF and production for VH, not shown here), heavy resonance (RES) and charged current (CC)

Chapter 2

ATLAS experiment at the LHC

2.1 LHC

The Large Hadron Collider (LHC) [17] is a hadronic collider located at CERN placed between France and Switzerland and located inside a 27 km long tunnel at a depth between 50 m and 175 m. Inside the collider two beams of protons divided into bunches enter with a frequency of 40 MHz and are accelerated in two distinct vacuum tubes every 25 ns.

The protons that are made to collide are obtained starting from hydrogen atoms (gaseous) from which the electrons are removed through an electric discharge. These are then injected into the first stage of the acceleration chain, composed of a LINAC, and after passing through various intermediate stages they are inserted into one of the two equal and concentric rings of LHC.

For protons acceleration are used superconducting magnets in order to minimize energy losses; for using magnets in a superconductive state it is necessary to cool them to a temperature of -271.3°C, therefore a large part of the accelerator is connected to a distribution system of liquid Helium, which acts as a refrigerant.

Inside the ring there are mainly two types of magnets: the dipoles, used to curve the trajectory of the proton beams, keeping it almost circular, and the quadrupoles, used to focus the beam on the transverse plane.

So, the beams produced and accelerated in LHC are manipulated in the full path to maximise the collision rate to a value of 2556 proton filled bunches during the Run-2 (out of the maximum allowed of 2808), which works at 13 TeV.

One of the most parameter for a collider is the *luminosity*, which represents the number of collisions that can be produced in a given collider per cm^2 and per second. It is defined by the instantaneous luminosity, which depends only by the machine characteristics and not by a specific physics processes. In fact instantaneous luminosity is proportional to the total number of colliding particles (number of bunches n_b for particles contained in each bunch N_b), and to the bunch revolution frequency f_rev, and is inversely proportional to the beam transversal area (by σ_x and σ_y which are the root mean square of the beam width along x and y directions). Instantaneous luminosity can in fact be expressed as

$$L = \frac{N_b^2 n_b f_{rev} \gamma}{4\pi\sigma_x\sigma_y} F \tag{2.1}$$

where γ is the Lorentz factor and F is the correcting geometrical form factor which takes into account the fact that the beams are not completely collimated but they collide at a certain angle.

Then the total luminosity, which is a fundamental for the evaluation of the amount of datas delivered by LHC and recorded by the experiment, is defined over a time period and called integrated luminosity for the fact that it is defined as

$$L_{int} = \int L dt. \tag{2.2}$$

Typical integrated luminosity values for LHC during the Run-2 are 156 fb^{-1} delivered data of pp collision of which $147 fb^{-1}$ are the ones recorded by the ATLAS experiment. In the figure 2.1 is it possible to observe how the total integrated luminosity changes in time, and it's possible to see its values for LHC delivered data, ATLAS recorded ones and the amount of certified good quality data acquired during the LHC Run-2 at $\sqrt{s} = 13 TeV$. A summary of the selected LHC

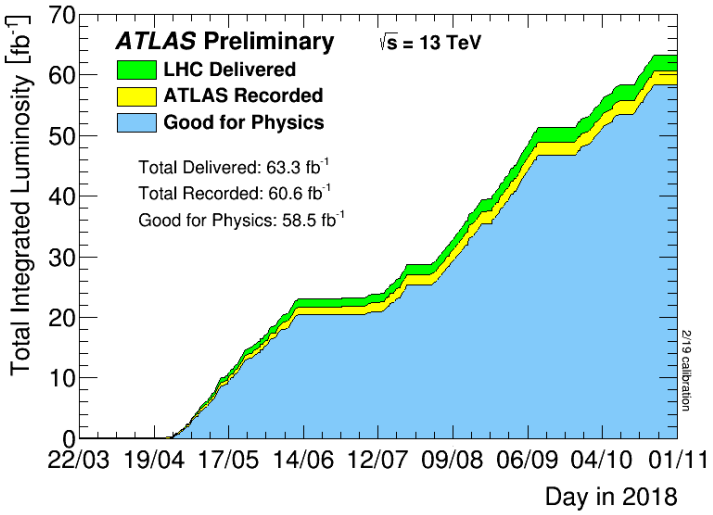

Figure 2.1: Integrated luminosity versus time delivered to and recorded by ATLAS and the amount of good quality data luminosity during 2018 [8].

parameters for pp collision at $\sqrt{s} = 13$ TeV in 2015-2018, which represent the LHC machine performance during Run-2, are presented in the following figure

Parameter	2015	2016	2017	2018
Maximum number of colliding bunch pairs (n_b)	2232	2208	2544/1909	2544
Bunch spacing (ns)	25	25	25/8b4e	25
Typical bunch population (10^{11} protons)	1.1	1.1	1.1/1.2	1.1
β^* (m)	0.8	0.4	0.3	0.3–0.25
Peak luminosity $\mathcal{L}_{\text{peak}}$ ($10^{33}\,\text{cm}^{-2}\text{s}^{-1}$)	5	13	16	19
Peak number of inelastic interactions/crossing ($\langle\mu\rangle$)	~ 16	~ 41	$\sim 45/60$	~ 55
Luminosity-weighted mean inelastic interactions/crossing	13	25	38	36
Total delivered integrated luminosity (fb^{-1})	4.0	38.5	50.2	63.4

Figure 2.2: Selected LHC parameters during Run-2, the values are representative of the best accelerator performance during physics operation [8].

Along the LHC ring are placed five different experiments: ATLAS, CMS, ALICE and LHCb, located as shown in figure 2.3.

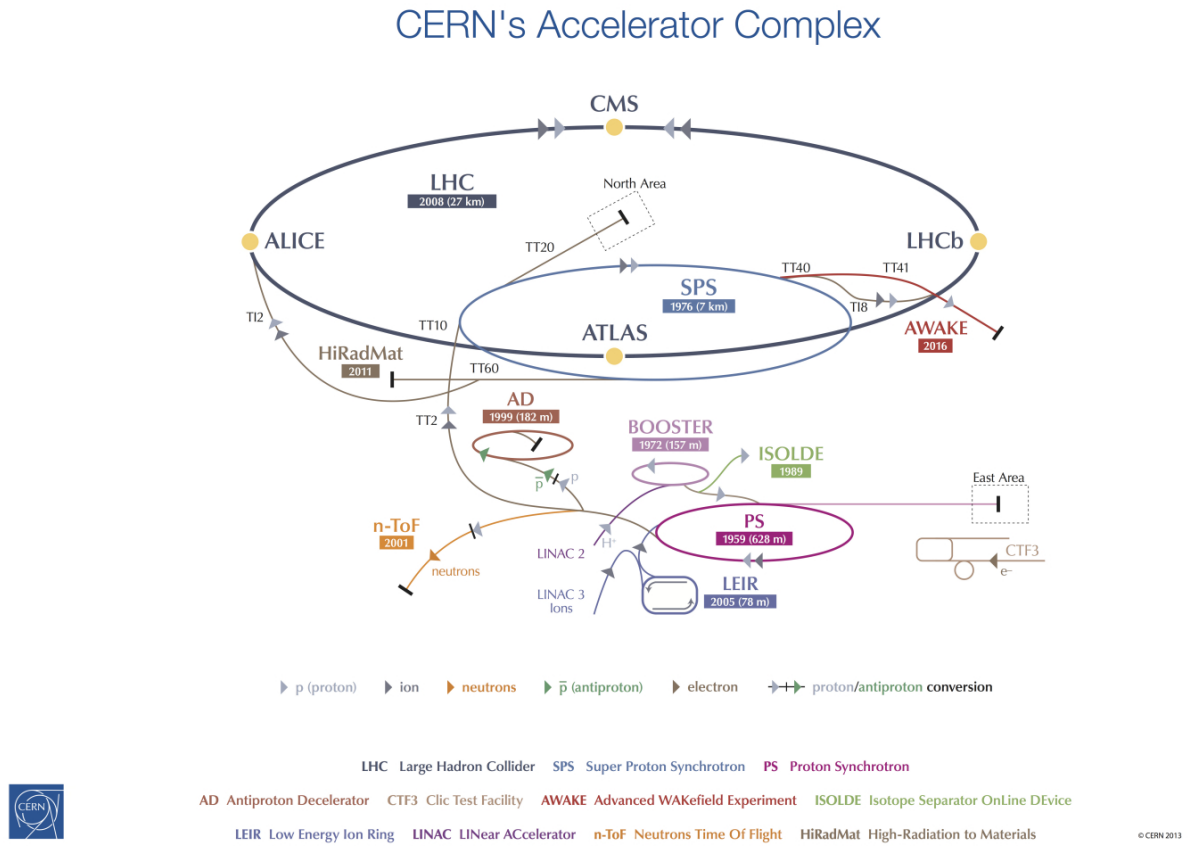

Figure 2.3: LHC Complex

Below we will analyze in detail only the ATLAS experiment, around which this thesis work revolves.

2.2 ATLAS experiment

A Toroidal LHC ApparatuS (ATLAS) is one of the four experiments through which the products of proton-proton interactions are studied in the LHC. The ATLAS experiment has a cylindrical structure 46 meters long with a diameter of 25 meters, and the detector is symmetrically positioned around the interaction point.

The structure of the ATLAS experiment consists of a series of concentric layers each consisting of a specific detector for certain physical quantities and particles. Starting from the primary interaction vertex there are: the inner detector (ID), the electromagnetic calorimeter (ECAL), the hadronic calorimeter (HCAL) and the muon spectrometer (MS). A trigger system and a data acquisition system are also associated with the entire experimental apparatus [1].

Below, after introducing a coordinate system useful for the description of the components, we will analyze individually the detector layers that make up the ATLAS experiment.

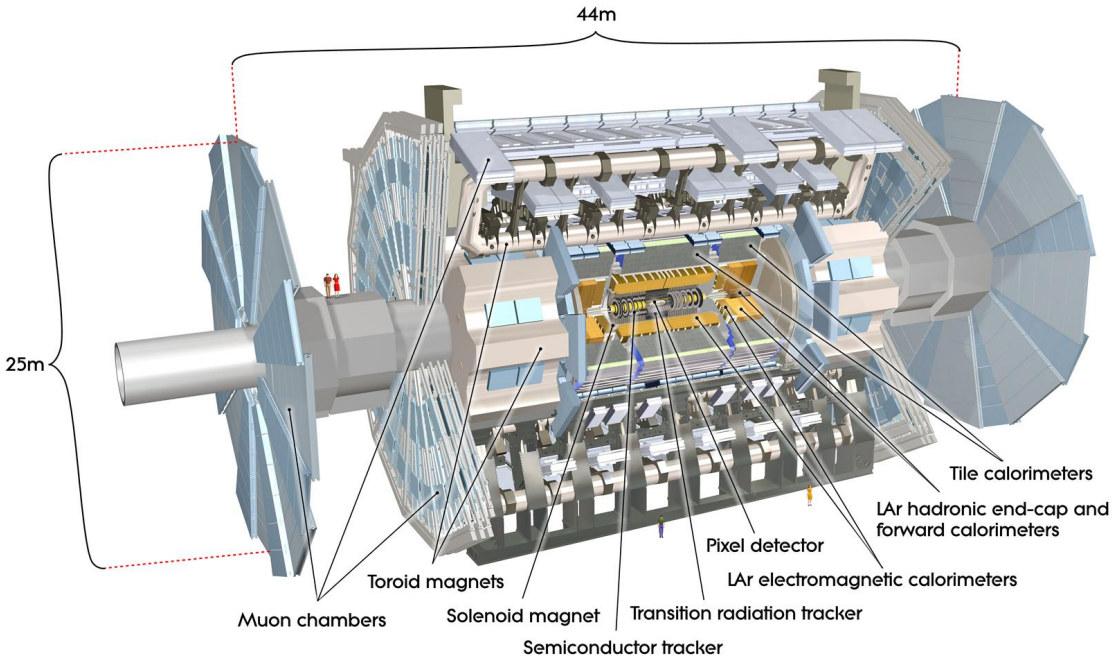

Figure 2.4: ATLAS experiment [1]

2.2.1 ALTAS coordinate system

The ATLAS detector uses two different coordinate systems, both having as their origin the point of interaction of the proton beams:

1. A Cartesian reference system (x,y,z), where the z axis coincides with the beam axis (which coincides with the symmetry axis of the apparatus), the y axis is directed upwards and the axis x horizontally towards the center of the LHC ring.
   This reference system is particularly useful since, being LHC a hadronic collider and therefore not knowing exactly the energy of the center of mass (the interactions between hadron quarks don't allow to estimate a precise center of mass energy value because their momentum is not known in a precise way but as a distribution), we use the momentum in the transverse plane (x, y), p_T which is a Lorentz invariant for transformations along z.

2. A polar reference system (ϕ, θ), where ϕ is the azimuthal angle of rotation around the z axis and θ is the polar angle of rotation around the y axis, which originates on the z axis and has positive values by rotating counterclockwise with respect to the (z, y) plane, as shown in figure 2.5

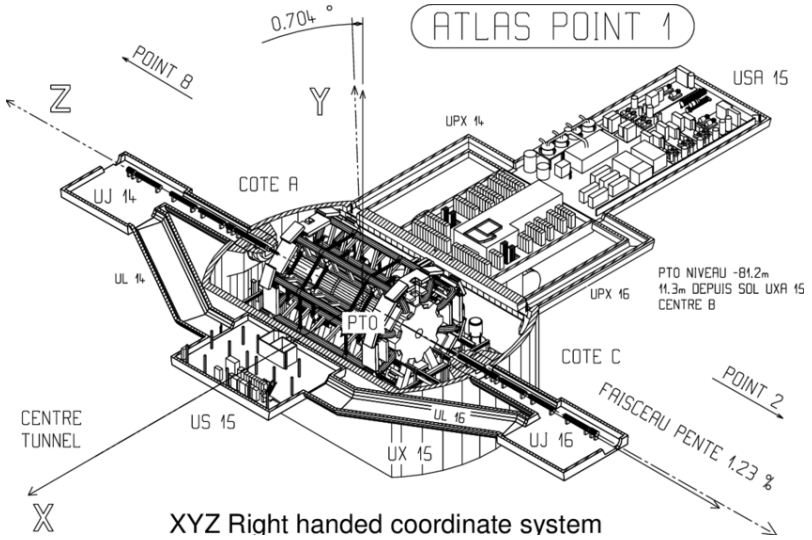

Figure 2.5: Coordinate systems in ATLAS experiment

In the ATLAS experiment, the pseudorapidity η is used instead of the angular coordinate θ, and is defined as

$$\eta = -ln(tan(\frac{\theta}{2})) \tag{2.3}$$

that is therefore 0 for $\theta = \frac{\pi}{2}$ and that goes to infinity for $\theta \to 0$ (figure 2.6). The pseurodapidity turns out to be an advantageous variable as it is a Lorentz invariant.

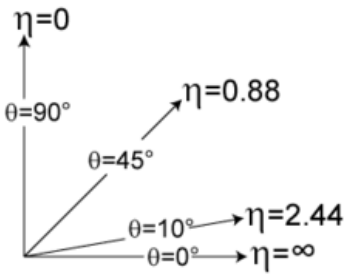

Figure 2.6: Pseudorapidity plot.

According to the pseudorapidity, two subdivision regions of the ATLAS experiment can therefore be defined:

- **Barrel region** ($|\eta| < 1.05$), which represents the central part of the detector close to the interaction point
- **Endcap regions** ($1.05 < |\eta| < 2.7$), which includes the two parts of the experiment located at both ends and which behaves as a *cover* in order to not lose the events produced that don't pass through the barrel region.

2.2.2 Inner Detector

The inner detector (ID) covers the region closest to the interaction vertex,
it has a cylindrical structure and has the task of measuring the momentum, the charge and the trajectory of the particles produced in the primary interaction vertex, covering in pseudorapidity 'a region of about $|\eta| < 2.7$ beyond which the information coming from it about the events is lost and only those coming from the HCAL are available.

For this the ID is placed in a magnetic field of 2T parallel to the z axis and the trajectory of the particles inside it is curved allowing the measurement of the quantities indicated above.

It is composed of three sub-layers of detectors based on different technologies, which extend both in the barrel and in the endcap regions and which allow to reach high values of granularity in the measurements.

- **The Pixel Detector**, which uses silicon pixels to achieve high granularity near the interaction vertex, which is of fundamental importance to collect sets of high precision measurements as close as possible to the primary vertex. It is composed of a system of four layers in the barrel (placed at radii of about 4cm, 11cm and 14cm) and three discs for each endcap (with radii between 11cm and 20cm), which overall complete the angular coverage of the detector, each consisting of pixels with a size about $50\mu m$ x $400\mu m$ fragmented along η, ϕ, z.
  The Pixel Detector is designed to return three precision measurements for each track, thus allowing to determine the resolution on the impact parameter.

- **The Semiconductor Tracker** (SCT), based on silicon strip technology, it is composed of eight layers of strips in the barrel and nine for each endcaps, for a total surface area of about $60m^2$ of silicon with about 6.2 million readout channels. It is designed to return four precision measurements for each track, useful for the reconstruction of the impact parameter and the momentum.

- **The Transition Radiation Tracker** (TRT), based on straw chambers technology, it exploits the ionization of the charged particles of the Xenon mixture that fills the 4mm diameter and 150cm long tubes of which it is made. Having such a small diameter, the TRT can work on high rates of particles and is able to identify electrons thanks to the tracks left by the photons of radiation produced by the electrons themselves in the radiator between the straws. The barrel is made up of about 50000 straws, each divided into two parts to facilitate the readout; while in the endcaps there are 320000 radial straws, each with its own readout system. The TRT allows to acquire a set of about 36 "point-like" measurements for each trace and to discriminate between electrons and hadrons by setting two thresholds for each straw and considering that the photons emitted by the electrons will be the only particles to exceed the highest.

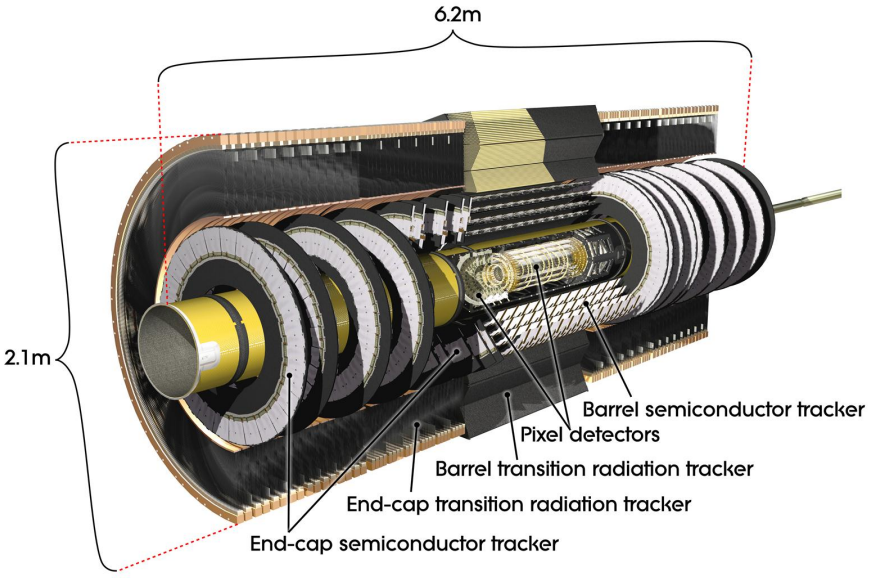

Figure 2.7: Three-dimensional view of the Inner Detector of the ATLAS experiment

A dynamic alignment algorithm is associated with the Internal Tracker [5] as it must provide extremely precise measurements. In fact, the misalignments of the chambers due to effects such

as the fluctuation in temperatures or the variation of the intensity of the magnetic field applied can lead to a non-stationarity of the elements that make up the detector and to a subsequent degradation of the resolution and accuracy of the measurements of the tracks.

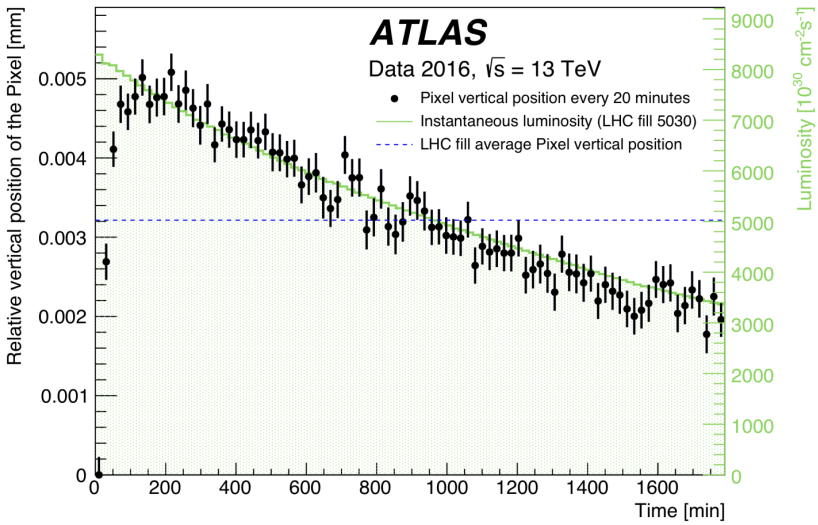

Figure 2.8: Variation of the relative vertical position of the Pixel Detector with the variation of the time and then of the LHC luminosity.

