Improving e & γ reconstruction at ATLAS using a convolutional neural network

Energy Regression at ATLAS using Machine Learning

Master Thesis in Computational Physics
Written by *Malte Algren*
2020-2021

Supervised by
Troels C. Petersen

University of Copenhagen

NAME OF INSTITUTE:	Faculty of Science
NAME OF DEPARTMENT:	The Niels Bohr Institute
AUTHOR(S):	Malte Algren
EMAIL:	`malte__algren@hotmail.com`
TITLE AND SUBTITLE:	Improving e & γ reconstruction at ATLAS using a convolutional neural network - Energy Regression at ATLAS using Machine Learning
SUPERVISOR(S):	Troels C. Petersen
HANDED IN:	2020-2021
DEFENDED:	2020-2021

NAME ______________________

SIGNATURE ______________________

DATE ______________________

Abstract

For the energy calibration of electrons and photons, the ATLAS experiment employs a boosted decision trees (BDT) that utilizes scalar variables extracted from the detector to calibrate the energy. However, the variables are a summary of the spatial and energy properties of the Inner Detector (ID) and the Electromagnetic Calorimeter (ECAL) and not all information from the ID and ECAL remains within these variables, particularly the imagery of the ECAL. This imagery lead to the development of a convolutional neural network model called *DeepCalo* that was be able to take advantage of the additional ECAL images. *DeepCalo* achieved a general performance improvement of $22.4 - 25.9\%$ ($12.4 - 18.3\%$) for electrons (photons) on Monte Carlo simulated Data (MC). *DeepCalo* was also tested on measured data from the ATLAS experiment. Here, it archived a general improvement of 9.4% (2.4%) for electrons (photons). The channels tested were $Z \to ee$ (MC/data), $H \to \gamma\gamma$ (MC) and $Z \to \mu\mu\gamma$ (MC/data). Furthermore, *DeepCalo* achieved an improvement independent of detector region and energy range, expect for high $|\eta|$ photons, and it attained robust performance at increasing pileup.
A method for unlocking the potential to train directly on data was also developed, where training with a mixture of MC and data samples for $Z \to ee$ revealed a general improvement of $22.1 - 25.3\%$ for electrons in MC, and an improvement of 18.3% for data.
The *DeepCalo* model was also able to predict the uncertainties on its predictions and these could in turn be used for estimating the uncertainties on invariant masses. This increase in reconstruction performance would help in the discovery of proposed Higgs decays, where low statistics pose a problem.

Acknowledgements

I would like to thank my main supervisor Troels C. Petersen. We have now worked together for two years, and it has been a pleasure working with him both academically and socially.
I would also like to thank Daniel Nielsen, Helle K. Leerberg and Lukas Ehrke for helping me in the early stages of the project and a special thanks to Frederik Faye, who started the development of *DeepCalo* and was a big help when I started.
A big thanks to my office mates, Christian Michelsen, Sara D. Pinholt, Aske Rosted, Mads J. Storr-Hansen and Rasmus F. Ørsøe for creating a fun work atmosphere.
Lastly, I would like to thank Anna Xiao Tan for her comments on my work and her support during this project.

© 2021 Malte Algren
MSc Thesis,
Niels Bohr Institute, University of Copenhagen

Thesis submitted October 1 st 2021, for the completion of the degree Master of Science (MSc) in Computational Physics. LaTeX template: A Tufte-Style Book TUFTE-LATEX.GITHUB.IO/TUFTE-LATEX/.

First printing, October 2021

Introduction

The purpose of this thesis is to develop a Machine Learning algorithm that improves the energy reconstruction of particles at the A Toroidal LHC ApparatuS (ATLAS) experiment at the Large Hadron Collider (LHC). The current reconstruction algorithm, $E\text{calib}^{(BDT)}$, does not take the internal layout of the ATLAS detector into consideration, whereas the algorithm developed in this thesis will. The developed algorithm is named *DeepCalo* and is a convolutional neural network (CNN) usually applied in visual analysis of imagery. It will predict the energies of electrons or photons with increased precision compared to current methods. It is very beneficial for the ATLAS experiment as a whole to improve the reconstruction of electrons and photons, as new physics discoveries require high precision in the energy reconstruction.