For example, in the figure 2.8 is it possible to observe the variation of the relative vertical position of the Pixel Detector with the variation of the collision intensity (which decreases during the proton filling in the LHC ring) .

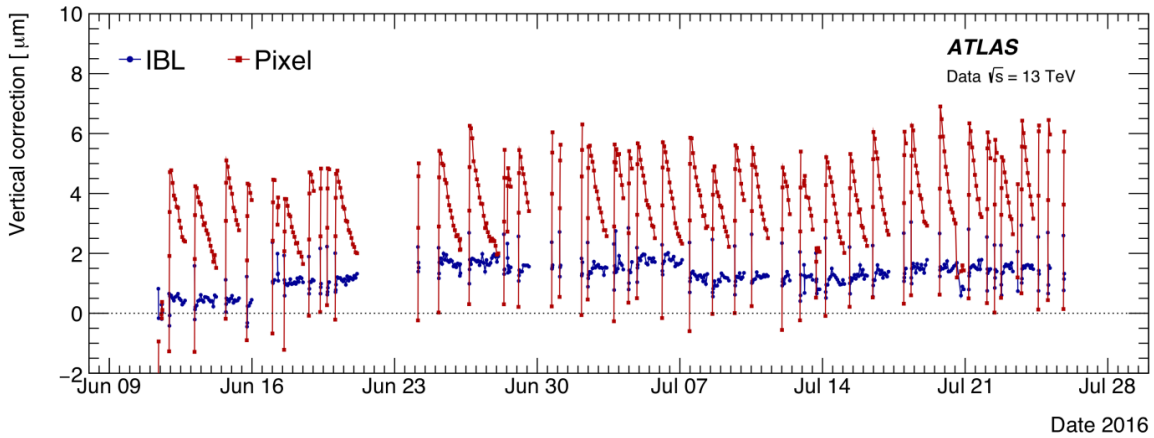

Figure 2.9: Correction to the vertical position of the Pixel Detector carried out by the dynamic correction algorithm recorded between June and July 2016.

In figure 2.9 it is possible to observe the dynamic correction made with the alignment algorithm of ATLAS over time. It calibrates the recorded data using alignment constants which are collected approximately every 20 minutes during the first hour of data-taking and after 100 minutes during the rest of the fill.

2.2.3 Calorimeters

The Inner Tracker is surrounded by two layers of calorimeters, each specific for the identification of a certain type of particles and which overall cover a pseudorapidity region of $|\eta| < 4.9$. Both the calorimeters are a sampling one, i.e. composed of alternating layers of passive absorber and activator used for the detection. In both of them the active material layer is liquid Argon, while the passive absorber layer is composed of different materials in the two calorimeters.

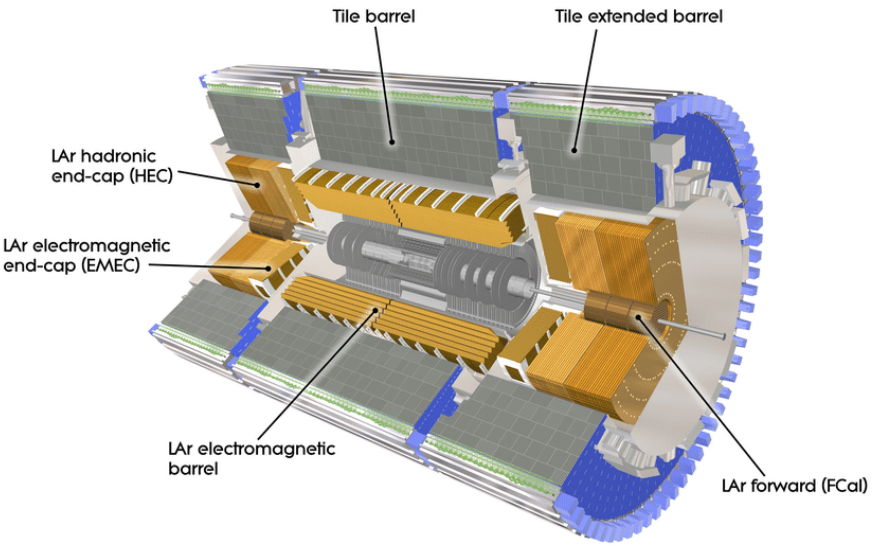

Figure 2.10: ATLAS Calorimeter System.

Electromagnetic Calorimeter (ECAL)

The electromagnetic calorimeter measures the energy of particles that are mainly affected by electromagnetic interaction, i.e. electrons and photons. When a photon or an electron produced by the primary vertex interacts with the material of the ECAL it initiates the production process of the electromagnetic shower, which mainly consists in the alternation of two sub-processes: the creation of pairs e^+e^- by very energetic photons and the emission of a photon for Bremsstrahlung by a produced electron/positron. The energy of the particle that generates the shower is then divided between its final components, and the creation of the shower is interrupted when the energy of the last particles produced is less than the critical energy at which the ionization process is favored compared to the Bremsstrahlung one and after which the particles of the shower quickly lose energy in the calorimeter.
The energy information is obtained from the amplitude of the electrical signal generated when the calorimeter completely absorbs the energy of the shower (returning for the electrons an energy value $E \sim pc$ where p is measured momentum from the tracker).
The ECAL passive layer is composed of Lead (Z = 82) and it covers a pseudorapidity region of $|\eta| < 1.45$. Its structure is such as to cover the entire development of the electromagnetic shower and is therefore defined by its properties. In particular, given an electromagnetic shower, the *radiation length* X_0 is defined as the distance that a particle has to travel in the medium such

that its energy is reduced by a factor of 1/e. It is in fact defined by the relationship

$$E(x) = E_0^{-\frac{x}{X_0}} \tag{2.4}$$

where E_0 is the initial energy of the particle, x is the distance traveled in the material and E(x) is the energy at a given distance x.
On the basis of this definition, the ECAL calorimeter of the ATLAS experiment was built with dimensions about $23X_0$ [22].

Hadron Calorimeter (HCAL)

The hadron calorimeter measures the energy of charged and neutral particles interacting by strong interaction, ie hadrons (protons, neutrons, pions, etc.). When hadrons pass through a material they interact with nuclei producing charged or neutral pions. The charged pions give rise to electromagnetic showers, the neutral ones interact with the nuclei producing new hadrons and can therefore lead to the subsequent formation of new hadron showers, or they decay into two photons which can give rise to new electromagnetic showers. The production process of hadronic swarms is therefore more complex and includes both an electromagnetic and a hadronic component.
The reference quantity describing the size of a hadronic shower is the nuclear interaction length λ_I which is greater than X_0 and defined through the same relationship. In order to contain the entire hadronic shower the HCAL is built with a size of $11\lambda_I$ [22].
The reason why HCAL is located externally to ECAL because when hadrons pass through the ECAL they lose a small and negligible loss of energy.
Also in this case the energy is measured starting from the amplitude of the electrical signal generated by the complete absorption of the particles by the calorimeter itself, but the structure of HCAL is divided into three different calorimeters, each of which uses a different material for the passive layers of the sampling calorimeter.
From the layer closest to ECAL we find, in order, the Tile Calorimeter, composed of Steel and scintillation plates, the LAr Hadronic End-Cap Calorimeter (HEC), which is always a sampling calorimeter composed of layers of Copper absorber and liquid Argon, and the LAr Forward Calorimeter (FCal), placed in the endcaps and composed of layers of Copper alternating with pairs of Tungsten layers, which allows for simultaneous measurements of hadrons and electromagnetic component energy.
It is important to notice that the charged particles identified by the calorimeter, which works better in high-energy measurements, have been previously studied by the Inner Tracker, which works better at lower energies as the curvature of a particle with less energy is greater. For neutral particles, on the other hand, calorimeters are the only source of information.

2.2.4 Magnets system

By introducing a variable r such that $r^2{=}x^2 + y^2$ we can describe the magnet system present in ATLAS.

The ATLAS detector has two distinct magnetic systems: a solenoid in the innermost barrel

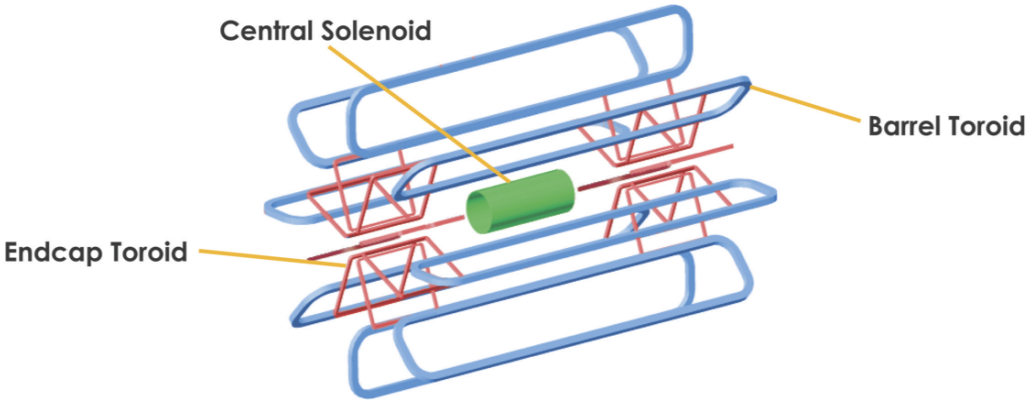

Figure 2.11: Schematic view of the ATLAS magnet system

region, which generates a magnetic field parallel to the beam that bends the charged particles in the (x, y) plane perpendicular to the beam, and a toroidal magnet system placed in the barrel and in the endcaps, which curves the trajectory of the charged particles in the (z, r) plane, used in the outermost part for muons detection. In particular the toroidal system is composed by 8 barrel toroid coils and 16 coils in the endcap ones (there are two endcap for each barrel). In figure 2.12 you can see how the magnet system of the ATLAS experiment curves the trajectories of the various particles.

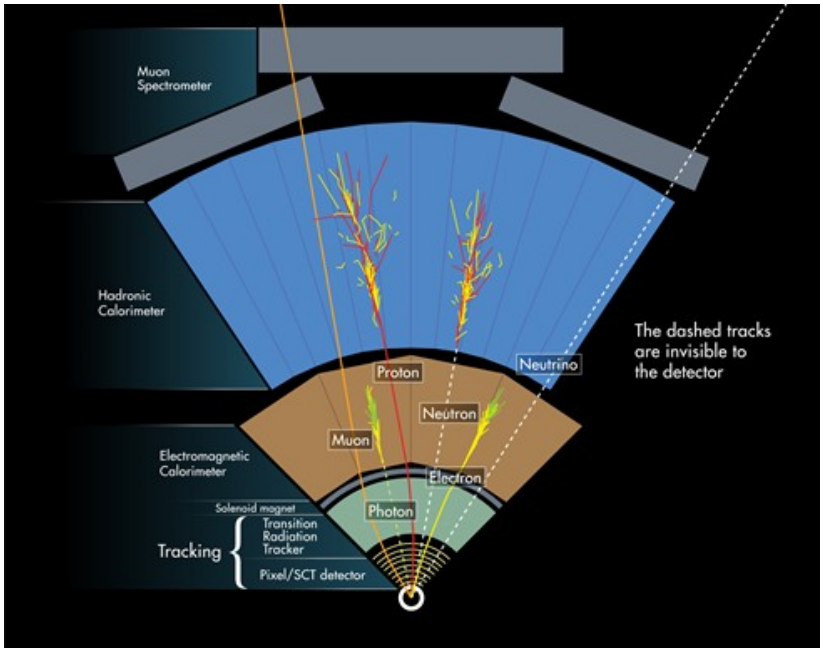

Figure 2.12: Curvature of different particles in the ATLAS experiment

2.2.5 Muon Spectrometer

The outermost layer of the ATLAS experiment is composed by a muon spectrometer, which studies this type of particles that are able to overcome all the previous layers without losing an appreciable amount of energy and which turn out to be the only SM particles capable to traverse the entire detector (plus neutrinos which are not detected at all).
By releasing only little energy in the calorimeters, muons can only be measured in a tracking system: the ATLAS experiment aims to improve the reconstruction of their transverse momentum for high values (not well measurable by the Inner Tracker).
In order to measure the momentum of high-energy muons with good precision, the muon spectrometer must have a lever arm L (equal to the distance between the first and last muon chamber) greater than the innermost detectors, this because the measure of the momentum is given by the following relation (setting c=1):

$$p = 0.3qBR \tag{2.5}$$

with R radius of curvature of the trace, B intensity of the magnetic field, q charge of the particle (in units of elementary charge).
The muon spectrometer of the ATLAS experiment consists of two systems: a series of detectors used to measure with high precision the position of the muon on the plane in which it is deflected and thus reconstruct its momentum, and a series of trigger chambers in charge of make a quick selection of events.
In particular, the detectors used for measuring the position (and therefore the momentum) are called tracking chambers, they are of two types: the MDTs, which cover the barrel region for a coverage in pseudorapidity of $|\eta| < 2$; and CSCs, used for tracking in the endcaps and covering a region of $2.0 < \eta < 2.7$. The components used instead for the trigger are the RPCs, which in the barrel region cover a range of $0.0 < \eta < 1.0$; and the TGC, which in the endcap instead cover a region in pseudorapidity equal to $1.0 < \eta < 2.4$ [16].

Type	Purpose	location	η coverage	Channels
MDT	Tracking	barrel+endcap	$0.0 < \eta < 2.7$	354k
CSC	Tracking	endcap layer 1	$2.0 < \eta < 2.7$	30.7k
RPC	Trigger	barrel	$0.0 < \eta < 1.0$	373k
TGC	Trigger	endcap	$1.0 < \eta < 2.4$	318k

Figure 2.13: Summary of the components of the ATLAS Muon Spectrometer

This system is associated with a system of magnets composed, for barrel and endcap, of 3 air-core toroids, each with 8 coils of dimensions $25m \times 7m$ in the barrel and $9m \times 4m$ of the endcaps, for a total overall magnetic field inside each coil of about 1T which deflects the muon trajectory as shown in figure 2.12. As in the case of the Inner Tracker, to the Muon Spectrometer is associated an alignment system, consisting of about 12k sensors, which allows precision impulse measurements and 3D reconstructions of the positions in the chambers (precision of about $50\mu m$) .

The muon spectrometer therefore allows a measurement of the sagitta of the tracks produced by the muons in the three layers of MDT with a resolution of about $500\mu m$ and a total measurement of the coordinate in the bending plane with a resolution of about 60-70μm.

The trigger chambers, located at the ends of the central MDT layer, select events based on the muon momentum and the coincidence with the muon bunch-crossing time, providing

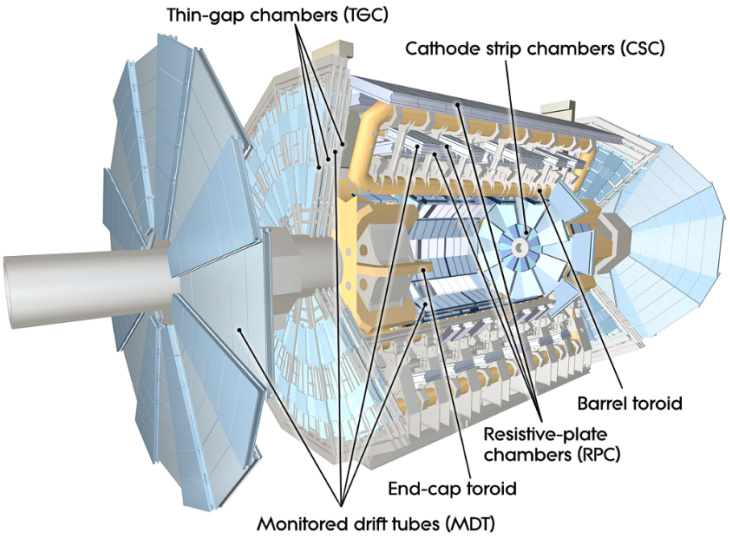

Figure 2.14: Layout of the ATLAS Muon Spectrometer

a second measure of the coordinate in the muon system perpendicular to the beam axis (not bended) with an accuracy of 5-10cm.
The various components of the Muon Spectrometer of the ATLAS experiment will be analyzed individually. Its location in the experiment can be shown schematically in figure 2.14 and, in the transverse and longitudinal planes respectively in figure 2.15 and in figure 2.16.

Monitored Drift Tubes (MDT)

The Monitored Drift Tubes (MDT) are gas detectors based on the measurement of the drift time of the electrons inside them. More precisely, they are cylindrical-shaped drift chambers with a diameter of $\simeq$ 30mm filled with Ar/CO_2 gas [13].
They cover a region in pseudorapidity $|\eta| < 2$ and overall these chambers, composed of a series of layers of drift tubes operating at an absolute pressure of 3 bar, reach an average spatial resolution of about $80\mu m$ for each tube.

When a charged muon passes inside the tubes it ionizes the gas molecules creating electron-ion pairs which are collected respectively on a Tungsten wire placed in the center of the tubes (with a diameter of 50 mum and placed at potential of 3080V) and on the tubes' walls. The amplitude of the signal in the tubes is given by the avalanche generated by the electrons, and in general this type of detectors work very slowly because they depend on the drift speed of the charges in the gas ($\simeq$2cm/microsecond).

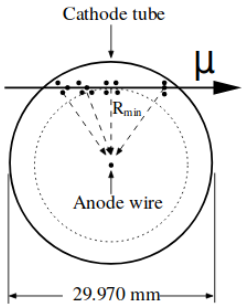

Figure 2.17: Cross-section of a MDT tube

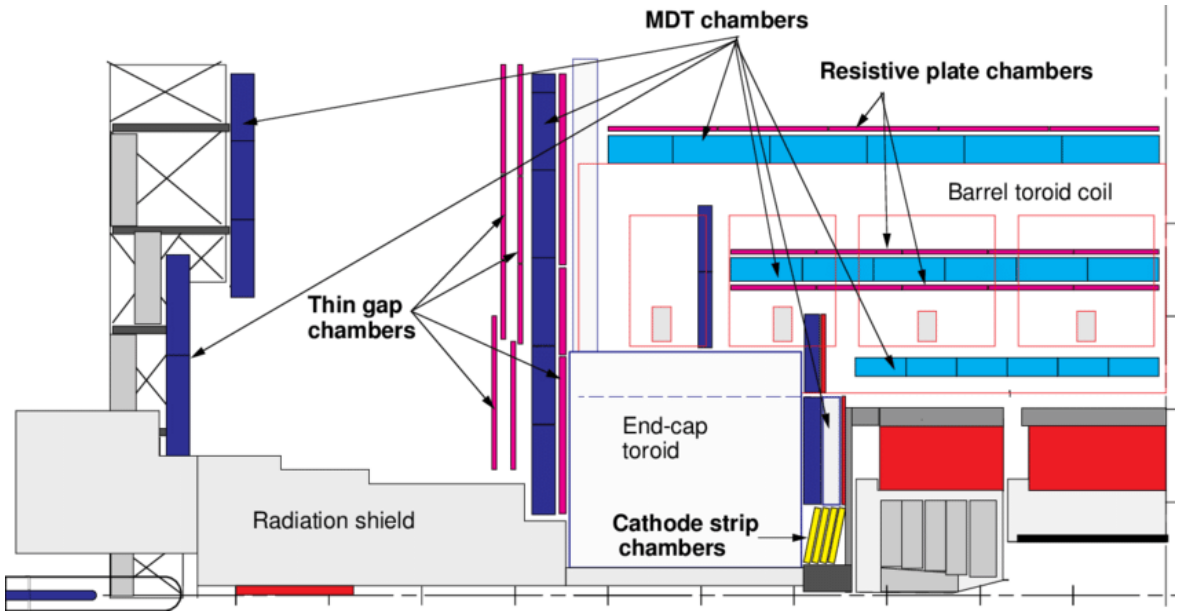

Figure 2.15: View of the ATLAS MS layout in the (z,y) plane for a small azimuthal sector containing the barrel toroid coils.