The first chapter will give a brief introduction of the current theory explaining the behavior of elementary particles, the Standard Model (SM). Afterwards, a detailed description of The Large Hadron Collider (LHC) and ATLAS experiment will be presented. Here, the emphasis will be on the ATLAS experiment, as its layout has motivated the analysis and construction of *DeepCalo*. Lastly, we will give a more elaborate motivation for the analysis than given above. *The second chapter* will contain an introduction to Machine Learning (ML) and explain the fundamentals of Network-based ML with a focus on convolutional neural networks. *The third chapter* will explore the variables selected for *DeepCalo* and introduce the data pipeline from xAOD or DxAOD to `.tfrecord` files. *Chapter four* will describe the development of the *DeepCalo* architecture and the optimization of it to find the most accurate model. *Chapter five & six* will evaluate the performance of *DeepCalo* on electron ($Z \to ee$) and photons ($H \to \gamma\gamma$ and $Z \to \mu\mu\gamma$), respectively. Here, the performance for E_T, $|\eta|$ and $\langle\mu\rangle$ will be evaluated. Performance of the *DeepCalo* on both Monte Carlo simulated data (MC) and real world data from ATLAS will be assessed. Note that real world data from ATLAS will be referred to simply as Data with a capitalized D throughout the thesis. *Chapter seven* will be a discussion on the performance of *DeepCalo*. It will contain a discussion of the metrics applied to Data and the shortcomings of *DeepCalo*, with suggestions on how performance may be improved. *Chapter eight* is the final chapter. It contains the conclusion of the thesis with a summary of the performance of the different *DeepCalo* models.

Contents

1 Particle Physics

This chapter will introduce the underlying fundamentals of particle and detector physics. It will encompass only that which gives the reader the necessary knowledge to understand the context of this thesis and motivation for the analysis. For an in-depth introduction into particle physics, please refer to [20] and [34].
The first part of this chapter, section 1.1, will elaborate on the Standard Model (SM) and the elementary particles within. Most of this section is adapted from [20] and [34]. Next up is section 1.2 and 1.3, which will lay the foundation for understanding the ATLAS Experiment and how fundamental properties of particles are measured and reconstructed. Lastly, section 1.4 will motivate the analysis and focus of this thesis.

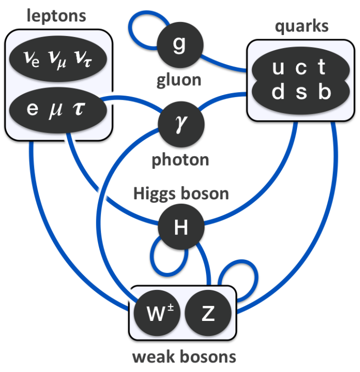


Figure 1.1: The figure shows interactions between fermions-bosons and bosons-bosons of the SM particles.

1.1 The Building Blocks of the Universe

The holy grail of particle physics is the theory of the SM. Developed in the 1970s, the SM is currently the best description of the elementary forces governing our universe. The SM assumes 17 fundamental particles divided into two main categories, namely fermions with spin $\pm 1/2$ and bosons with spin $1 \vee 0$. An overview of the SM particles can be seen in figure 1.2.
The fermions consist of quarks, leptons, and their anti-particle counterparts. A specific combination of quarks make up the hadrons[1] eg. the proton is composed of *uud* quarks. Quarks have an electrical charge of $q = -1/3 \vee 2/3$, whereas leptons have a -1 charge. All fermions have to obey the Pauli exclusion principle[2], yet protons consists of *uud* quarks. This is due to spin being able to have direction $\pm$ e.g. $s_u s_u s_d = \uparrow\downarrow\uparrow$.
At last, the bosons are the force carriers that regulate the behavior of the particles. The connection between the forces can be seen in figure 1.1. Bosons are divided into subgroups depending on how and on what they exert their forces on. The subgroups are listed below.