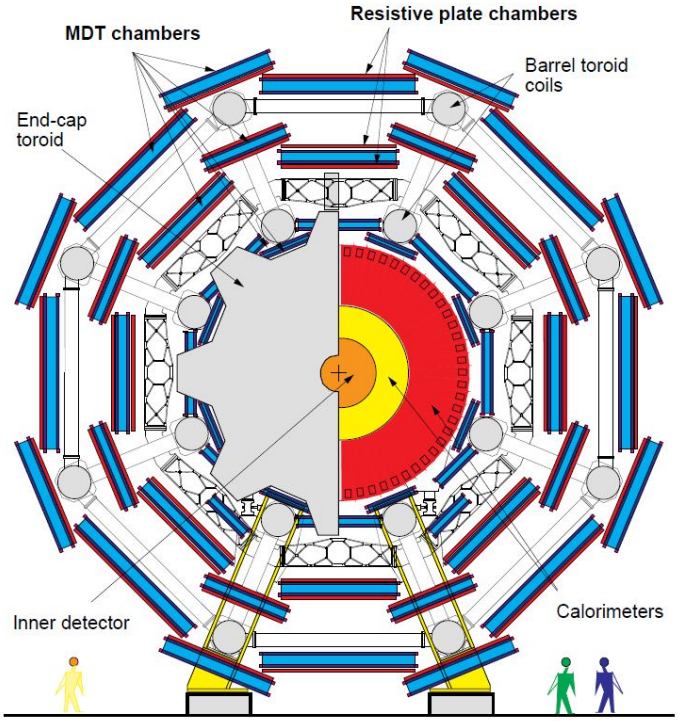

Figure 2.16: View of the ATLAS MS layout in the (x,y) plane.

Cathode Strip Chambers (CSC)

The Cathode Strip Chambers (CSCs) work in small
corner regions of the experiment (they cover the forward region $2 < |\eta| < 2.7$) and are therefore subjected to a higher rate ($\simeq$ 1000 Hz/cm^2 compared to $150 Hz/cm^2$ supported by the MDTs), which would not be well sustainable by the MDTs.
These are Multi Wire Proportional Chambers (MWPC), i.e. detectors formed by two cathode layers composed of strips between which is placed an anode layer.

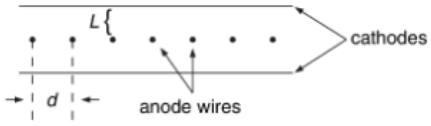

Figure 2.18: Sketch of a generic MWPC.

When a muon passes inside a CSC it generates a negative signal on one of the anode wires, which induces a positive signal on the strips belonging to both cathode planes. The position of the tracks is obtained by interpolating the charges read at the ends of these cathode strips and combining the information with the charge read at the ends of the hitted anodic plane. The CSCs reach a resolution per camera of $40\mu m$ in the bending plane and about 5mm in the transverse plane.
In the ATLAS experiment for data acquisition is required a fast trigger for muons selection, for this reason the so-called L1-Muon Trigger System is associated with the chambers use for precise muon tracking. This L1 trigger selects the candidate muons by assigning to it the correct bunch crossing of the corresponding LHC and classifies them within one of the six classes into which the transverse impulse is divided. It uses two different chambers, one for the barrel and one for the endcaps.

Resistive Plate Chambers (RPC)

Resistive Plate Chambers (RPCs) are used in the barrel region for $|\eta| < 1.05$ and provide six position measurements along the muon's trajectory in the spectrometer with a spatio-temporal resolution of approximately 2cm x 2ns. They cover an area of about $4000m^2$, with 3700 gas volumes, and operate under a toroidal magnetic field of about 0.5T.

The whole system of RPCs is divided into three concentric cylindrical sectors corresponding to radii of about 6.8m, 7.5m and 9.8m, where a different RPC station is located in each of them as shown in figure 2.19. Each RPC consists of two parallel layers of detectors placed at a distance of about 2mm filled with a gas mixture of $C_2\ H_2\ F_4$ (94.7%) $- C_4\ H_{10}$ (5%) $- SF$ (0.3%) and between them there is an electric field of about 4.9 kV/mm, which allows the formation of avalanches and therefore the creation of the signal [9].

Thin Gap Chambers (TGC)

The Thin Gap Chambers (TGC) cover a region in pseudorapidity for $1.05 < |\eta| < 2.4$, and in order to acting as a trigger for the region of endcaps they also allow to obtain a measure of the azimuth coordinate.
They are MWPCs filled with a gas mixture $CO_2 : n - C_5H_{12}$ (55:45) with a distance between wire and cathode less than the one between two wires (1.4mm vs 1.8mm) placed at a voltage of 2800V. The total system can be divided into three distinct stations of TGCs, as shown in figure

2.19.
They reach a speed of deliver signals within 5-25ns, and both give a further measure of the coordinate in the bending and transverse planes.

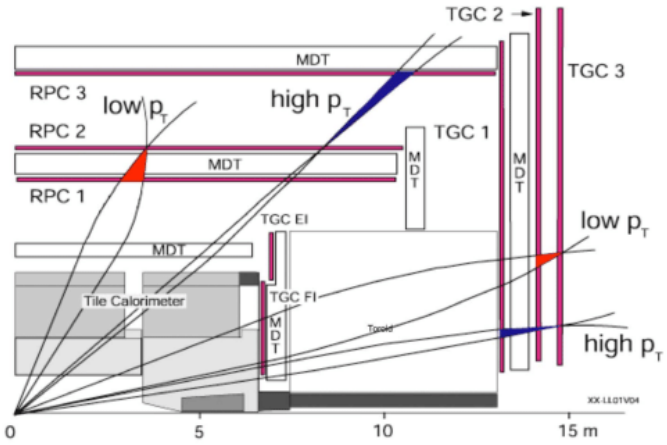

Figure 2.19: View of the different stations of RPC and TGC in the MS for the trigger system [13]

2.2.6 ATLAS performances: muon resolution in the Muon Spectrometer

Considering the work that will be shown in the rest of the thesis, it is useful to mention the resolution on the measurement of muons by the Muon Spectrometer. In particular, in the ATLAS experiment, four different factors contribute to the muons resolution, shown in figure 2.20:

1. Resolution of muon measurements in the tubes, considering also the calibration ($\propto p_T$);
2. Degradation given by the wrong chamber alignment ($\propto p_T$);
3. Degradation given by the muons passage through the calorimeters, in which they lose a small (and in general negligible) amount of energy, it's measurable from brem showers);
4. Homogeneous background given by the multiple scattering of the particles with the detector walls (cavern noise).

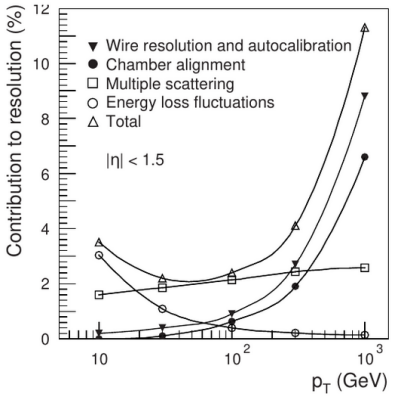

Figure 2.20: Contribution to the muon resolution of the Muon Spectrometer, averaged over $|\eta| < 1.5$ and azimuthal angle in a standard sector [26]

2.3 ATLAS trigger system

The ATLAS trigger and data acquisition system (TDAQ) [10] allows the selection of the events of interest with respect to the total ones. The event production rate during the Run-2 LHC is about 40MHz, so, due to limitation in data storage, in the ATLAS experiment a three-level trigger system was used for its reduction and event selection:

1. **Level 1 (L1)**, it's hardware-based and must work at frequencies as similar as possible the bunch crossing frequency of the LHC beams, so it works with low-granularity images taken by calorimeters and MS and define a region of interest (RoI). The information by calorimeters consists in clusters of energy deposits and missing transverse energy $\not{E}_T$ while the muon spectrometer ones are provided by trigger chambers and use transverse momentum and track position. The L1 trigger level reduces the event rate from 40 MHz to 100 kHz with a latency of 2.5 μs.

2. **Level 2 (L2)**, software-based, which integrates the RoI data over the full detector with a latency of $\simeq$ 10 ms.

3. **Event Filter (EF)**, it represents the last level of online selection and establishes, applying complex trigger algorithms, which events must be archived for data analysis. EF works with a latency of few seconds and at the end of the trigger chain the event rate is reduced at 200 Hz.

The system formed by L2 and EF is the so called High Level Trigger (HLT). The total action of the ATLAS trigger system is showed in figure 2.21.

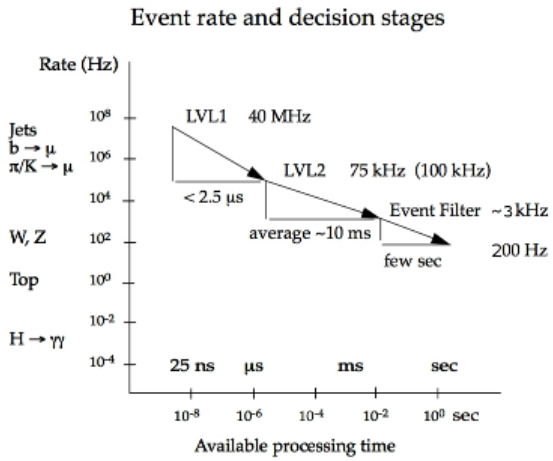

Figure 2.21: Variation of the event rate through the different trigger stages as a function of the response time of the components.

The L1 trigger level, having to work at very high frequencies, uses information with reduced granularity coming from the calorimeters and the Muon Spectrometer, and reduces the rate to about 75kHz.
The L1Calo trigger processes the signals coming from the calorimeters and analyzes them in parallel with a Cluster Processor (CP), which identifies photons, electrons and candidate tau

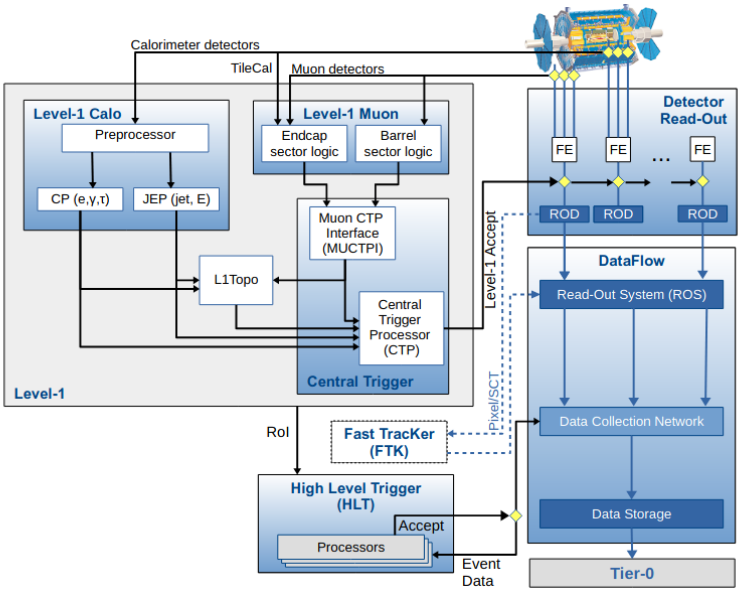

Figure 2.22: ATLAS TDAQ System during Run-2 containing relevant componentsa for triggering, read out and data flow [10]

leptons above a fixed threshold, and with a Jet / Energy-sum Processor (JEP), which instead identifies candidates for jets and elaborates an overall value for total and missing transverse energy.
In the muon spectrometer, the L1Muon trigger takes the low granularity information provided by the RPC and TGC chambers to select muons with high p_T.
The L1 trigger decision is then made by the Central Trigger Processor (CTP), which receives the information received from the calorimeters and trigger chambers in the Muon Spectrometer and whose information processing time defines the detector dead time (in the which no further events are acquired). It is able to process events that occur with a rate that is above the maximum for the detector readout (100 kHz) and below the bunch crossing rate (about 40 MHz), with a latency of 2.5 μs.
When a certain event passes the trigger L1, the region of interest (RoI) is defined in the plane (η, ϕ) in which the event will be observed with the second level trigger, and the Front -End detector (FE) reads the event information released in each detector component, sending the data first to the ReadOut Drivers (RODs), which works as pre-processors, and then to the ReadOut System (ROS), which acts as a buffer for the second trigger level.

The High Level Trigger (HLT), the second selection stage for the events selected by L1, is a software-based trigger that uses algorithms for the online reconstruction of some variables associated with the processes to decide whether or not they pass a given threshold. of interest, becoming more and more precise stage after stage.
It, as said before, is composed of the trigger level L2, which takes the information relating to the RoI identified by the trigger L1 and in about 40ms reduces the rate up to 3.5kHz; and an Event Filter, which using a series of offline algorithms reduces the rate down to about 200Hz in about 4 seconds.
The total scheme of the trigger system with the corresponding variation of the data flow, can be visualized in figure 2.23.

As anticipated in the previous paragraph and as shown in figure 2.19, the current trigger system L1 for muons is composed of the RPCs chambers, in the barrel region, and TGCs, in the endcaps. In both cases the trigger system uses the signals coming from three stations and looks for the coincidences between the cameras along a track that starts from the interaction vertex. The algorithms used depend on the type of muons you are looking for. In the case of low p_T muons, given a signal in the outermost RPC (RPC2), signals are sought (at least on one of the two RPC plates) coming from the innermost one (RPC1) within a region delineated by the joining of the hit revealed by the RPC3 and the interaction vertex. In the case of high p_T muons, on the other hand, the signal is also detected by the outermost RPC (RPC3) and the hit present on it is used for coincidence.
The collected signals are then processed using boards that amplify, shape and discriminate them. A board then combines the results obtained from the algorithms for high and low p_T and the results, both for the barrel and for the endcaps, are finally sent to the Muon Central Trigger Processor Interface (MUCTPI), through which the collected information is combined and send to trigger L2.

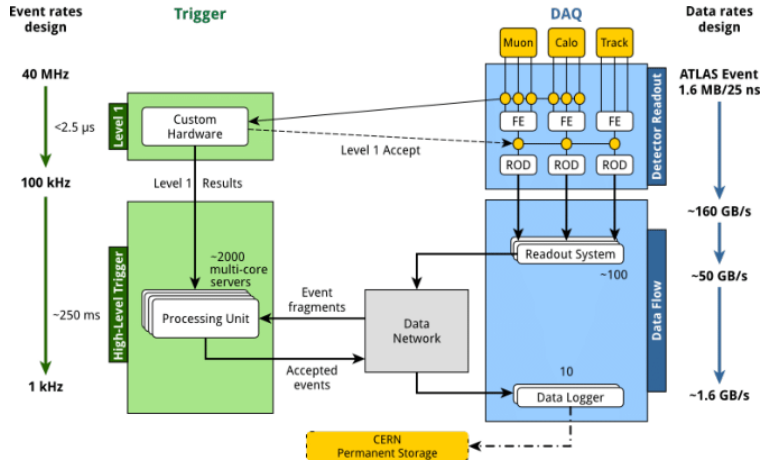

Figure 2.23: Complete scheme of ATLAS TDAQ system in Run2 [4]

2.4 HL-LHC: the upgrade

High Luminosity LHC represents a new phase of the LHC complex in which the luminosity of the original attitude will be increased, after 2025, until reaching an instantaneous peak luminosity of 7.5 x $10^{34} cm^{-2} s^{-1}$, corresponding to about 200 pileup [1] and the integrated luminosity will be increased by 250 fb^{-1} per year until to reach a value of 4000 fb^{-1} [25].
These upgrades in the ATLAS experiment will allow to study the properties of the Higgs boson, to make further precision measurements on SM and to investigate candidates of bSM theories (such as the detection of LLPs), in fact increasing the luminosity means increasing the number of events produced by a given collision, making it easier to see rare processes, and the luminosity can be increased by reducing the size of the beams at the collision point. In figure 2.24 the timeline of the experiment upgrade is shown. The upgrades to LHC will include both the accelerator ring (magnets, cooling systems and replacement of radiation-damaged components) and the detectors, which will have to work with a much higher amount of data than the current

[1] the pile-up is the number of events per bunch crossing

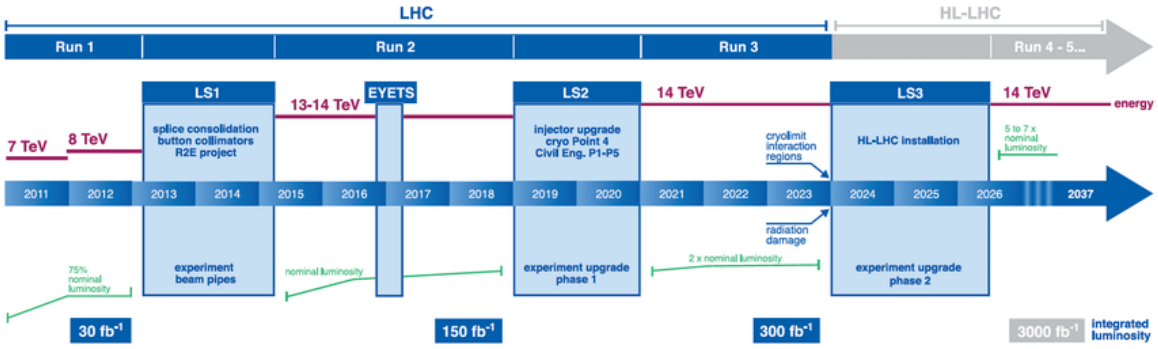

Figure 2.24: LHC plan for the next decade showing the steps for achieving Phase-II of High Luminosity [6]

one, in which the trigger system will be designed to be more robust and flexible, using external FPGAs and not custom ASICs placed on detectors for the calculation of coincidences.
In particular, the ATLAS trigger and data acquisition system (TDAQ), responsible for the online selection of events and the store of those selected for offline analysis, will have to support a higher peak brightness and its new structure will have two levels.
The first trigger level (L0 trigger), hardware-based, will use the information acquired by the Calorimeters (L0 Calo) and the L0 Muon trigger system. For the second level, of HLT, software-based, the possibility of putting a farm of FPGA accelerators to do the tracking in the inner detector in phase-2 with conventional algorithms is being studied.
In particular, Level0 will take the information received from the Calorimeters and the Muonic Spectrometer, passed through the respective Global Triggers with a latency time of about 5 μs, after which the events will be stored in readout buffers up to that the trigger makes the decision. The events selected in this way by L0 will then be sent to the HLT with a rate of 1MHz.
The upgrade of the acquisition system therefore requires changes in the trigger and readout modes of the muon spectrometer cameras, in particular of the RPCs, TGCs and MDTs.

The upgrades made with respect to the muon spectrometer will include the installation of new generations of RPCs in the Inner Barrel (BI), the design of a new set-up for the MDTs (to obtain space for the new RPCs), an upgrade of the TGCs, which there will be three instead of two with the aim of reducing the fake-trigger rate, the addition of a trigger for regions of high pseudorapidity ($2.7 < |\eta| < 4.0$) and finally the entire system of power for low and high voltage. In figure 2.25 the trigger scheme for Level-0 in during the Phase II is reported. The on-detector DCT boards sample the RPC data and send the digitized data on optical fibers to the off detector Sector Logic (SL) boards which perform the trigger selection and the readout [27].

As can be seen from the figure, the Muon Trigger decisions will be made by the Sector Logic (SL) both in the barrel and in the endcaps, using the information coming from the Muon Trigger Chambers, by the Tile Calorimeter and by the MDT Trigger Processor. The information collected by the SLs will then be collected by the Muon-to-Central-Trigger-Processor (MuCTPi), and the data corresponding to the muon hits will be read through the Front-End Link Interface eXchange (FELIX) and finally sent to the HLT.
A most in-depth analyzes of the upgrades that will be made to the RPC and MDT Chambers are reported in the Appendix A.
For the Sector Logic Unit, which performs the selection of the acquired events based on the

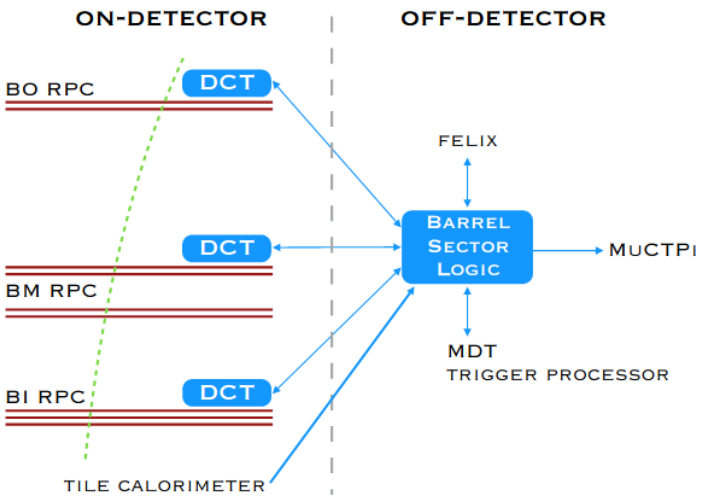

Figure 2.25: Level-0 trigger scheme for the barrel region for Phase II [27]

selected triggers using the information of the calorimeters and RPCs, it has been proposed that it be physically placed in an area outside the detector, the USA15, in order to reduce the radiation-damage and to be able to access the trigger system, for maintenance or modifications, in an easier way [12].
Furthermore, the pre-processing of the data acquired by the RPCs that was previously done by the Coincidence Matrices will be done by the Data Collectors and Transmitter (DCT), one for each RPC station, which will collect and pre-process the signals and then send them through to unique optical fiber to the SL.
As shown in figure 2.26, the structure of the DCT board includes an FPGA, used to sample and do the zero suppression of data coming from RPCs; a GBTx chip that serializes the FPGA input and output data; a GBT-SCA chip for monitoring and configuration; and finally by a Versatile Link VTRx, for electrical-to-optical transmission.

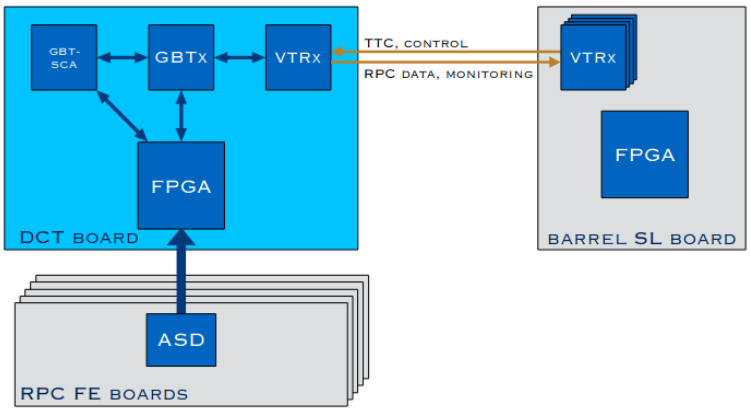

Figure 2.26: Block diagram of a DCT board [12].

Then, the pre-processed data coming from the RPCs and the calorimeters will be transported to the SL through DCTs, when they are collected, the MDTs will verify the goodness

of the event and in case of positive outcome the SL will take care of transferring the event to the Muon to Central Trigger Processor interface (MuCTPi), which, working together with the Central Trigger Processor (CTP), will pass the event to the HLT.
The SL will consist of 32 Field Programmable Gate Arrays (FPGAs), which are end-user programmable hardware devices that allow easy modification of their implementations. FPGA implementation requires the use of VHDL, a specific programming language for hardware description.
Another of the major upgrades designed for this new phase of LHC is proposed for the ATLAS experiment and consists in the hypothesis of an hardware based track L1 trigger that makes selection decisions using the information on the tracks from the signatures left in the Calorimeters and in the Muon. Spectrometer combining them with the precision ones made by the Pixel and Silicon Strip detectors of the ID. The combination of these signals is then processed offline by sophisticated algorithms that significantly reduce the trigger rate.
A given trigger algorithm is generally written using software programming languages, such as C++, and can be defined using a convolutional neural network.

Chapter 3

Neural Networks

Neural Networks (NN) offer a very powerful toolset that allows to solve problems as classification and regression, such as the recognition of a certain element in a given image or prediction problems of a continuous variable.

One of the advantages of using neural networks lies in the fact that they are able to solve problems without the need to have a specific model on which to base (therefore is not necessary to define precise parameters what describe the given theory that you want to study), but NN are able to learn by *training* themselves on a series of examples given to them at the beginning. Another advantage is in their speed of processing complex data.

Their idea is inspired by the information processing mechanisms in the biological nervous system, in particular the human brain; taking as a starting point the processing mechanisms of external agents or the connections between single neurons for the passage of information.

NN are then composed of a series of layers, each of which contains a certain value of neurons that are connected to each other and to the other layers' neurons through interconnection system.

The simplest model, which is the basis of the structure of the today's neural networks, is the one proposed by Frank Rosenblatt in 1958, the *perceptron*. It proposed that each neuron had a local memory composed of a vector of weights w, each associated to a given element of the input vector x, which define their "importance", and a bias vector. So the perceptron computed the sum of all the weighted input vectors and put the result in an activation function, which establishes the neuron output.

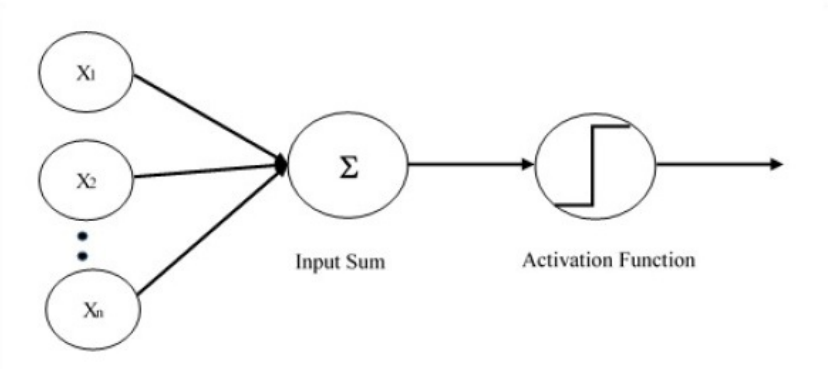

Figure 3.1: How Perceptron works

The Rosenblatt perceptron was able only to perform binary classification tasks, so the so-called

Multi Layers Perceptron was developed. It consist in a systems composed by several perceptrons organized in different layers and able to employ arbitrary activation functions (in the original perceptron only the step function was implemented).
The implementation of arbitrary and various activation functions is important in Machine Learning because introduce a non-linear component to the NN model and there exist several functions for different tasks. In particular NN can be divided in two main classes, based on the problem they have to solve:

- Classification NN: neural networks have to be able to divide the input images in different classes according to a given characteristics
- Regression NN: the NN have to be able to predict the value of a continue variable defined in a given range.

Also activation functions are characteristic for a given task, for example we can have:

- **Rectifier Linear Unit activation function (ReLu)**: defined as f(X) = MAX(0,X) and used to cancel the negative values obtained from the previous layer
- **Sigmoid activation function**: has the following form

$$Sigmoid(x) = \frac{1}{1 + e^{(} - x)} \tag{3.1}$$

  . and is used to multi-class classification problems as it's able to return the values corresponding to the belonging class of a certain input.

- **Softmax activation function**: mainly used in multi-class classification problems it returns, for each input, the belonging probability value to each class. It's defined so that the sum of the output values for an input is always 1, and therefore it's like a probability distribution and is defined as:

$$Softmax(x_i) = \frac{exp(x_i)}{\sum_j exp(x_j)} \tag{3.2}$$

Below we will analyze the Convolutional Neural Networks, which are basically multi-layer perceptrons with a special structure.

3.1 Convolutional Neural Networks

The basic structure of a neural network provides that this has an input layer, some hidden central layers, and an output layer. Convolutional Neural Networks (CNNs) are neural networks used for the processing of images (or audio signals) in which the hidden layers are composed of neurons interpreted as convolutional kernels that are applied on each patch of a given image, defining or not the activation of a given neuron.
In particular, imagining to have a dataset composed of images made by $N \times M$ pixels, these filters are $n \times m$ matrices that are passed, with defined stride, along the input data matrix $N\times$ (with $n, m < N, M$), and which allow to extract different characteristics of the given images according to the type of filter chosen.

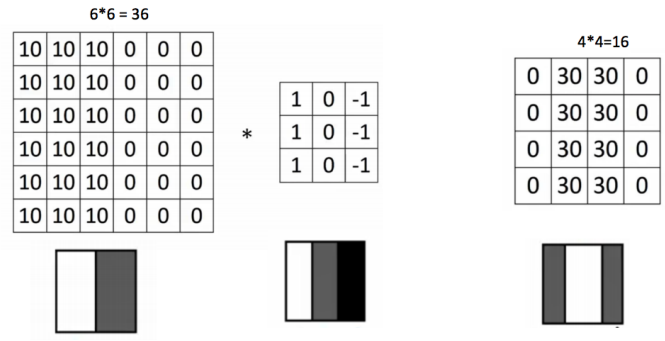

Figure 3.2: Operation of the convolution between a layer and a given filter.

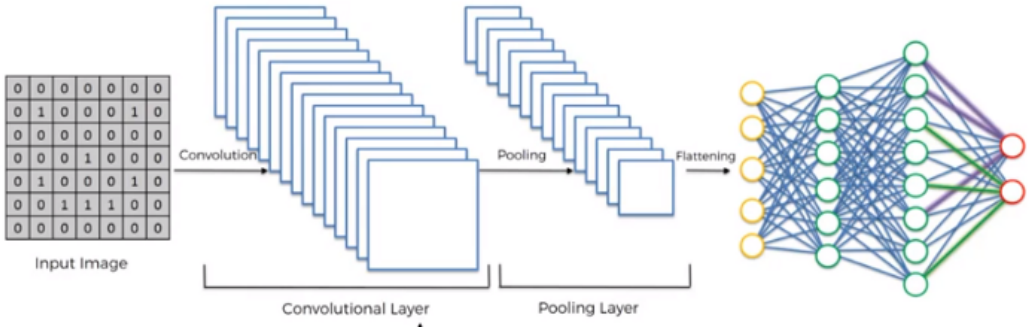

Figure 3.3: Simplest block diagram of a CNN

When a filter is passed on the initial matrix, a new matrix is obtained with the values given by the product of the submatrices for the chosen filter, called a feature map and in general smaller than the starting matrix, and using different filters it is possible to explore different characteristics of a given image.
Filters can be of different types, and based on their characteristics a different feature map is obtained, which allows easier and faster processing and which defines one of the main characteristics of the input image (for example vertical, horizontal or diagonal lines).
The simplest scheme of a CNN can be thought of as the following:

$$Input \rightarrow Conv \rightarrow ReLu \rightarrow Pooling \rightarrow FullyConnected \rightarrow Output \tag{3.3}$$

Imagining to be able to incorporate the ReLu layer in the Convolutional one, the simplest scheme of a constituent block of a CNN can be represented as in the figure 3.3.
So let's analyze the introduced layers.

- **Input layer**: in CNNs the inputs can be images or audio files whose dimensions, in the case of images, are defined with the number of pixels that compose them
- **Convolutional layer**: convolutional layers have within them the definition of the filters, defined by parameters such as the number of filters, their dimensions, the stride, which describes the size of the step made by the filter between one convolution and the next, and the padding, which allows to increase the input size of a given convolutional layer filling it with zeros, with the aim of not losing information (i.e. dimensions) in the output, as the outermost pixels of an image will always be analyzed less because the filters there they

will pass over less times.

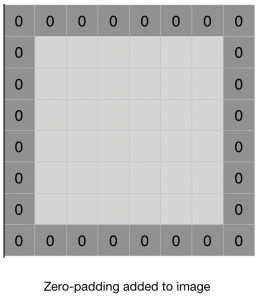

Figure 3.4: How Zero Padding works for a given image.

- **ReLu layer**: ReLu layers (Rectified Linear Unit) are non-linear levels that improve training speed without reducing accuracy, by applying the function $f(x) = max(0, x)$ for each value present in the layer, with the aim to eliminate all negative values (considered not useful) obtained from the previous layers. These layers act as an activation function, activating a certain node only if the input is above a certain threshold.

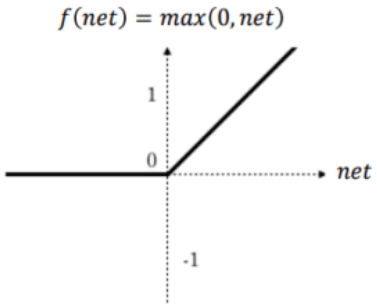

Figure 3.5: ReLu activation function trend.

- **Pooling layer**: it consists of a convolutional layer with a filter that has the stride dimension equal to the filter one (in order to never operate more than once on the same data) and that for each submatrix on which the filter passes it returns the maximum/minimum value (MaxPooling and MinPooling). These types of layers are used to simplify images, making them simpler but still maintaining information about the features of interest. The MaxPooling layer therefore reduces the size of the images, and is used because when you have very large weights associated to certain pixels, their exact position is no longer important as much as their relative position with respect to other higher weights. In figure 3.6 it's possible to observe how a MaxPooling layer acts on a 2D input image.
- **Flattening layer**: this layer doesn't change the value of the input elements but act only on their structure. The purpose of this layer is in fact to flatten the feature map obtained from the Pooling level in a single column in order to put them into the Fully Connected network for processing (figure 3.7).
- **Fully Connected layer**: it is fundamental and allows you to obtain a network where all neurons are connected both with those of the previous layer and with those of the next

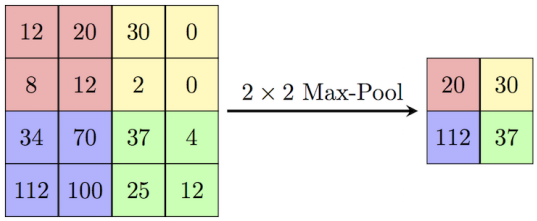

Figure 3.6: How MaxPooling works.

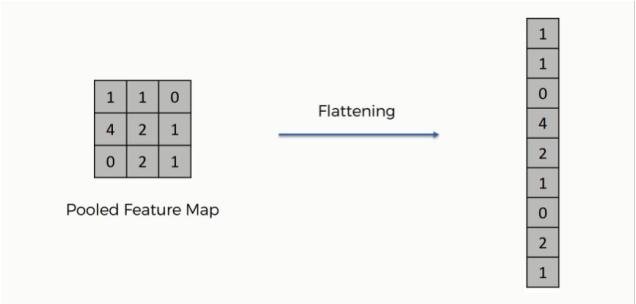

Figure 3.7: How Flattering layer works [23]

one, allowing to solve the given problems having a more complete view of the information available. It operates by applying a linear function of the type $y = x\dot{w} + bias$, with x input and w weights associated, and then returns an input connected to the output. As showed in figure 3.8.

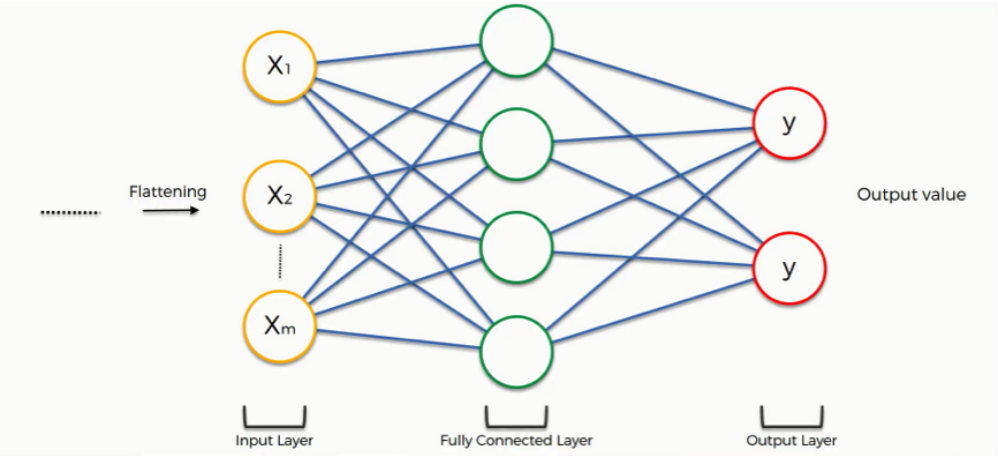

Figure 3.8: Fully Connection Step [24]

A network composed in this way is then introduced into a dense network. The **dense layers**, defined in the same way as the Fully Connected layers only by the number of neurons belonging to them, allow to observe the network globally and not only by going, for each neuron, to see the information contained by its adjacent ones. In fact, the dense layers represent the final stage of the structure of dense neural networks, and allow the connection between all the neurons in the network. They are also defined with activation functions, one for each layer, and these are often alternated with data modulation layers, such as those of Dropout.

Dropout layers represents a solution in cases there the network is found to be memorize instead of learning (overtraining) and allow to increase the total prediction efficiency of the given network. In fact dropout layers allow to obscure, in the network training phase, a data input fraction, leading to have a slower fit performance, as the network must understand by itself some of the missing data, which leads to have ad overall lower resolution in the train variables but reaching in this way an overall higher prediction efficiency on the test variables.
Finally, the **Output layer** of a given network may have a different structure based on the required task. In fact, it is always composed (in our cases) by a dense layer, in which the number of neurons and the type of activation function change according to the purpose for which the network is written. In particular a network that acts as a regressor will have as output layer a dense layer with a single neuron activated by ReLu which produces in output the estimate of the value to predict associated with the given input configuration.

In the case of the classifier, on the other hand, there may be two distinct situations: 1) the last dense layer has a single output neuron and is activated with Softmax, which returns in the output the probability of belonging of a given event/image to each classes defined so that the sum is equal to 1; 2) the last layer has a number of neurons equal to the number of classes and is activated with a Sigmoid, and returns in output, for each input, the most probable class of belonging.
CNN structures with more layers are called Deep Neural Networks (DNN).

3.2 Training

CNNs then consist in series of layers where each layer is composed of nodes (neurons), and each layer is associated with an activation function that decides whether a given neuron of the layer is to be activated or not, based on its "importance", defining an overall output of the layer which will be the input of the next layer.

Once the model has been defined, it must be trained. The role of training a neural network is of fundamental importance as through it the weights and bias values associated with each neuron are chosen and optimized.

Depending on the type of purpose for which the network is used, there are different types of training: *supervised training*, where you give a set of inputs and the corresponding set of outputs and the network then learns to generalize the results for each input, used for classification and regression; *unsupervised training*, where only the inputs are given without the respective outputs, used to identify the logical structures of the input data; *reinforcement training*, where are provided only the result to be obtained and an action that acts as a 'reinforcement' which allows you to get closer to the ideal result, and it's used to train the network to eliminate the actions that take you away from the ideal result; and finally the *semi supervised*, where mixed data sets are provided, ie inputs with corresponding output and even without.

During this work were used only supervised models, for this reason from now on in the definition of the model it is understood that this is supervised training.
When working with supervised models for training, functions capable of evaluating the accuracy of the model are used, called **loss functions**, which compare the results obtained by passing the data forward (i.e. from input to output) through the model with true values.
Some examples of loss functions that we use in training our neural networks are the Mean Square Error function (MSE), used in regressors, and the Categorical Crossentropy, used in multiclass classification networks.

MSE is calculated by taking the mean square deviation between the value predicted by the

network and the true one, while the Categorical Crossentropy, defined as

$$Loss = -\sum_{i=1}^{outputsize} y_i \times log\hat{y}_i \tag{3.4}$$

with $\hat{y}_i$ i-th output scalar value, target size equal to the number of output scalar values, and y_i value of the corresponding target, is used to compare the belonging probability distributions between the hypothesized class and the real one.

Another loss functions, used in multiclass classification problems, is the sparse categorical crossentropy. It is defined in the same way as the categorical crossentropy but the two functions are used in two distinct cases: the categorical crossentropy is used when the targets are one-hot encoded, i.e. the belonging to a certain class is defined by a series of values contained in an array of dimension N, each of which defines the probability of belonging to each of the N classes, while the sparse categorical crossentropy is used when the targets are integers, so the classification is not done using vectors but simple integer numbers.
For example, in the case of three classes, the one-hot encoded classification provides a structure of the type [1,0,0], [0,1,0], [0,0,1], while in the case of integers we would have the targets 1, 2, 3.

During the training, therefore, the model itself adjusts weights and bias values of each neuron with the aim of minimizing the value of the loss function. This minimization is done through a technique called ***gradient descent***: the model moves, with small steps, towards the direction where the gradient is smaller than the previous value, until it finds the local minimum of the function.

In detail, defining the cost function $C()$ as the square of the sum of the differences between the obtained and the expected activation, as a function of the parameters associated with the entire network; for each point $C(w)$ is taken the tangent to the curve and the network moves to right or left by an amount equal to the inclination of the tangent (which therefore decreases near the minimum).
Given W the vector of the network parameters, and $-grad(C(W))$, the training wants to find the values for which $w_i + c_i$ is minimum.

Another type of functions used to evaluate the model performances are **metrics**, which basically can be chosen from the same range of loss functions but which have the characteristic of not being used to update the parameters of the model during its training, but only to monitor the performances. One of the metrics used in our networks is *Accuracy*, which is used in the multiclass classification and allows the training to decide which metric to use based on the output dimension and the chosen loss function (for example, it can use the binary accuracy for classification problems with 2 labels, categorical accuracy in the case of targets one-hot encoded or sparse categorical accuracy in the case of multiclass classification with integer targets).

During the training ***Optimizers*** are also used. For each step of training (called *epoch*) they evaluate the previous weights values obtained in order to find, if it exists, the global minimum of the loss function, without stopping at the first local found.

A technique that allows to reduce the error associated to each neuron is the so-called *back-propagation*, i.e. the data flow not only from input to output but also in the opposite direction, which allows to associate an error on the prediction/classification and to be able to adjust in the right way the network parameters.

Adam optimizer algorithm, for example, defining the learning rate [1] in a variable way, represents a stochastic gradient descent method (a gradient descent which works at each iteration

[1]Learning Rate is a hyper-parameter which defines the step size taken for each epoch while approaching the minimum of the loss function

considering as the cost function gradient value the one calculated only on a small group of parameters) that is based on adaptive estimation of first-order and second-order moments (i.e. mean and uncentred variance).