[1] There are two types of hadrons. The mesons, which are composed of two quarks (an quark and anti-quarks) and the baryons composed of three quarks. All hadrons have integer charge.

[2] Two or more identical fermions are not allowed to occupy the same quantum state in a system.

- **Strong force**: Exerted by the massless gluons. It is the force binding quarks together in a nucleus. However, the internal connection and amount of gluon within hadrons are unknown, making hadrons internal structure chaotic.

- **Weak force**: Exerted by the massive $W^\pm$ and Z bosons. The weak force art on the fermions, and is able to convert them eg. convert

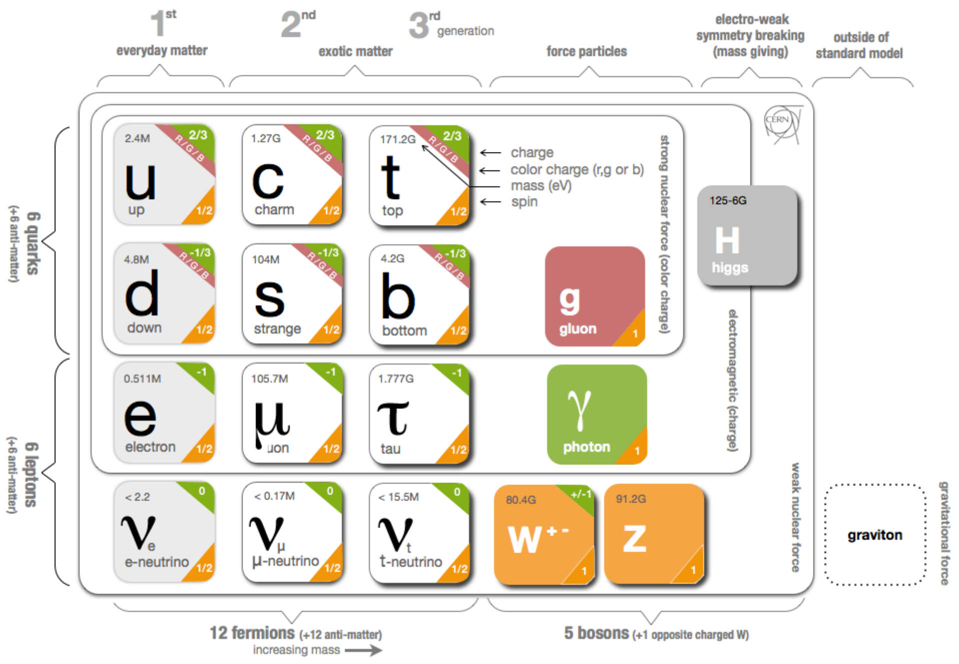

1st everyday matter
2nd exotic matter
3rd generation
force particles
electro-weak symmetry breaking (mass giving)
outside of standard model
2.4M 2/3 R/G/B u up 1/2
1.27G 2/3 R/G/B c charm 1/2
171.2G 2/3 R/G/B t top 1/2
charge
color charge (r,g or b)
mass (eV)
spin
4.8M -1/3 R/G/B d down 1/2
104M -1/3 R/G/B s strange 1/2
4.2G -1/3 R/G/B b bottom 1/2
g gluon 1
125-6G H higgs
0.511M -1 e electron 1/2
105.7M -1 μ µon 1/2
1.777G -1 τ tau 1/2
γ photon 1
< 2.2 0 ν e-neutrino 1/2
< 0.17M 0 ν μ-neutrino 1/2
< 15.5M 0 ν t-neutrino 1/2
80.4G +/-1 W+- 1
91.2G Z 1
graviton
strong nuclear force (color charge)
electromagnetic (charge)
weak nuclear force
gravitational force
6 quarks (+6 anti-matter)
6 leptons (+6 anti-matter)
12 fermions (+12 anti-matter)
increasing mass
5 bosons (+1 opposite charged W)

Figure 1.2: Overview of the Standard model from [36]. The charge, color, mass and spin of the particles can all be seen in the figure as well as groups and subgroups.

a proton to a neutron, by exerting a electron and anti-neutrino to keep the charge constant.

- **Electromagnetic force**: It is exerted by the massless photon (γ) that interacts with all particles with an electric charge.
- **Gravitation force**: Gravitons have not been observed yet, but the force should act on all particles with mass. As seen in figure 1.2, gravitons are not described by the SM.

Lastly, the Higgs boson (H) does not exert a force but interacts with particles of mass through a field called the Higgs field, see figure 1.1. The neutrinos[3], the u,d-quarks, the electrons, the glouns and the photons are stable particles while others will eventually decay to stable particles. By isolating the byproduct of a decay, the parent particle can be found, e.g. the Higgs can decay to two photons ($H \rightarrow \gamma\gamma$), where the combined invariant mass of the photons will result in the Higgs mass.

[3] However, neutrinos will oscillate between types.

1.1.1 *Properties of decay*

Advances in particle physics have made new studies in the former mentioned particles more difficult. Particle colliders and detectors are increasing evermore in size to challenge the SM at higher energies. Due to the short lifetime of most SM particles, they cannot be studied directly, but only by their byproduct. In the case of a two-particle byproduct (eg. $H \rightarrow \gamma\gamma$), the invariant mass of the parent particle[4] is given by

[4] Parent particle is the particle that decays into the byproduct.

$$M^2 = 2p_{T1}p_{T2}\left(\cosh(\eta_1 - \eta_2) - \cos(\phi_1 - \phi_2)\right), \tag{1.1}$$

with p_T being the momentum of the byproduct (when $E >> m$ and the particles are massless, $E_T = p_T$ holds), and both η and ϕ being geometric properties, that will be expanded in section 1.2. Due to Heisenberg's uncertainty principle, the mass of particles can fluctuate

$$\Delta E \Delta t \geq \hbar/2 \Leftrightarrow \Delta m \Delta t \geq \hbar/2. \tag{1.2}$$

This in turns, means that the shorter lifetime a particle has, the less certain is its mass, which gives rise to a width in the invariant mass of the particles. A Breit-Wigner distribution (BW) is used to describe this distribution. The equation for BW can be seen in equation 1.3.

$$\frac{dN}{dm} = \frac{N_{tot}}{\pi} \frac{\Gamma/2}{\Gamma^2/4 + (m - m_0)} \tag{1.3}$$

where $\Delta m \approx \Gamma$. For decays of unstable particles, there will often be multiple decay channels. The ratio of decays in the ith channel is called the branching ratio[5], Br_i and with a partial width of $\Gamma_i = Br_i\Gamma$. In chapter 5 and 6, the BW convoluted with a Crystal Ball distribution (BW$\otimes$CB) will be used to fit the resonance of the Z-boson in both Monte Carlo simulated data and real data measured at ATLAS. The resonance must follow a BW, but because the real world is noisy and the detector has imperfections (see section 7.5 in the discussion), the convolution with CB must describe these impurities. The effects of the CB convolution can be seen in section 9.1.1 along with the expression for CB.

[5] The branching ratio is the fraction of particles that has a specific decay channel.