To train the model to recognize certain input patterns, and to test whether it actually learned correctly, the input dataset is split into several sub-sets, each used for a different purpose.

In particular there will be: the *train dataset*, on which data the CNN performs its training, the *validation set*, which is used to monitor the performance during the training and which is defined as fraction of the original train dataset, and the *test dataset*, which is used post-training to test the performance of the trained CNN.

Chapter 4

DNN implementation using Keras with Tensorflow

During this work only commercial tools and applications were used with the aim of avoiding the handwriting of firmware or other similar complications by the user, in particular in fact, the construction of the trigger algorithms using DNNs, their training and their implementation on hardware devices has been based only on the use of the publicly available libraries for deep learning TenosorFlow and Vitis AI.

Tensorflow is an open source library which supports languages as C++, Java and Python and contains libraries for the implementation of algorithms for Machine Learning and Deep Learning, for our work we have used its last released version, the Tensorflow 2.6.0, released in August 2021, writing programs using Python. The advantage of using Tensorflow lies in the fact that in it the nodes of a network (each representing a mathematical operation) and the tensors formed by the grouping of them are well-defined Python objects on which it is easy to operate. Keras is an application programming interface (API) for Python used for building neural networks which is a front-end for libraries like Tensorflow. With Keras is possible to implement and train classification and regression algorithms based on neural networks that use operations which can be implemented in a simple way by TensorFlow.
The Keras advantage lies in the simplicity of the definition syntax for the various DNN layers and for the train and test operations, whose implementation with TensorFlow alone is instead more complex.
During this work an algorithm based on Deep Neural Network is implemented using Keras libraries with Tensorflow: a regressor DNN model for LLPs selection.

The regressor has been implemented with the aim of writing an algorithm able to predict the decay length value of a given particle by observing the tracks of its decay products reconstructed in the MDT chambers, finally deciding whether to accept or reject the event depending on the set threshold. Then, after having trained this model and studied its performances as trigger algorithm, a pipeline was built for its implementation on hardware devices as FPGA commercial accelerators.

Below there is a description of the available dataset and the DNNs that we have trained and then implemented on hardware, defining its structure, characteristics and performance.

4.1 Dataset

The analyzed dataset is composed of events containing candidates of Long Lived Particles which decay into SM particles whose tracks can be observed in the muon spectrometer chambers. In

particular our idea is to implementing a trigger algorithm for the selection of this type of events:

$$\begin{aligned} &\gamma_d \rightarrow \mu^+\mu^-, \pi^+\pi^- \\ &\pi_V \rightarrow b\bar{b} \end{aligned} \tag{4.1}$$

where γ_d is the *dark photon* which can have a mass in the range between 100 MeV and 3 GeV, and which can decay, according to its mass value, into a pair of muons or pions; and the *hidden pion* π_V can have a mass value between 10 and 30 GeV and can decay in a couple $b\bar{b}$.

These two particles are both theorized by hidden sectors bSM theories, which predict the existence of new sectors separated from the Standard one (called *hidden sector* or *dark sector*)that can be described by different quantum numbers than those of the SM. In particular the dark photon arise from an hidden sector theory which can be described by the symmetry group U(1), while the hidden pion is predicted by theories described by the symmetry group SU(3).

Particles of hidden sector theories can be produced and can decay in LHC through particles, like the Higgs Boson, which work as *mediators* between the SM sector and the dark one.

Then, considering the detectable final states associated to each analyzed event, we can consider that in the case in which the dark photon γ_d decays into a pair of muons we'll have in the final state 2 charged tracks, otherwise, in the case in which in the final state pions or a pair $b\bar{b}$ is produced, considering that pions interact with the detector and quarks b hadronize due to the quark confinement, in the final state we'll have an undefined number N of tracks.

In order to exploit the ability of CNNs to detect patterns for a given set of images, the signals released by the LLPs decay products in the Muon Spectrometer were represented as images. In particular the dataset used is composed of a sample of 10k simulated images acquired by the MDTs chambers placed in the inner barrel region of the Muon Spectrometer which contains 2 or 10 tracks.

The images were simulated starting from the hits left by the charged particles on the three MDT stations, projecting them in a two dimensional mesh with constant and fixed size of each pixel (thus losing the relative information on the reciprocal distance between the stations) with the goal to predict the decay point of the neutral particle from the pattern of hits into the image produced by the decay products. The two steps are showed in figure 4.1

For our purpose we have decided to consider, for simplicity, that decay products do not interact before the Muon Spectrometer, and then the 2-tracks final states are associated to muons and pions and 10-tracks ones to the $b\bar{b}$ production.

To each image is then associated a resolution based on the ATLAS MS precision on the muon momenta measurements, considering values taken similar to the real values acquired during the Run-2 for the resolution, and also is added a uniform background given by the noise and cavern background which are present in the whole region.

Then to each image are associated the following physical quantities of the hidden particle: its radial decay length L_r, its pseudorapidity η and its amount of transverse momentum p_T, and the idea is to train a DNN to predict the radial decay length value associated to each event.

These variables were then generated with homogeneous distributions for the entire depth of the detector, in particular:

- the decay length L_r, expressed in meters, was extracted in a uniform way between 0 m and 4.9 m
- the pseudorapidity η was extracted uniformly between the values -0.6 and 0.6
- the transverse momenta p_T was extracted uniformly in the range [1,200] GeV, assigning to each particle of the final state, in the case of two-body final state a fraction of 0.5 and in

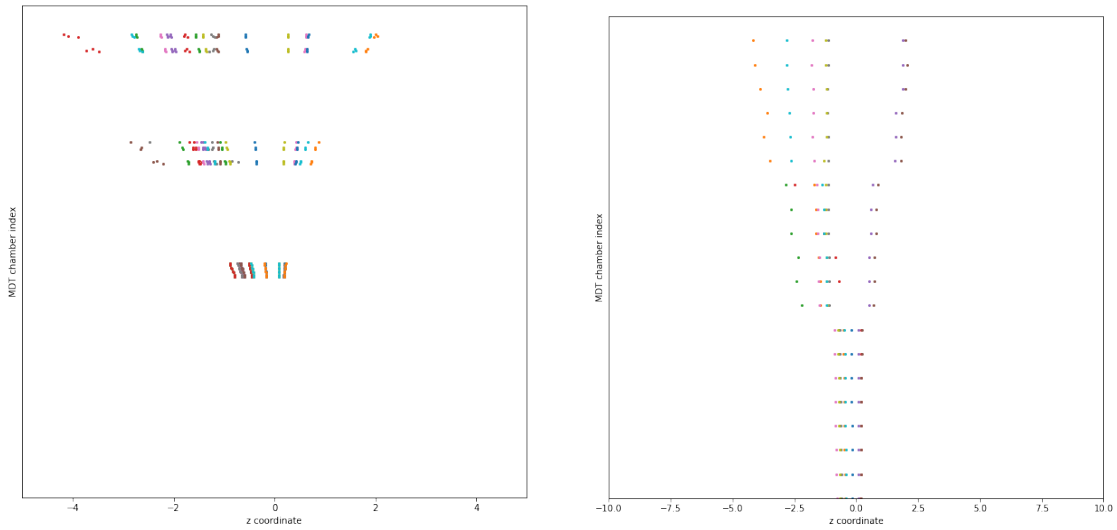

Figure 4.1: Simulation steps for 10-tracks events, *On the left* the simulated images composed by the hits left by the charged particles produced on the three MDT stations; *On the right* their projection in a two dimensional mesh with constant fixed pixel size.

the case of 10 bodies a fraction of 0.1 of total momentum of the decayed particle, adding to each of them a Gaussian smearing according to the expected momentum resolution of the muon spectrometer detector.

In this way are therefore constructed 10k images with dimensions 20×333 (number of MDT stations $\times$ granularity along the z axis) which show the decay products of a decayed neutral particle in the calorimeters on the (y,z) plane.

The simulation steps of the images acquired by the MDTs can be observed in figure 4.2, while the final figures, one for each of the two types of simulated events, are shown in figure 4.3.

Then let's analyze the implemented DNNs.

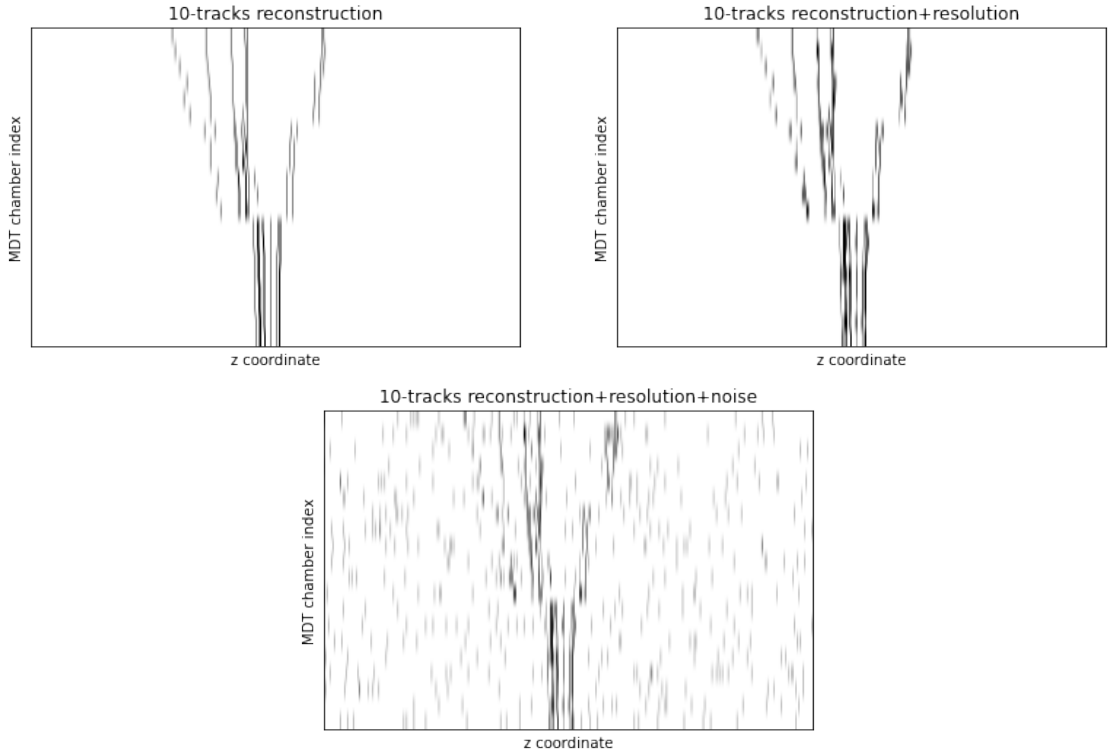

Figure 4.2: Reconstruction steps for 10 tracks images acquired by the MDTs. *By the left to the right*: only tracks, tracks with resolution, tracks with resolution and noise.

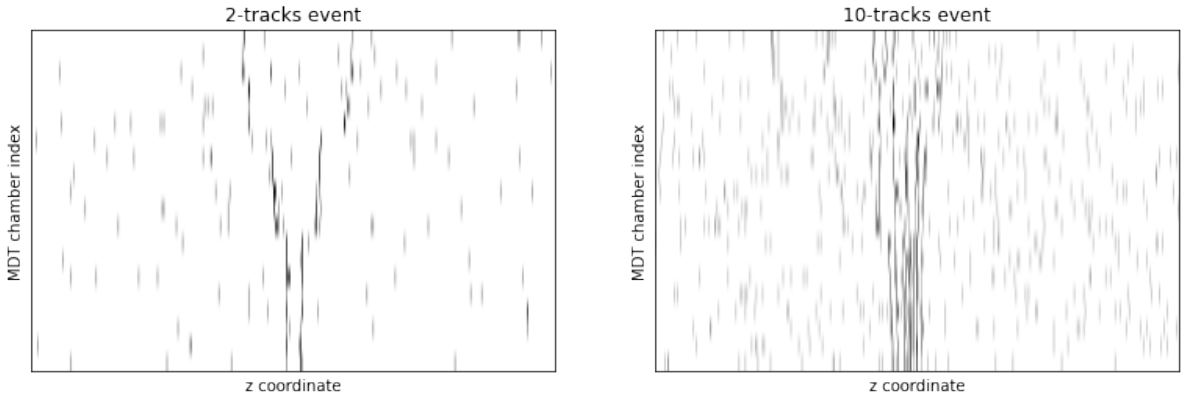

Figure 4.3: *On the left*: the reconstruction of the image in the MDT with two particles in the final state. *On the right*: the reconstruction for 10-body final states.

4.2 Regression DNN

The starting point of our work is the construction and training of a regressive DNN which, given the simulated images acquired by the MDTs, predicts the decay length value associated to the events. In particular, it's used supervised training, i.e. the interesting associated labels were provided during the entire training, so that the network was able to compare the predicted values with the real ones associated with each image. In particular only the decay length L_r label is used.

The implemented model is composed of three convolutional layers, the result of which is inserted, passing through a Flatten layer, in 3 dense layers, each associated with a ReLu activation function. A schematic image of the network is shown in figure 4.4, with the final number of parameters and of the trainable ones is reported.

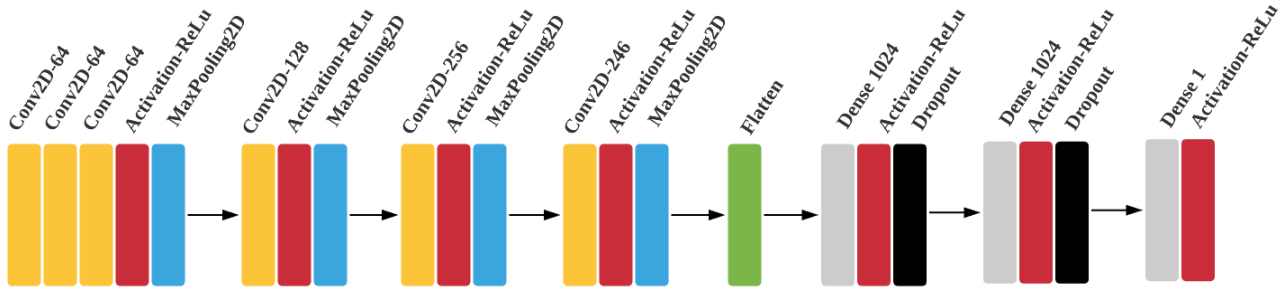

Figure 4.4: Regression DNN scheme. To each type of layer is associated a different color, in particular the yellow layer are Convolutional layers in 2 dimensions, and are expressed with the size of the kernel, red layers contain the ReLu activation function, the blue layers are the MaxPooling (always 2D) ones, the green is the Flattering layer, grey layers are the Dense ones, described with their number of neurons, and finally the black layers represent Dropout Layers.

The model is then compiled using the Adam optimizer, and choosing as loss function the Mean Squared Error distribution ('mse') and as the metric function, used for the model performances estimation, the Mean Absolute Error function ('mae'). One of the characteristics of this network training is that the Adam optimizer algorithm is chosen to have a learning rate with a value which varies during the training in the following way:

```
LR_ST = 1e-2
def lr_decay(epoch):
 if epoch < 5:
  return LR_ST
else:
  return LR_ST * tf.math.exp(0.2 * (5 - epoch))
```

so that the increment steps of each epoch decrease when approach the minimum of the loss function, in order to have more detailed information near that region.
Then, training the model, the optimal performances estimated with the validation set are:

```
10 tracks : loss - 0.11     mae - 0.24
2 tracks  : loss - 0.36     mae - 0.43
```

and the loss function and the metric trends can be observed in figure 4.5 for the 10 tracks model and in figure 4.6 for the 2 tracks model.

In the figures two different trends are shown: the blue one, which correspond to the trend of the given function computed on the train dataset, and the orange one, corresponding to the same function computed on the validation dataset (the validation dataset is fixed to be the 20% of the total train dataset in both cases). It's easy to imagine that the reason why the performances on the prediction of the decay length values are worse for 2-tracks events lies in the fact that for this type of events the DNN has less information to train on.

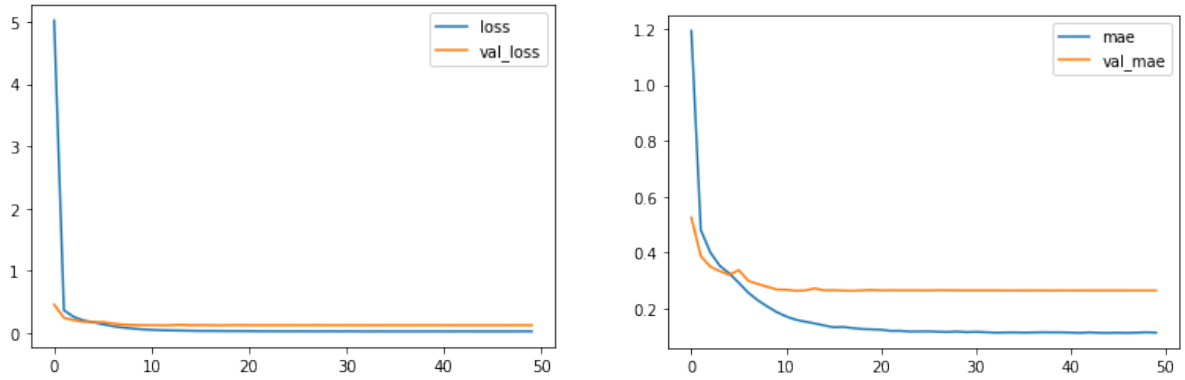

Figure 4.5: *On the left*, the overlap of MSE trends for the validation dataset and for the training one for 10-tracks events, *On the right* their overlap MAE trends.

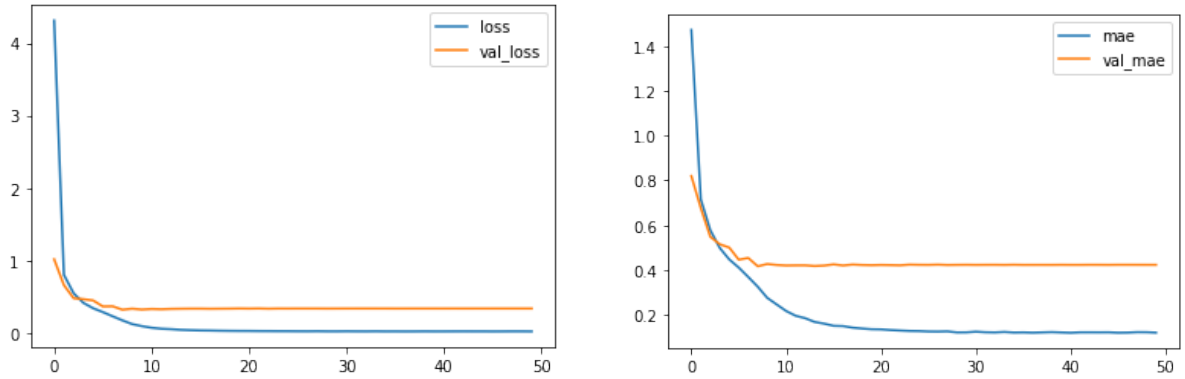

Figure 4.6: *On the left*, the overlap of MSE trends for the validation dataset and for the training one for 2-tracks events, *On the right* their overlap MAE trends.

The performances differences in the prediction between the two datasets can be observed from the scatter plots in figure 4.7, which show for each predicted value of L_r which is its *confidence interval* in the real values (i.e. which truth values can correspond to a certain predicted value).

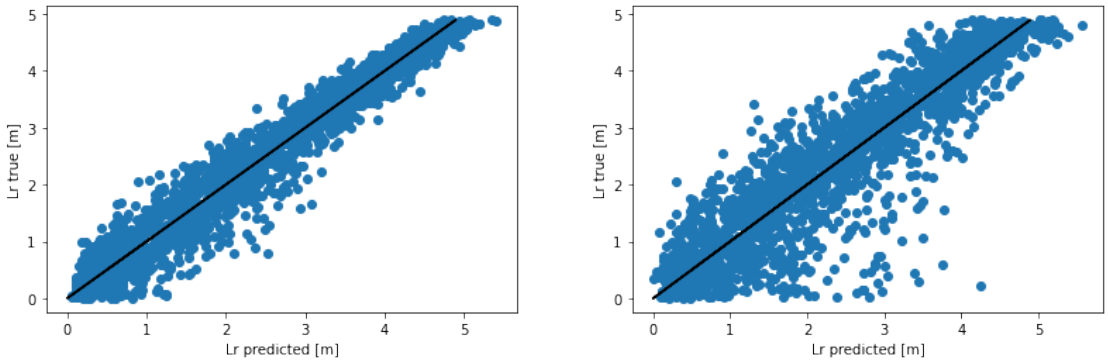

Figure 4.7: Scatter plot between predicted and true values of L_r for 10 tracks events (left) and for 2 tracks events (right).

The scripts used for DNN model and dataset definition and for its training are reported in the Appendix B. Having to work as a trigger algorithm for physical event selection, after the training the trigger performances for the model was studied in terms of its resolution curve and efficiency plot.

4.2.1 Resolution on the predicted radial decay length L_r

The resolution curve reports the estimated resolution on the predicted decay length from the DNN model as a function of the true decay length of the particle. The resolution in this specific case is defined as the sigma of a gaussian function fitted on the distribution of the truth-predicted radial decay length in different bins of truth decay length of the particle. In particular it's built following the following steps:

1. the distribution of the L_r true values taken from the test dataset is divided into fixed width bins
2. for each bin the histogram of the difference between the predicted values and the true values of L_r is constructed
3. each histogram is compared with the Gaussian distribution and the sigma value of the Gaussian is then considered as the prediction error for the L_r values within a certain bin
4. the plot sigma vs. L_r is constructed

Figure 4.8 shows the histograms for some of the bins into which the distribution of the L_r values for the 2-trace events has been divided, together with the Gaussian distribution which better fits the trend, of which the sigma represents the prediction resolution of the L_r value of the corresponding bin.

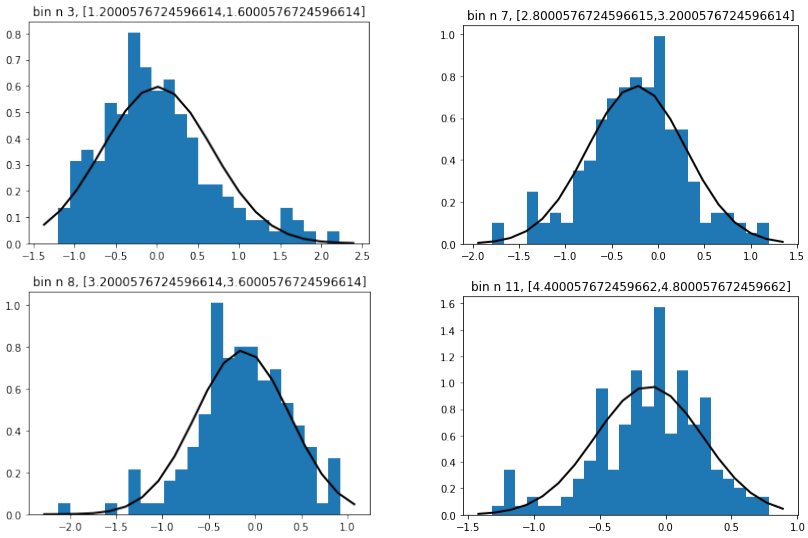

Figure 4.8: Trend for different bins (i.e. different values of L_r) distribution of the difference between the true value and the predicted values associated to the same bin. They turn out to be well-fitted by the gaussian function and then the sigma associated to the Gaussian distribution of each bin is considered as the resolution of the measure for the given L_r value. On the top of the image is reported the number of the bin and the range interval of L_r values in it included.

Similar trends are obtained for all the bins (in total 13 for both types of events) and the two resolution curves have been constructed on the basis of the sigma values extrapolated from the Gaussians. So, following the reported steps the resolution curves shown in the figure 4.9 were then constructed for 10 and 2-tracks events.

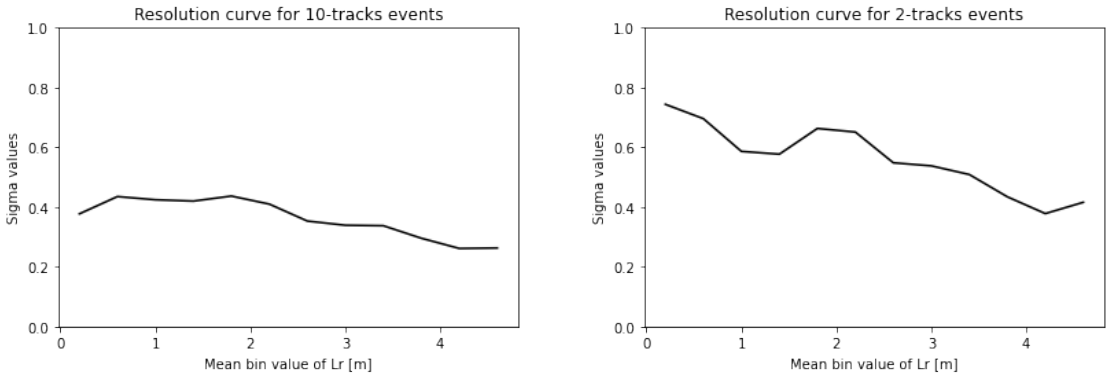

Figure 4.9: Resolution curves for 10 tracks events (left) and 2 tracks events (right).

They show that the error on the measurement decreases as the predicted value of L_r increases, and which for large L_r values the resolution is small for both cases. It allows us to think of using the trained DNN as the basis for a trigger algorithm for the selection of LLPs, which generally have large values of L_r (i.e. from inside the calorimeters to the end of the MS).

4.2.2 Efficiency in selecting highly displaced decays

The efficiency plot for a given trigger algorithm allows to assign, for each possible fixed threshold on the reconstructed quality used to select the events of interest, the ratio between the true events that have predictions that pass the threshold and the total true events.

$$\epsilon = \frac{Lr_{test}[predicted > \alpha]}{Lr_{test}} \tag{4.2}$$

In fact, given a certain threshold value α, the ideal efficiency plot trend is the step-function one, which results to be 0 for predicted values less than the α value and 1 otherwise. In reality, however, given a certain threshold value α, resolution effects make the actual performance of the efficiency plot smoother than the step-function and, even for predicted values greater than α, does not reach efficiency values equal to 1 but there is a plateau region at smaller values.

Then, if the efficiency plot is considered together with the analyzed events distribution trends, it allows to compute the fraction of events which pass despite they have values smaller than α. It's computed taking the distribution histogram of the true L_r values of the test dataset, and then taking the distribution histogram given by the predicted values for the dataset test which result greater than a fixed value. So, dividing bin-by-bin these two histograms the efficiency plot is founded (and reported in figure 4.11 for both 2 and 10 tracks events).
Considering that our DNNs must work as trigger algorithms, an optimization of their parameters will therefore be aimed at reducing the amount of background signals taken for a certain threshold value (events containing LLP that we would like to identify with these trigger algorithms are extremely rare compared to the SM signal, which is at least 10^5 times large).
After having defined and estimated our selection algorithms performances, in order to use them as trigger algorithms in the ATLAS experiment, is necessary to find a way to implement them

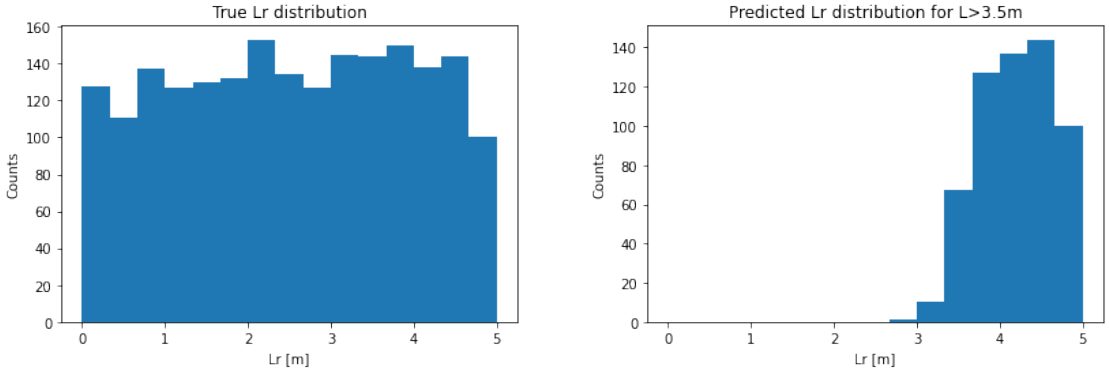

Figure 4.10: *On the left*: Histogram of the total true values of L_r of the test dataset; *On the right*: Histogram of the test dataset events associated to a predicted value $L_r > 3.5m$ for 10-tracks events.

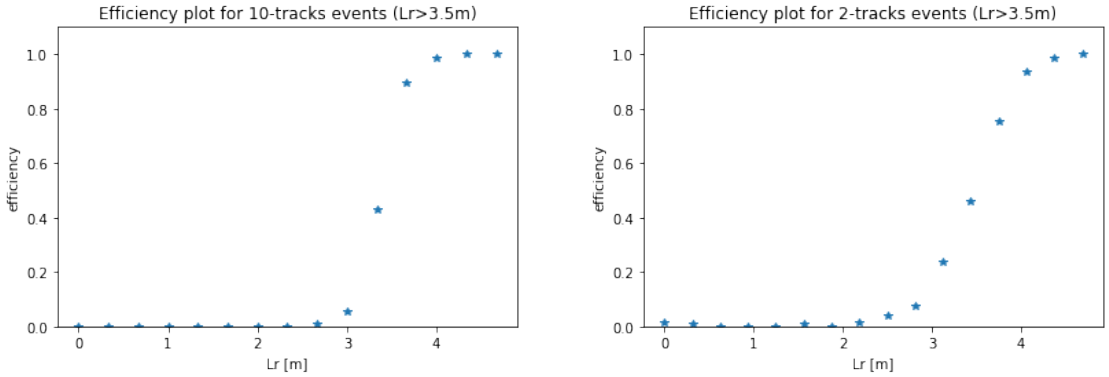

Figure 4.11: Efficiency plot for 10 tracks events (*left*) and 2 tracks events (*right*) both for $L_r > 3.5m$.

on commercial accelerators such as FPGAs. This requires both to optimize the DNN model in order to fulfil the thigh requirements in terms of resource (memory size occupation) required by the FPGA accelerators, and to eventually synthesize (compile) the code in the firmware needed by the FPGA processor.

Chapter 5

Implementation of the trained DNN model in commercial FPGA accelerators

The final goal of this work is to eventually build a demonstrator in which the developed DNN models described in the previous chapter are implemented in a test slice of the ATLAS HLT trigger, in order to study the technical and physical performances of the proposed system.

In particular the scope of this thesis is to implement the first step of a demonstrator for the HLT trigger with a proof of principle of the method. So in this chapter using the Vitis AI library from Xilinx we will optimize the CNN models for memory occupation and compile them for the latest generation of FPGA based accelerators commercially available in order to estimate the performance in terms of inference speed compared to conventional CPU and GPUs.

5.1 Vitis-AI

Vitis AI is a software platform which allows to optimize, compress and "translate" in hardware languages (VHDL) DNNs, in order to implementing them on Xilinx hardware devices. It is a commercial tool and provides a work pipeline that doesn't require the user to handwrite the firmware or something similar. [28]

It is composed of a series of optimized IP cores, tools, libraries, templates and examples that allow it to be an easily accessible platform for all users.
Vitis-AI supports popular frameworks such as Pytorch, Caffe and TensorFlow and its libraries offer APIs for both C ++ and Python. It allows the implementation on Deep-Learning Processor Units (DPUs),

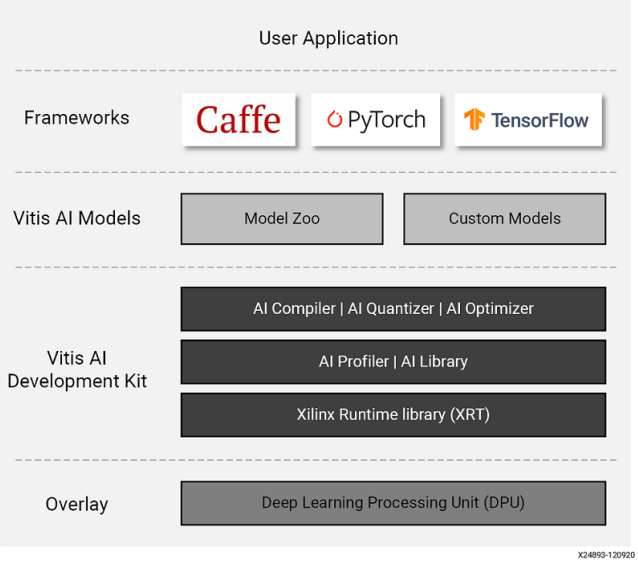

Figure 5.1: Vitis AI Development Kit

which are specific programmable machines
for the implementation of deep neural networks algorithms, and provides many devices on which it's possible to tests the performances switch to physical hardware.
It is composed of a series of key components that allow the user an efficient and simplified implementation, below we will give a brief overview of each of them. An overview of the Vitis AI available development kit is showed in figure 5.1.

5.1.1 AI Optimizer

It consists of an optimizer model that allows, by performing pruning and fine tuning operations on the model, to obtain a simplified model with minimal accuracy degradation. It requires a commercial license to run, and the deep compression it performs takes that AI inference to high values.

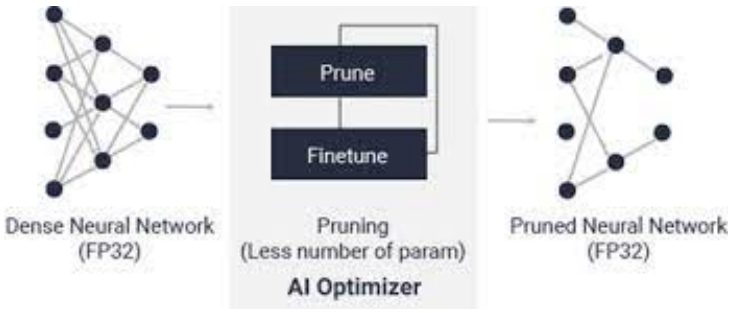

Figure 5.2: Vitis AI Optimizer

As showed in figure B.5, using pruning and finetuning techniques, the new optimized model results to have a smaller number of parameters without losing accuracy. In particular, the pruning approach allows to remove model nodes associated with the lower weights, providing a new model which contains only interesting features maintaining the total information. During this work it was not necessary to use the AI Optimizer as the developed model was already sufficiently simple to not require further pruning and simplifications other than the usage of the quantization technique as described in the following paragraph.

5.1.2 AI Quantizer

Quantization is the process of transforming a given Deep Learning Model into an equivalent representation which uses lower precision parameters, and through the quantization the final model results to have improved its execution performances and its efficiency, and also it allows model execution on specific accelerators.

The AI Quantizer is used for the quantize, calibrate and fine tuning a given DNN model, it works by converting the weights and activation functions of a given network from 32-bit floating point (F32) to integer values at 8 bit (I8), thus reducing the amount of memory required by a factor 4 and increasing both speed and power efficiency compared to the floating point model ($\simeq$ 1.5-4 times faster in computation).

After quantization it is therefore possible to use integer computing units, and the representation of weights and activation functions is done using the lowest bits.

Model quantization provides a fundamental step for DNNs implementation because of FPGAs have a limited memory and bandwidth and then having a smaller model (reduced by a

factor 4) using only integer values makes the hardware devices works better. However, the parameters F32-I8 conversion is a lossy one, because of I8 has a biggest values of information channels and then the quantized model precision results reduced.

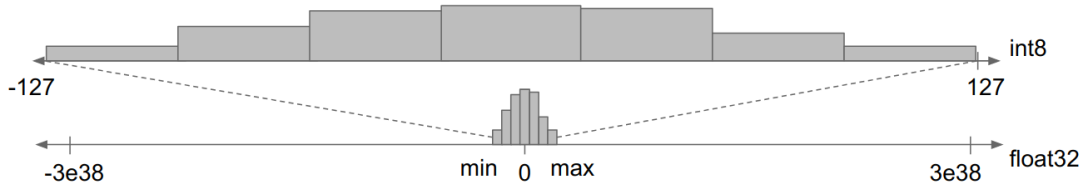

Figure 5.3: Small F32 range values mapped into the I8 range.

Then, after the quantization it's often necessary to adjust the converted parameters, and this can be done either without re-training the quantized model, and this process is called Post-training quantization (PTQ), or re-training it with the Quantization aware training (QAT) technique. These two approaches are both supported by Vitis AI Quantizer for Tensorflow2,

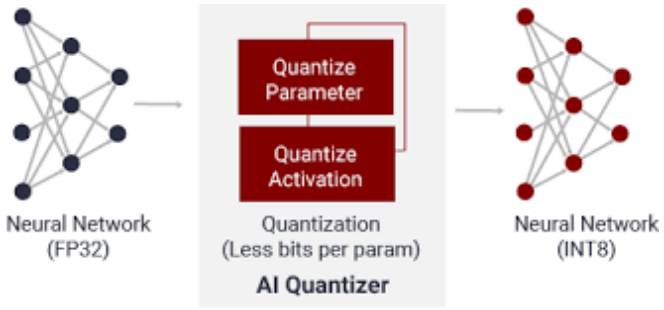

Figure 5.4: Vitis AI Quantizer

and below a further discussion is reported.

Two different approaches can be used to get DNN parameters in a quantized format, and Vitis I Quantizer for Tensorflow2 supports both.

Post-training quantization (PTQ)

Post training quantization [20] is a technique which provides the original model training, using then floating-point computation, and then the trained model is *frozen* and its parameter quantized.

This quantization technique, which consist principally on weights and activation functions conversion by 32 bit floating point variables to 8 bit integer ones, introduces a quantization error both on parameters as well as on the activations, resulting in a final accuracy degradation.

Quantization aware training (QAT)

Quantization aware training [21] is another quantization technique which is introduced for improving both speed and memory occupancy performances without having the model accuracy degradation.

QAT works compensating the parameters quantization error by training the DNN using its quantized version during the forward pass.

The Vitis-AI quantizer supports the most common layers used in neural networks, such as convolutional, pooling, dropout and fully connected layers.

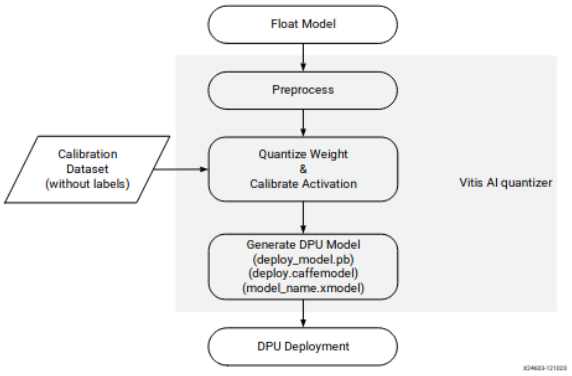

Figure 5.5: Flow of AI quantization

In figure 5.5 the quantization flow can be observed: the Vitis-AI quantizer takes the floating-point model and makes a pre-processing that consists of fold batch-norms and removal of unnecessary nodes of the network, after which it performs the quantization of the weights, biases and activation functions in the required bit dimensions, and finally, after having performed the calibration (using between 100 and 1000 input images as a source), the quantized model is transformed in a DPU deployable model (in .xmodel format in the case of TensorFlow2) which can be implemented in the DPU.
The model thus quantized, as well as the original floating-point one, can be evaluated using a checkpoint file that shows the new dimensions of the model and that gives an estimate of the new accuracy achieved (for the regressor DNN it's expressed in terms of *mae*.

5.1.3 AI Compiler

The Vitis AI Compiler allows to map the quantized DNN model, in a sequence of instructions highly optimized for the DPUs.

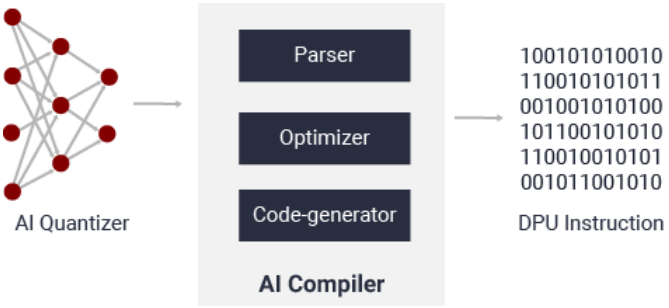

Figure 5.6: Vitis AI Compiler

As shown in figure 5.6, after parsing (i.e. after have analyzed the structure) the topology of the quantized and optimized model it has as input, the compiler generates an intermediate representation (IR) on which it then executes more sophisticated optimizations, such as layer merging, instruction scheduling and more efficient (re)use of on-chip memory.
The compilation is done fot the specific DPU microarchitecture chosen for the algorithm implementation, and Vitis AI provides a list of architectures described in the compilation algorithm (and downloaded together with the Vitis-AI interface).

5.1.4 DPU

Deep-Learning Processing Units (DPUs) [15] are electronic circuits designed for the implementation of deep learning algorithms. They offer high efficiency and good performance like CPUs and GPUs, and generally have a separate data memory and specific instruction set architecture (ISA) which allows for the efficient implementation of many convolutional neural networks.
They require instructions to implement a neural network and accessible memory locations for input images as well as temporary and output data. A program running on the application processing unit (APU) is also required to service interrupts and coordinate data transfers. DPUs are composed of series of efficient and scalable IP cores that represent the ultimate point of the implementation of our algorithms.