1.2 *The Large Hadron Collider (LHC)*

The Standard Model has been a breakthrough in particles physics, promising to unify properties of fundamental particles. Nevertheless, it is still a theory that must be experimentally tested and challenged. The theory contains parameters that need to be experimentally measured and, hence, countless experiments have been made to determine the parameters or to disprove the theory.
Parameters and postulations from the SM require testing at very high energies. Consequently, the circular Large Hadron Collider (LHC) at CERN was constructed. With its circumference[6] of 27 km, it was able to observe the Higgs boson (edicted by SM) at 125 GeV [8] and [9] in 2012. Although the Higgs was discovered in 2012, much of the analyzes, at the LHC today, are focused on confirming the several decay channels of the Higgs. However, due to the small branching ratio of the channels, this is very difficult and require large amount of statistics, which is dependent on the reconstruction capabilities of the experiment.

[6] Large particle accelerators are requied to have a large circumference, otherwise all the energy will be lost to synchrotron radiation, when bending charged particles. This is not a problem in linear accelerators.

The LHC consists of four main detector experiments: the CMS, ALICE, LHCb, and ATLAS, where each detector has a different composition and is optimized for different physics research. This thesis will focus on the ATLAS Experiment, and a brief introduction to the experiment will follow in the section below. The LHC has had two runs, *Run* 1 (2009-2013) and *Run* 2 (2015-2018). At the time of writing, it is closed but will open for *Run* 3 in 2021. The years of shutdowns are due to upgrades and maintenance of the collider and detectors. Some of the important specifications of LHC and ATLAS are

- Center of mass energy (CoM) at $\sqrt{s} = 14$ TeV can be achieved in proton-proton (pp) collisions.
- Bunch crossing are due every 25 ns with bunch sizes upwards of 10^{11} particles.
- A luminosity[7] at 146.9fb^{-1}.

[7] Luminosity is a measure of interactions per bunch crossing. If the luminosity increases, the amount of particles that collide do as well. Inverse femtobarn ($(10^{-15})^{-1}$) is a typical unit when measuring the number of particle collision events.

Now, with further upgrades during the shutdown between *Run* 2 and 3, the LHC and ATLAS Experiment are getting ready for High-Luminosity-LHC (HL-LHC)[25]. It seeks to increase the luminosity at the LHC to 4000 fb^{-1}, which would be devastating if the reconstruction algorithms do not follow.

This thesis will focus on the proton-proton (pp) collision at ATLAS, which is a very complicated phenomenon. Due to the proton structure (explained above), a pp collision resembles collisions between two filled trash cans, as the internal structure is unknown, and a spray of particles will fly out as they collide. Lucky, these particles can be studied. With the high center-of-mass energy at the LHC, a pp collision can create many interesting events for a physics analysis. However, this high center-of-mass energy and increase in luminosity comes with a caveat. A large number of particles will be created at each crossing, and most of these are not of interest. These events will contribute to the *pileup*[8] $\langle \mu \rangle$, which can pollute interesting particle phenomena. With a rise in luminosity, $\langle \mu \rangle$ will increase and challenge existing analytic algorithms. In addition, protons rarely hard-scatter[9], giving rise to high transverse momentum (p_T), but rather often occur with an angle, producing events with high pseudorapidity η, which can lead to slightly different behavior than hard-scattered ones.

[8] Pileup is noise (from other particles) within the cluster of the particle. Quantified as the average interactions per crossing. With the increase in luminosity, the pileup will increase as well.

[9] Where the protons collide with each other head on.

1.2.1 *A Toroidal LHC ApparatuS*

For further information about the ATLAS Experiment see [7] and [13]. The ATLAS detector is the largest general-purpose detector at LHC, able to analyze pp and heavy-ion collisions. It is built in cylindrical layers centered around the LHCs beam crossing. Before elaborating further on the setup of the ATLAS detector, a brief introduction in the underlying coordinate system and geometric will follow. In Cartesian coordinates, the origin lies at the interaction point[10] in the center of the detector. The z-axis points along the beam axis while the x-axis points into the center of the LHC and the y-axis points orthogonal upwards towards the sky. Due to the circular symmetry, spherical coordinates are used instead: the azimuthal angle $\phi = [0, 2\pi]$ in the xy-plane, and the polar angle $\theta = [0, \pi]$ for the zy-plane, and rapidity[11] for velocity. However, protons consist of sub-

[10] The interaction point is at the beam crossing, where the two particle beams are crossed.