Vitis-AI uses a precise scheme to define the names of the different DPUs, each of which shows the different purposes for which it can be used. In particular, the DPU naming convention is shown in the figure 5.7:

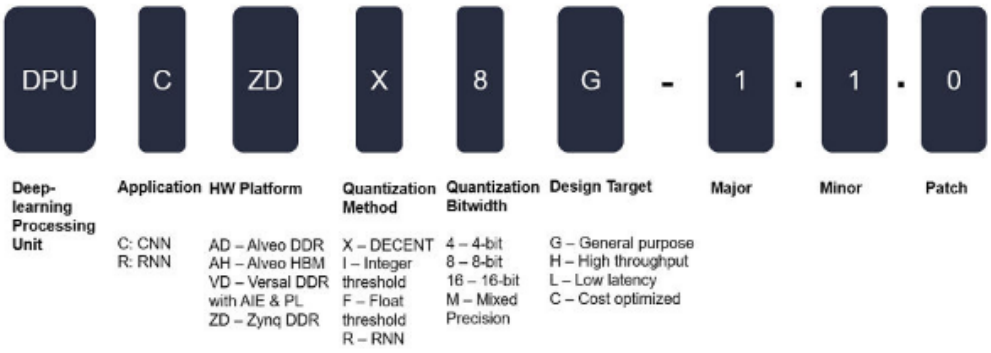

Figure 5.7: DPU Naming Convention.

For our purpose we will focus the analysis on two types of devices:

- DPUCZDX8G, su ZCU102 accelerator card;
- DPUCAHX8H, su ALVEO accelerator card e ALVEO U50 accelerator card.

Considering the notation introduced then the first, the one used for the implementation on ZCU102, is a DPU used for the implementation of DNN on Zynq DDR hardware platforms that uses a decent 8-bit quantization method on a general purpose target, while the second, which implements ALVEO U50 (as visible from the "AH" element in the name), is always used for the implementation of DNN and also uses a decent quantization method that provides a quantized data size of 8 bits with a high throughput target.
Then let's analyze in more detail the structure of the selected DPUs.

ZCU102: DPUCZDX8G

The DPUCZDX8G IP has been optimized for Xilinx MPSoC devices [28]. This IP can be integrated as a block in the programmable logic (PL) of the selected Zynq-7000 SoC and Zynq UltraScale+ MPSoCs with direct connections to the processing system (PS). The configurable version DPU IP is released together with Vitis AI. DPU is user-configurable and exposes several parameters which can be specified to optimize PL resources or customize enabled features. In figure 5.8 its architecture is reported.

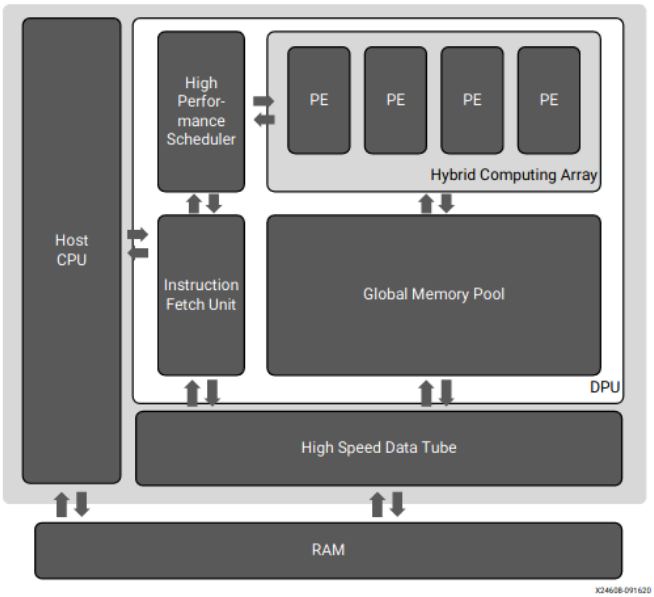

Figure 5.8: DPUCZDX8G Architecture [28]

ALVEO U50: DPUCAHX8H

These Xilinx DPUs are programmable engines optimized for the implementation of convolutional neural networks with high throughput [28]. They consist of a high performance scheduler module, a hybrid computing array module, an instruction fetch unit module and a global memory pool module. They use a special instruction set that allows efficient implementation for many neural networks, and their structure can be shown in the top-level block diagram in figure 5.9.

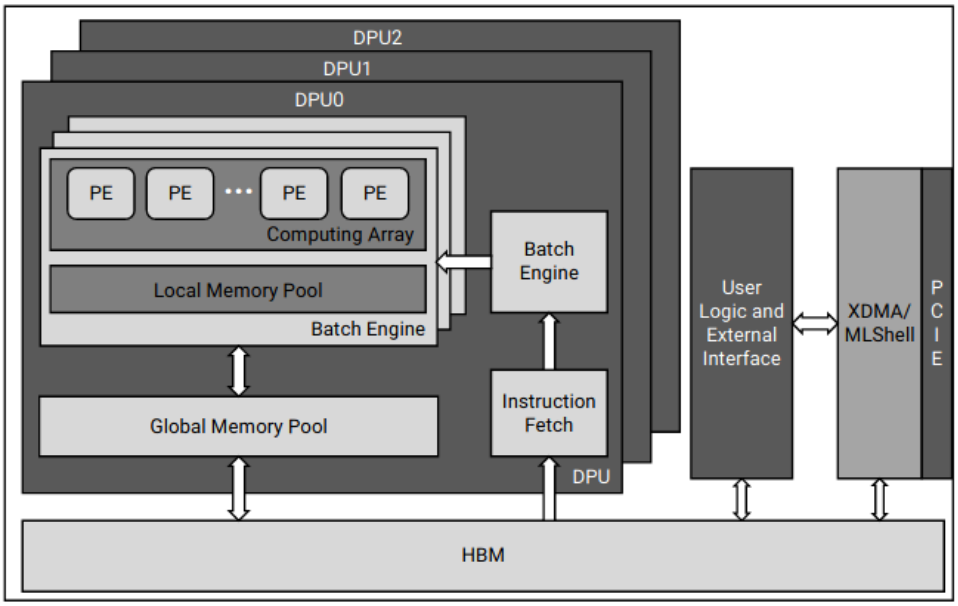

Figure 5.9: DPUCAHX8H Top-Level Block Diagram [28]

Chapter 6

Implementation in hardware of the DNN regression model

In this chapter the followed steps for the DNN regression models described in the previous chapter implementation on hardware devices are reported.

The final goal of this work is the so-called "Feasibility Test", which consists in the study of the possibility of implementing our DNN based trigger algorithms of hardware devices like FPGAs.

Therefore, the final part of this work consists in the effective quantization of the trained DNNs, using TensorFlow libraries, thus evaluating the new quantized models performances, going to see if these quantized models can be compiled on hardware devices using Vitis AI tools; finally going to make an estimate of the processing times that can be reached on this type of devices.

6.1 DNNs quantization using QAT techinque

For the quantization of the DNN models the QAT technique was used, in order to minimize the prediction performances degradation, and the used script is reported and explained in the Appendix B.

As explained in the Section 5.1.2, the quantization aware training technique consists not in the weights and activations of the trained float original model conversion in I8 objects, but in this case the model quantization of is done first its training, thus obtaining a model, called *QAT Model*, containing the performances of the quantized model and at the same time the information of the original non-quantized original model.

For the quantized models obtained then the prediction performances values were compared with the float model ones, both expressed in terms of the Mean Absolute Error value, obtaining the following values:

	10-tracks	2-tracks
Floating Model mae:	0.24	0.42
QAT Model mae:	0.23	0.38

As expected, the prediction performances of the QAT and Float models are almost equal unless statistical fluctuations, and the scatter plots between the predicted values for the two models are showed in figure 6.1 for both 10 and 2 tracks events.
The quantization of DNN models is fundamental as well as advantageous, as it offers a final

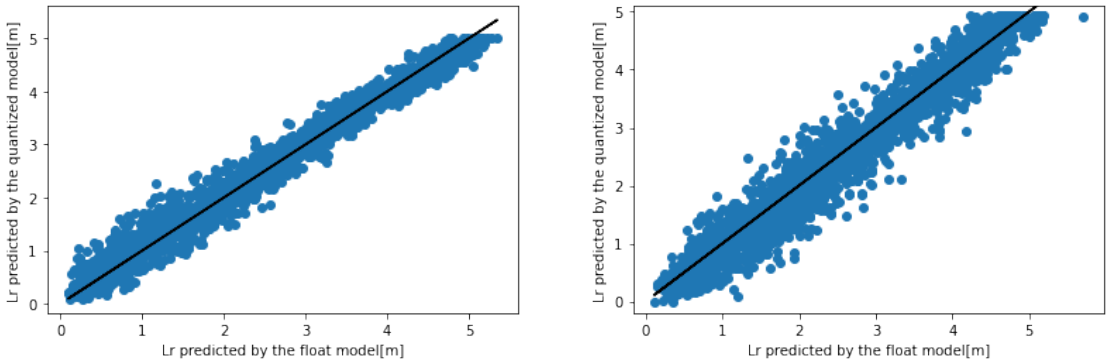

Figure 6.1: Sccatter plots between predicted values by the floating model and values predicted by the QAT one, *on the left* for 10-tracks events and *on the right* for 2-tarcks events.

model of smaller size than the initial one (memory occupancy is reduced by a factor of 4), which can therefore be implemented on FPGA devices that have reduced memory.

In fact it's possible to compute the differences between the memory occupancy for the floating model and for the QAT one, finding the following values:

```
Float model memory occupancy in Mb:      10.96
QAT model memory occupancy in Mb:         2.77
```

where the same result was found for both models, as the difference between the two is in the dataset and not in the layers structure.

Another advantage of the quantization of the model lies in the fact that, in addition to being smaller and therefore implementable on FPGAs, by containing only I8 values, the new model allows use the maximum acceleration of the DPUs, also resulting in a faster data transfer in memory.

Then, given the two QAT Models, we have used Vitis AI Compiler for their compilation for specific targets, compiling the two models for $U50$ and $ZCU102$ DPUs introduced in the Section 5.1.4, obtaining the firmware source file for both the architectures.

After obtaining the technical and physical performances for the quantized models, the last step for the feasibility test is the study of FPGA processing times, in order to observe whether or not these devices are advantageous compared to today's CPU devices used for ALTAS HLT event selection.

6.2 Estimation of FPGA processing times performances

One of the fundamental characteristics of trigger systems for data acquisition in particle experiments is their latency time, which defines their performances and which the physicist would like to be as short as possible. As mentioned before, one of the characteristics for which FPGAs are advantageous devices is their data processing time, which is much shorter than that achieved by CPU and GPU.

In order to study the processing time performances of our algorithms on FPGA devices, we had to make an estimate starting from the comparison with known time values for similar DNN models.

In particular we have done the benchmark of our model on a CPU device and of the quantized one on a GPU device, then comparing the obtained benchmark acceleration values with those provided by Xilinx for models and datasets similar to our ones using the same hardware devices.

The Xilinx benchmarks used are those provided for the CNN VGG16, as it is made up of an architecture similar to our model one, with a dataset composed by images Imagenet 244×244 on which the CNN operates as a 16-class classifier [14]. Then the benchmark values for this architecture are mediated with other similar architecture values, taking then into account the differences in the dataset and in the network structure.

The values provided for the CPU/FPGA and GPU/FPGA inference time ratios were then used to scale our model CPU and GPU inference values, thus obtaining an estimate for our FPGA inference times.

Is important to note that the considered scaling values, obtained comparing the performance of the various devices reported on their data sheets, are given both for the CPU and the GPU for processing 32-bit floating point data.

The CPU and GPU performance values is done taking the floating model and the trained QAT one, both for 2-tracks and 10-tracks events, using a timer to measure the prediction time of the L_r values for N images of the dataset, thus obtaining the processing time per-frame (expressed in seconds) or, inversely, the number of frames processed per second (fps).

The CPU and GPU used for this measurement are the CPU Intel Xeon E5-2698 2.2 GHz and the GPU is the Nvidia Tesla V100 32GB, both on Nvidia DGX-1 512GB RAM server [18] [7]. The FPGAs for which an estimate of the performances is going to be made are those shown in the Section 5.1.4: the ALVEO U50 and the ZCU102 [15].

Then by the Xilinx and the CPU and GPU provided benchmarks the computed scale factors for CPU and GPU with respect to FPGA devices result to be, for the U50:

FPGA vs CPU: x38
FPGA vs GPU: x1.9

Then the DPU U50 results to be about 2 times faster than the GPU and 38 times faster than the CPU, and also the ZCU102 performance values released by Xilinx result to be 4 times worse than the U50 [29].

Subsequently the inference time for our DNN for 8000 images prediction is measured, obtaining, for the 2-tracks events the following values:

	CPU Xeon (*fps*)	GPU V100(*fps*)
Float model (F32)	348	7000
QAT model (F32)	229	5000

Table 6.1: Performance values in terms of processed frames per second for the Intel Xeon E5-2698 2.2 GHz (called *CPU Xeon*) and for the Nvidia Tesla V100 32GB (called *GPU V100*), both for non quantized model (*Float model*) and for the quantized one (*QAT model*).

Which correspond to processing times per frame equal to:

	CPU Xeon (*ms/frame*)	GPU V100 (*ms/frame*)
Float model (F32)	2.9	0.14
QAT model (F32)	4.5	0.19

Table 6.2: Performance values in terms of necessary time for processing a single frame expressed in ms per $frame$.

And, unless statistical fluctuations, similar values were found for 10-track event processing.

Therefore it is possible to observe that in the passage between CPU and GPU the improvement in processing times is up to a factor $O(10)$, but there are no significant differences in the times obtained for pre and post quantization models on both devices.

Then, considering the computed scale factors the resulting estimated value for the DPU ALVEO U50 is 0.06 $ms/frame$, which corresponds to 13300 fps, and for the ZCU102, which is 4 time slower than the U50, a value of 0.28 $ms/frame$ (3325 fps).

Therefore, considering that now in the ATLAS HLT are used only CPU devices [16], which don't have exactly the performance of the one we have used but which we can consider as the so called *state of art*, the results obtained show how, using accelerator hardware devices such as the analyzed DPU U50 and DNN based trigger algorithms, it would lead to a gain of a factor about 38 on HLT processing times.

It's important to consider also that the data sheets reports for the DPU U50 a power consumption of 75 Watts [2], while for the CPU Xeon a value of 135 Watts [18] and for the GPU V100 about 300 Watts [7]; and then we can say that also the performance per Watt consumed will be even higher.

Instead, the ZCU102 results to have worse performances than the U50, but in any case compared to the current devices used in the ATLAS HLT it would lead to higher processing time performances (but slower than the U50 ones), and a lower energy consumption.

Conclusions

In this thesis project we studied the possibility of implementing novel trigger algorithms for the HLT of the ATLAS experiment on commercial FPGA accelerators, proposing a *feasibility test* for the LHC-Run3 and the HL-LHC.

The implemented algorithms exploit the advantages of Deep Neural Networks, i.e. convolutional neural networks with multiple hidden layers able to identify patterns in the input images to solve regression problems (i.e. prediction of a continuous variable). The defined trigger algorithms is then used to predict, for simulated images acquired by the MDT chambers placed in the Inner Barrel region of the Muon Spectrometer, the radial decay length associated with it.

With this type of algorithm we want to select the so-called Long Lived Particles, which are particles with an appreciable lifetimes values considered in many of the beyond Standard Model theories. In fact, the simulated images used was composed of the resulting image arising from two type of processes: $\gamma_d \to \mu^+\mu^-, \pi^+\pi^-$ and $\pi_V \to b\bar{b}$, which corresponds to 2 and 10 visible tracks in the MDT chambers.

Using Tensorflow we have therefore implemented and trained a deep neural network capable of predicting the radial decay length associated with each MDT image, and in order to then implement it on FPGA accelerator devices it was necessary to use to techniques such as quantization.

The quantization is also done using tensorflow; it consists in the weights and activations functions conversion from floating point values at 32 bits to integer values at 8 bits, thus making the model smaller and with variable types easily processed by hardware accelerator devices like FPGAs. We have used a quantization strategy called Quantization Aware Training, where the original model is pre-quantized before the training, thus avoiding a degradation of model prediction performances.

The quantized model performances were then studied, verifying that these were similar to those obtained from the starting floating point model. Furthermore, quantization showed a reduction of the memory occupation of the new model by a factor 4, making it therefore implementable on FPGAs, which have a reduced memory width.
The last step for the implementation on hardware devices was finally done using Vitis AI, a software developed by Xilinx, which is specialized in tools and devices for AI inference acceleration, compiling the quantized model for different target devices: the ZCU102 and the ALVEO U50.

The processing time performances of the final compiled model are then studied. They were were estimated scaling on known benchmarks for similar model implemented on the same harwdare devices, comparing the performance values in terms of frame per second (fps) for specific CPU and GPU devices, both for the floaing original model and for the quantized and optimized one. he obtained performance values in terms of fps for ALVEO U50 and ZCU102 devices showed how the use of FPGAs in the ATLAS HLT with DNN based algorithms can lead to an increase in performance of a gain factor of 38 and to a simultaneous decrease in the power consumption.

Appendix A

RPC and MDT chambers upgrade for HL-LHC

A.1 RPC chambers upgrade

During the HL-LHC phase the RPCs detectors used in the MS will be replaced with detectors always of the same type but with three gas-gaps instead of two, in order to increase the geometric acceptance of the detector passing from a coverage of $\simeq$ 80% to one of 90 % [12]. They will also be associated with innovative and more flexible trigger algorithms for which it will not be required to have 3 hits on all three chamber stations (3/3) but it will be enough to have either 3 hits on 4 different stations or at least one hit on the station in the BI and one on that of the External Barrel (BO) (3/4 + BI-BO) [11].

The new configuration can be observed in figure A.1, where the three dashed arrows represent

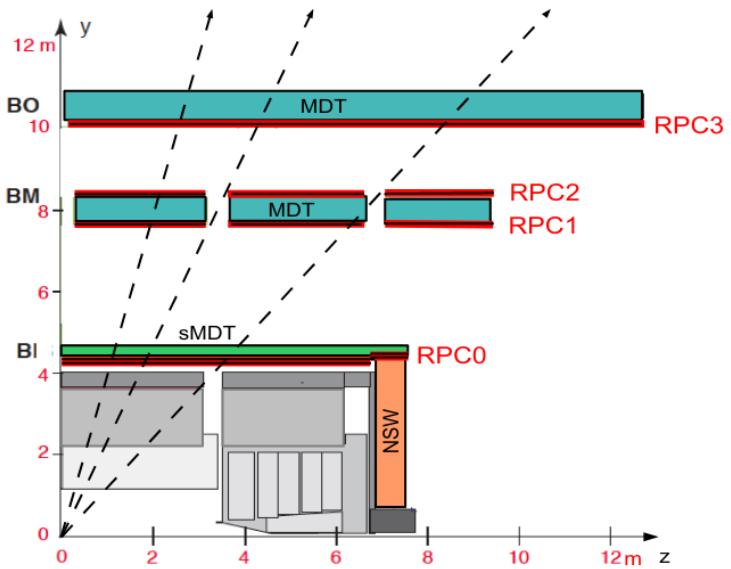

Figure A.1: Sketch of a transverse section of the barrel region with the new RPC station.

the possible muons trajectories: they can cross 4, 2, or 3 RPC chambers. The usage of the requirement 3/4 + BI-BO allows to improve the coverage of the trigger in the regions where no BM RPCs are installed due to the presence of the toroid coils.

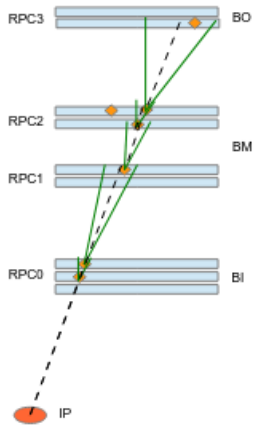

Figure A.2: Illustration of the RPC L0 barrel trigger algorithm.

It's important to consider also that during HL-LHC the RPC chambers will be subjected to a rate which will reach values of about 300 Hz/cm^2, and therefore will be necessary to reduce the gain of these detectors to make them more durable over time, thus lowering the working voltage. This will have consequences on efficiency, because the rate is highly dependent on η

and therefore the efficiency will be reduced up to 35% in the high η regions, for an average decrease of 15%; where however this loss will be compensated by the presence of the new RPC layer placed in the BI, which being of the new generation will have a much better efficiency at lower voltages.

The new L0 trigger algorithm in the barrel region of the muon spectrometer searches for hits between the different RPC planes within pre-defined coincidence windows, which define a maximum distance in ϕ or η for associating hits to the same muon candidate, and is applied independently for the two variables.

The request of the presence of two hits in consecutive layers within a given coincidence windows is then applied to each pair of hit layers, and in each step the center of the coincidence window on the next layer is obtained by tracking the joining between the IP and the hit point on the first layer. Through this algorithm it is therefore possible to measure the muon momenta through the curvature of its trajectory with respect to that delineated by the junction between the hit point and the IP (observing the width of the coincidence window, a threshold on the p_T is obtained), and the resolution on the impulse measurement is limited by the spread of the distribution of the interaction vertex along the beam line and by the spatial resolution of the hits recorded on the planes (figure A.2).

In figure A.4 it's possible to observe the variation of the geometric acceptance of the L0 barrel trigger with this new system of cameras and by varying the trigger requests. It is defined as the fraction of muons present that is accepted by the trigger, and it has been estimated using simulations that predict 100% of the detector efficiency.

Trigger efficiency can be observed in figure A.5, in particular the figure shows the variation of the product between efficiency and acceptance of the L0 barrel trigger with respect to muons reconstructed with $p_T = 25$GeV as a function of η. This quantity is defined as the fraction of reconstructed muons accepted by the trigger and its trend also includes the RPC detector efficiency.

Trigger	Requirement
3/3 chambers	`3[RPC1+RPC2] AND 1[RPC3]`
3/4 chambers	`(3[RPC1+RPC2] AND 1[RPC3]) OR (2[RPC0] AND 3[RPC1+RPC2+RPC3])`
3/4 ch. + BI-BO	`(3[RPC1+RPC2] AND 1[RPC3]) OR (2[RPC0] AND 3[RPC1+RPC2+RPC3]) OR (2[RPC0] AND 1[RPC3])`

Figure A.3: The hit requirements used in different RPC triggers. The left column shows the short name used in the text. The right column gives the coincidence scheme used for the selection logic [11].

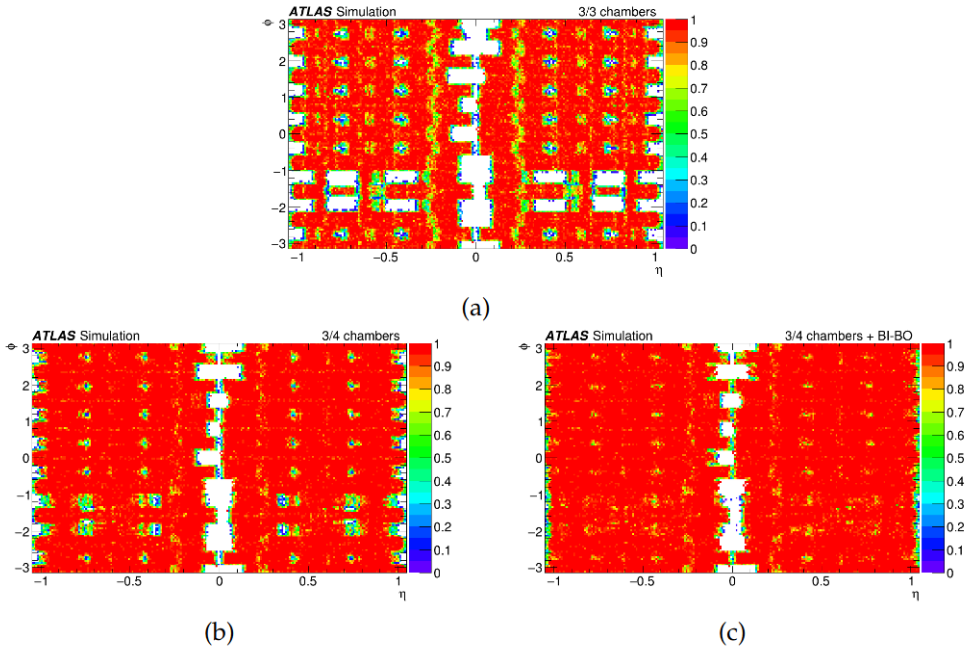

Figure A.4: Variation of the geometric acceptance in the L0 level of the barrel trigger with respect to a muon reconstructed with $p_T = 25$GeV in the plane $\eta - \phi$ in (a) the old configuration 3/3 chambers, (b) the new configuration 3/4 chambers and (c) the new 3/4 chambers + BI-BO configuration. White areas indicate regions with zero geometric acceptance [11].

A.2 MDT chambers upgrade

As mentioned before, the main upgrade of the MDTs will be done with the aim of making more space for the new chamber of the RPCs system; in particular these will be replaced in the BI with new MDTs which are composed of tubes with a smaller diameter (15mm instead of 30mm), in order to obtain more space in the BI for the new RPC layer, to reduce the maximum drift time of the electrons of in order to reduce the dead time of the detector and to reduce the gain losses at rates higher than 500kHz (since the charge density is inversely proportional to the radius of the tubes, by halving the radius then the charge density will be doubled and the loss in gain will results to be high for rate values 8 times greater than the previously supported ones). Also MDT trigger algorithms will be based, for the measurement of the muon p_T, on two variables: the polar angle difference between two tracks in two distinct MDTs (2-station trigger) and, in the case of three tracks present, the distance between the position of a certain track and

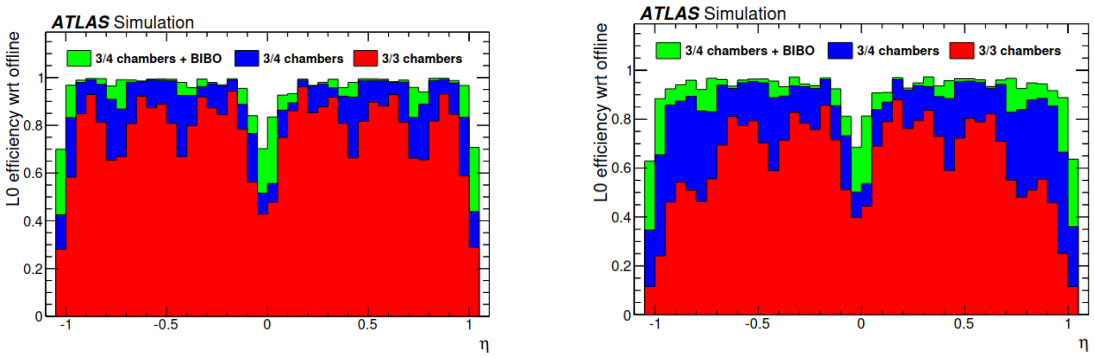

Figure A.5: Efficiency times acceptance of the L0 barrel trigger assuming *On the left* 100% hit efficiency fot the old RPCs and *On the right* the wort-case scenario. The histograms show the efficiency of the existing 3/3 chambers trigger, of the 3/4 chambers trigger including the BI layer, and the additional gain from the BI-BO trigger [11].

the junction of the other two (in the middle chamber it corresponds approximately to the test of the curved trajectory) (3-station trigger). In the images in figure A.6 the two concepts of 2 and 3 station trigger are showed. These two methods allow to obtain a reduction of the muon

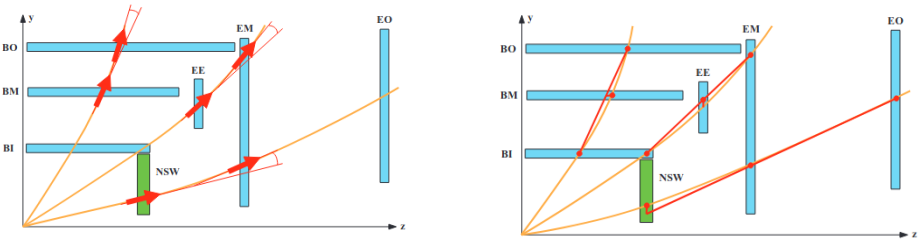

Figure A.6: *On the left*, an image of the 2-station trigger concept, showing examples of track pairs in BM and BO, EE and EM, NSW and EM. *On the right*, an image of the 3-station trigger concept [11].

trigger rate by a factor of about 2 and 3.3 respectively, thus increasing the selectivity of the muons with p_T near the threshold.

Appendix B

Used scripts

Here some of the codes developed for the thesis project are reported. In particular we can find the structure of the DNN built, together with the data modulation made for the training and the training itself, and the script used for QAT of the trained DNN float model.

B.1 Regression DNN code

For the development of the DNN code the Tensorflow and Keras libraries were used. The input size are those of the simulated images of the MDTs, so (20,333,1) where the third dimension means that the images are monochromatic ones. The first step is the definition of Convolutional blocks, i.e. sequences of Conv2D, Activation and eventually MaxPooling layers (figure B.1).
As we can see the third Convolutional block the MaxPooling has not been applied in order not

```
import tensorflow as tf
from tensorflow.keras import datasets, layers, models
from tensorflow import keras

inputs = keras.Input(shape=(20,333,1))

# First convolutional block
x = keras.layers.Conv2D(64, kernel_size=(3,3), padding='same')(inputs)
x = keras.layers.Conv2D(64, kernel_size=(3,3),  padding='same')(x)
x = keras.layers.Conv2D(64, kernel_size=(3,3),  padding='same')(x)
x = keras.layers.ReLU(name='ReLU')(x)
x = keras.layers.MaxPool2D((2,3))(x)

# Second convolutional block
x = keras.layers.Conv2D(128, kernel_size=(3,3), padding='same')(x)
x = keras.layers.ReLU(name='ReLU1')(x)
x = keras.layers.MaxPool2D((2,3))(x)

# Third convolutional block
x = keras.layers.Conv2D(256, kernel_size=(3,3))(x)
x = keras.layers.ReLU(name='ReLU2')(x)
x = keras.layers.MaxPool2D((1,3))(x)

x = keras.layers.Conv2D(256, kernel_size=(3,3))(x)
x = keras.layers.ReLU(name='ReLU3')(x)
x = keras.layers.MaxPool2D((1,3))(x)

# Flattening layer

x = keras.layers.Flatten()(x)
```

Figure B.1: Definition of Convolutional blocks, with their flattening for subsequent Dense Network implementation.

to reduce too much the size of the output x, and the output of the three Convolutional blocks is passed through a Flattening layer in order to insert it into the Dense network.
The Dense network is composed of 2 dense hidden layers each with 1024 neurons, each associated to a ReLu activation layer and a Dropout one, and the last one is composed by 1 neuron (figure B.2).

```
# Dense network
x = keras.layers.Dense(1024)(x)
x = keras.layers.ReLU(name='ReLU4')(x)
x = keras.layers.Dropout(0.2)(x)
x = keras.layers.Dense(1024)(x)
x = keras.layers.ReLU(name='ReLU5')(x)
x = keras.layers.Dropout(0.2)(x)
x = keras.layers.Dense(1)(x)
outputs = keras.layers.ReLU(name='ReLU6')(x)

# Final model definition
model = keras.Model(inputs=inputs, outputs=outputs)
model.summary()
```

Figure B.2: Dense Network and output layer

The output layer consists of 1 neuron with ReLu activation function and its outputs estimate the L_r value associated to the input image configuration. Then the defined model finds out to have (using the *model.summary()* method) the following parameter numbers: Which indicate

```
Total params: 2,871,681
Trainable params: 2,871,681
Non-trainable params: 0
```

Figure B.3: Parameters of the defined DNN

that all the parameters that describe the DNN are *trainable*, i.e. their values can be optimized during the following training phase.

B.2 DNN Training code

The model training is composed of several steps which must be done before the effective training, which consists in a fit between values founded using training and validation dataset.

The first step consists in the training and testing datasets. Considering that the analyzed dataset is called *'dataset'* and that we indicate as *labels* the label column contained in the dataset (composed of L_r, η_T), we have defined (and then reshaped) the different sets of datas in the following way:

```
Lr = labels[:,0]

from sklearn.model_selection import train_test_split
train_to_test_ratio = 0.8

X_train,X_test,Y_train,Y_test = train_test_split(dataset,Lr,train_size=train_to_test_ratio, shuffle=True, random_state=1234)

X_train = X_train.reshape((X_train.shape[0], 20, 333,1))
```

Figure B.4: Definition of the interesting variables L_r in the dataset, splitting of original dataset in train and test ones and reshaping.

Then, after defining the chosen optimizer algorithm, the Adam one, we have compiled the model setting as loss function the Mean Squared Error function (*mse*) and as metric the Mean Absolute Error function (*mae*).

```
LR_ST=1e-3
OPTIMIZER = tf.keras.optimizers.Adam(learning_rate=LR_ST)

model.compile(optimizer=OPTIMIZER,
              loss='mse',
              metrics=['mae'])
```

Figure B.5: Definition of the Adam optimizer algorithm and model compiling.

The last step before the training is the *call backs* definition. It's so important because of call backs help us to identify (and then save) the best trained model for each run. As previously indicate we have used a variable learning rate function for the Learning Rate Scheduler call back and also we have introduced a Model Checkpoint.

```
def lr_decay(epoch):
  if epoch < 5:
    return LR_ST
  else:
    return LR_ST * tf.math.exp(0.2 * (5 - epoch))

lr_scheduler = keras.callbacks.LearningRateScheduler(lr_decay)


model_checkpoint = keras.callbacks.ModelCheckpoint(
        filepath = 'best',
        monitor='val_mae',
        save_weights_only=False,
        save_best_only=True,
        save_freq='epoch')
```

Figure B.6: Call backs definition: Learning Rate Scheduler with variable learning rate and Model Checkpoint.

Finally we have trained the model. The number of epochs reported is the last one used, after several tests with different values it is resulted to be the better one: after about 30 epochs the training and validation trends are in the *plateau region* and loss function and metric values are stable.

```
history = model.fit(X_train, Y_train, epochs=30, batch_size=64,
                    validation_split=0.2, shuffle=True, verbose=1, callbacks=[lr_scheduler, model_checkpoint])
```

Figure B.7: Script for model training

As showed, the value for *validationsplit* variable which defines which portion of the train dataset will be used for validation and then for establish model performances, is set at 0.2, and this means that the 20% of the training dataset will be used for this scope; and the results obtained from the training are those reported in the Section 4.2.

B.3 DNN Quantization Aware Training

Here the script for the quantization of the regression DNN is reported. It starts by taking the best achieved model saved during the training (saved setting the *savebestonly* variable as $True$) and requires to use the Tensorflow model optimization package.

After the floating model loading, then, the second QAT step consists in the creation of the QAT model.

```
import tensorflow_model_optimization as tfmot

model = keras.models.load_model('floating_model.h5')

# quantization aware training
q_aware_model = = tfmot.quantization.keras.quantize_model(model)

q_aware_model.compile(optimizer=OPTIMIZER,
              loss='mse',
              metrics=['mae'])

q_aware_model.summary()
```

Figure B.8: Definition of the interesting variables L_r in the dataset, splitting of original dataset in train and test ones and reshaping.

It's important to notice that the resulting model is quantization aware but not quantized (i.e. the weights are float32 instead of int8). In fact this QAT model contains all the floating model information both with the quantized model prediction performances.

Then the last step is the QAT model training, where the train dataset and the call backs were defined as the same of the floating case, and also the number of epochs is chosen with the same technique.

```
q_aware_model.fit(X_train, Y_train, epochs=30, validation_split=0.2, verbose=1,callbacks=[lr_scheduler, model_checkpoint])
```

Figure B.9: Script for QAT model training

Finally the QAT model is trained and its performances can be evaluated, obtaining the values reported in Chapter 6.

Bibliography

[1] G. Aad et al. "The ATLAS Experiment at the CERN Large Hadron Collider". In: *JINST* 3 (2008), S08003. DOI: 10.1088/1748-0221/3/08/S08003.

[2] Xilinx Vitis AI. URL: https://www.xilinx.com/products/design-tools/vitis/vitis-ai.html.

[3] Juliette Alimena et al. "Searching for long-lived particles beyond the Standard Model at the Large Hadron Collider". In: *Journal of Physics G: Nuclear and Particle Physics* 47.9 (Sept. 2020), p. 090501. ISSN: 1361-6471. DOI: 10.1088/1361-6471/ab4574. URL: http://dx.doi.org/10.1088/1361-6471/ab4574.

[4] Borga Andrea et al. "The C-RORC PCIe card and its application in the ALICE and ATLAS experiments". In: *Journal of Instrumentation* 10 (Feb. 2015). DOI: 10.1088/1748-0221/10/02/C02022.

[5] 16th July 2020 ATLAS Collaboration. *Keeping the ATLAS Inner Detector in perfect alignment*. URL: https://atlas.cern/updates/experiment-briefing/inner-detector-alignment.

[6] 30th May 2016 CERN European Organization for Nuclear Research. *The High-Luminosity LHC (HL-LHC) project*. URL: https://cds.cern.ch/record/2199189/files/English.pdf.

[7] Nvidia Cloud and Data Center. *GPU NVIDIA TESLA V100 TENSOR CORE*. URL: https://www.nvidia.com/it-it/data-center/tesla-v100/.

[8] ATLAS Collaboration. *ATLAS EXPERIMENT - Public results Run 2*. URL: https://twiki.cern.ch/twiki/bin/view/AtlasPublic/LuminosityPublicResultsRun2#Luminosity_summary_plots_for_201.

[9] The ATLAS Collaboration. "ATLAS Muon Detector Commissioning." In: (30th July 2021). DOI: https://arxiv.org/pdf/2103.01029.pdf.

[10] The ATLAS Collaboration. "Operation of the ATLAS Trigger System in Run 2." In: (9th October 2020). URL: https://arxiv.org/pdf/2007.12539.pdf.

[11] The ATLAS Collaboration. "Technical Design Report for the Phase-II Upgrade of the ATLAS Muon Spectrometer." In: (2017). URL: https://cds.cern.ch/record/2285580/files/ATLAS-TDR-026.pdf.

[12] The ATLAS Collaboration. "Technical Design Report for the Phase-II: Upgrade of the ATLAS Trigger and Data Acquisition System." In: (15th June 2018). URL: https://cds.cern.ch/record/2285584/files/ATLAS-TDR-029.pdf.

[13] The Atlas Collaboration. "The ATLAS experiment at the CERN Large Hadron Collider." In: *Journal of Instrumentation* (2008), Chapter 6. URL: https://jinst.sissa.it/LHC/ATLAS/ch06.pdf.

[14] Vitis AI User Documentation. *U50/U50LV Performance*. URL: https://www.xilinx.com/html_docs/xilinx2019_2/vitis_doc/drj1583902382211.html.

[15] Vitis AI User Documentation. *Zynq DPU v3.3 IP Product Guide*. URL: https://china.xilinx.com/html_docs/vitis_ai/1_4/dpu_over.html.

[16] for the ATLAS muon collaboration E. Diehl. "ATLAS Muon Detector Commissioning." In: (27-31 July 2009). URL: https://arxiv.org/pdf/0910.2767.pdf.

[17] Lyndon Evans and JINST 3:S08001 Philip Bryant. "LHC Machine." In: (2008). URL: https://iopscience.iop.org/article/10.1088/1748-0221/3/08/S08001/pdf.

[18] Intel. *Intel® Xeon® Processor E5-2698 v4 Specifications*. URL: https://ark.intel.com/content/www/us/en/ark/products/91753/intel-xeon-processor-e5-2698-v4-50m-cache-2-20-ghz.html.

[19] "Long-Lived Particles at the Energy Frontier: The MATHUSLA Physics Case." In: (5th March 2019). URL: https://arxiv.org/pdf/1806.07396.pdf.

[20] TensorFlow Model Optimization. *Post-Training Quantization*. URL: https://www.tensorflow.org/model_optimization/guide/quantization/post_training.

[21] TensorFlow Model Optimization. *Quantization Aware Training*. URL: https://www.tensorflow.org/model_optimization/guide/quantization/training.

[22] 14th November 201 Paolo Lengo for Infn ATLAS Seminars. *The ATLAS Calorimeters*. URL: http://people.na.infn.it/~elly/TesiAtlas/SeminariAtlas/Detector/IengoAtlasCalo.pdf.

[23] 18th August 2018 SuperDataScience Team. *Convolutional Neural Networks (CNN): Step 3 - Flattening*. URL: https://www.superdatascience.com/blogs/convolutional-neural-networks-cnn-step-3-flattening.

[24] 18th August 2018 SuperDataScience Team. *Convolutional Neural Networks (CNN): Step 4 - Full Connection*. URL: https://www.superdatascience.com/blogs/convolutional-neural-networks-cnn-step-4-full-connection.

[25] G.Apollinari O.Bruning T.Nakamoto and L.Rossi. "High Luminosity Large Hadron Collider HL-LHC." In: (). URL: https://arxiv.org/pdf/1705.08830.pdf.

[26] *Technical Design Report for the Phase-II Upgrade of the ATLAS Muon Spectrometer*. Tech. rep. Geneva: CERN, Sept. 2017. URL: https://cds.cern.ch/record/2285580.

[27] "Technical Design Report for the Phase-II Upgrade of the ATLAS Muon Spectrometer". In: (15th December 2017). URL: https://cds.cern.ch/record/2285580/files/ATLAS-TDR-026.pdf.

[28] Xilinx. *Vitis AI User Guide*. 3th February 2021.

[29] Xilinx. *ZCU102 Evaluation Board User Guide*. URL: https://www.xilinx.com/support/documentation/boards_and_kits/zcu102/ug1182-zcu102-eval-bd.pdf.