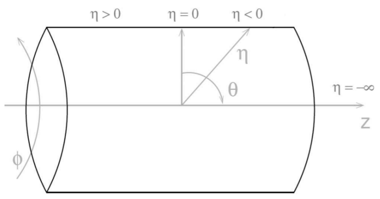
[11] Rapidity given as $\omega = \frac{1}{2} ln \left(\frac{E+p_z}{E-p_z} \right)$.

structures[12] with a non-uniform internal energy distribution, and because of this, the collisions will often be boosted[13]. Therefore, pseudorapidity is used instead of rapidity to describe the particle position in the zy-plane. Pseudorapidity is defines as $\eta = -\ln(\tan(\frac{\theta}{2}))$. When describing the position of particles, ϕ and η will be used. η and ϕ are illustrated in figure 1.3.

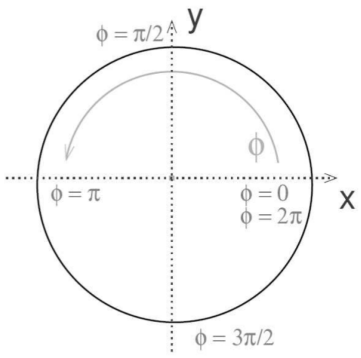
[12] Namely, quarks and gluons.

[13] If one of the particle in the collision has more energy than its counterpart, the byproduct emitted will have the same direction as the boosted particle.

An overview of the ATLAS detector is illustrated in figure 1.4. It consists of layers upon layers of advanced measuring equipment designed to analyze the outcome of a collision. The ATLAS detector is divided in 4 main sections. From the innermost section, closest to the beam-pipe, the Inner Detector (ID) is located. On top of the ID is the electromagnetic calorimeter (ECAL), followed by the hadron calorimeter (HCAL). The last section is muon spectrometer (MS).
In the following section, an in-depth description of the detector will be given. It will place emphasis on the ID and ECAL, since only these sections are used in the analysis. Most of the detector layers are separated into barrel and end-caps. The barrel is a cylinder enclosing the beam pipe, and centered at the beam crossing. The end-caps close the cylinder at each end, like a lid of a Pringles® can. The electronics from each section of the barrel and end-caps has to be powered, cooled, heated and send out data. This is carried out using service wiring routed in-between the barrel and end-caps. This region is called the crack region and has lower resolution and increased material budget compared to the rest of the detector. It will challenge the reconstruction within this region. This will be touched upon in the following sections.
The ATLAS detector can be viewed as a large camera with multiple lenses taking images of the particle properties and behaviors. It is these images that will be used in the analysis.



(a) z-y plane of the ATLAS detector.



(b) x-y plane of the ATLAS detector.

Figure 1.3: The figure shows the geometry of the ATLAS detector. It shows the range of η, ϕ and θ. The cylinder (*top figure*) and circle (*bottom figure*) indicate the ATLAS detector. Figure from [40].

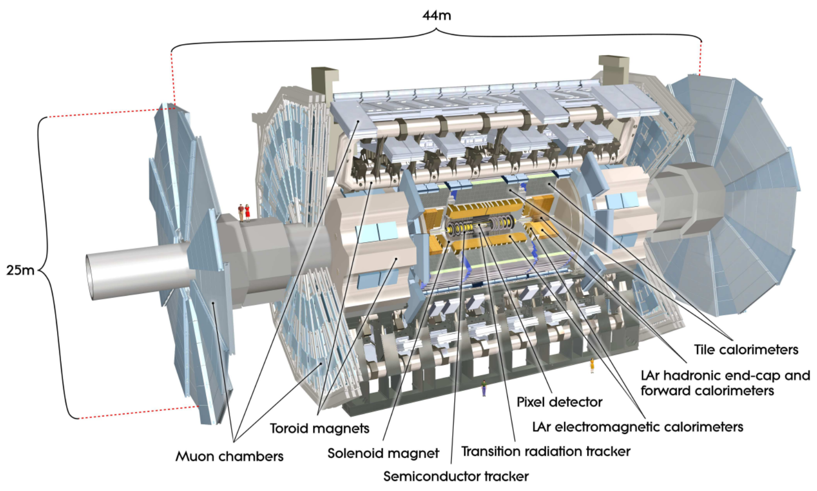

44m
25m
Tile calorimeters
LAr hadronic end-cap and forward calorimeters
Pixel detector
LAr electromagnetic calorimeters
Toroid magnets
Muon chambers
Solenoid magnet
Transition radiation tracker
Semiconductor tracker

Figure 1.4: An overview of the ATLAS detector with labels on the central detector parts. Many of these parts will be elaborated on in the following sections. The figure is from [13].

1.2.2 *The Inner Detector (ID)*

The Inner Detector is the first lens, taking images of collision. The IDs' purpose is to reconstruct the tracks left by charged particles. The ID has a radius of 1.15 meter and is located 5 centimeters from the collision center. A fine detector granularity is required to measure momentum and trajectory accurately. The ID measures tracks within $|\eta| < 2.5$ and $p_T > 0.5$ GeV. It is enclosed in a 2 Tesla magnetic field created by a solenoid magnet located between the ID and ECAL. It is able to bend track of charged particles, thus measuring their momentum[14]. The ID is built so only a small fraction of the particle energy is deposited in the ID, with most of the energy is saved for the ECAL.

[14] The trajectory radius of a charged particle in a magnetic field is given by $r = \frac{p_T}{qB}$, where p_T is the transverse momentum, q is the charge and B is the size of the magnetic field.

The ID is composed of three sub-detectors: the Pixel Detector, the Semiconductor Tracker (SCT), and the Transition Radiation Tracker (TRT). All sub-detectors are separated into barrel and end-caps, as seen in figure 1.5.

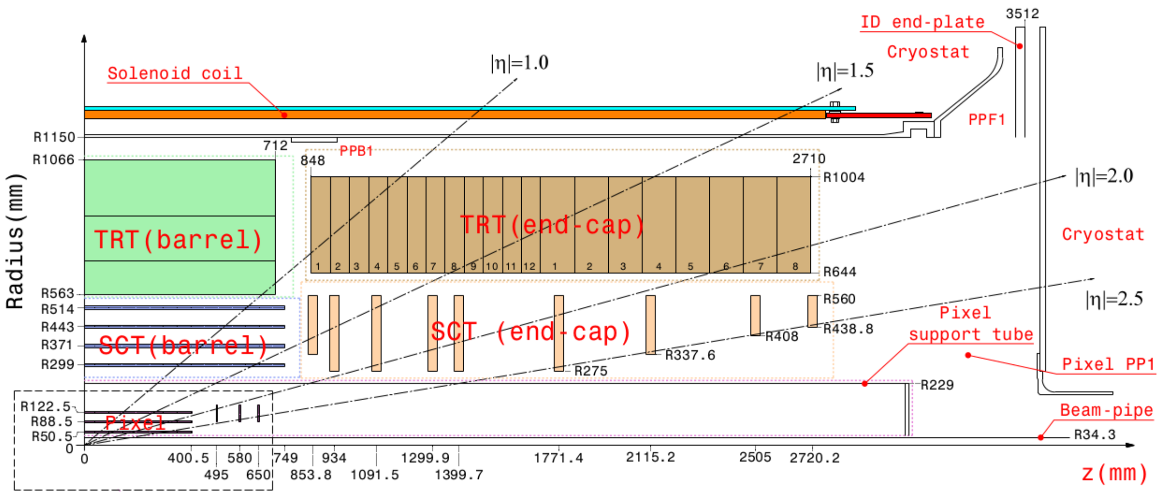


Figure 1.5: Schematic overview of the barrel layers within the ID before LS1. The figure is from [13]. The top figure shows the barrels and end-caps of the ID with all its layers and their $|\eta|$ coverage. Below that figure, a zoom-in of the pixel detector and a table displaying the radius and height of the layers can be seen.

Pixel Detector

The innermost layer of the ID is the *Pixel Detector*. It is built of semiconductors designed to detect charged particles. Originally, the Pixel Detector consisted of three layers in the barrel and end-caps, but during Long Shutdown 1 (LS1 2013-2014), an additional layer, Insertable B-Layer (IBL), was added, which was intended to better identify b/τ quarks[35] (because of their long lifetime.). However, this is not of interest for the analysis in this thesis. The three original layers can be seen in figure 1.5. Due to its short distance to the interaction point, the components of the Pixel Detector are required to meet strict specifications on radiation hardness and resolution.

The Pixel Detector consists of 1736 identical with a total of 92 million pixels. Each pixel has a size of $50 \times 250\mu m^2$ for IBL and $50 \times 400\mu m^2$ for the rest and covers $|\eta| < 2.5$ The concept behind the pixel sensors is similar to the one of solar cells. Both use liberated electrons to create a current. Semiconductors are useful as they have a small band gap of 1.12 eV between the valence band and conduction band. Thus, the price for liberating an electron is only 3.6 eV[15]. This is important, as only a minimum amount of energy must be deposited in the ID. Due to the voltage over the sensors, the liberated electrons will run as current moving toward the read-out channel. The reconstruction will back-track the current, and find the electron-hole pair to calculate the path of the charged particle. This concept follows condensed matter physics and is the approach utilized by all semiconductor-detectors at ATLAS [20]. Due to the number of pixels, the Pixel detector can track charged particles with very high precision, but due to its high price and material budget, semiconductor modules are only used in the Pixel detector and Semiconductor Tracker.

[15] For a noble gas the price is about 30 eV, and it is important to reduce the amount of energy deposit in the ID [20].

Semiconductor Tracker

The next section of the ID is The Semiconductor Tracker (SCT). The SCT is also composed of semiconductors, a total 4088 modules, but these are silicon micro strips with a larger footprint than the pixels in the Pixel Detector. The barrel of the SCT has 4 layers while the two end-caps have 9 layers. The strips of the barrel have a footprint of 12cm $\times$ 285μm and are tilted 40 $m$rad relative to each other to improve resolution. The barrel layers have a resolution of $17 \times 580\mu\text{m}^2$ in $(R - \phi) \times z$ and the end-caps have a resolution of $10 \times 115\mu\text{m}^2$ in $(R - \phi) \times R$. The SCT has coverage at $|\eta| < 2.5$ with the crack located at $|\eta| \approx 1.3$.

Transition Radiation Tracker

The last section of the ID is the Transition Radiation Tracker (TRT). The TRT is a gaseous detector composed of roughly 300.000 straws (in the barrel and end-caps). It measures the track trajectory in the $R - \phi$ plane. The straws have a radius of 2mm, with a centered golden-plated tungsten wire that measures 0.03 mm in diameter. In-between the wire and the straw a mixture of $XE/Ar/CO_2/O_2$ gas flows. A charged particle ionizes the gas and creates a current that is measured by the read-out channel. The detector barrel consists of 73 straw layers with the crack at $0.8 < |\eta| < 1.0$, whereas the end-caps has full coverage at $|\eta| < 2$.

As mentioned before, semiconductors can be more beneficial than gas, but whereas the Pixel detector and SCT estimate a district number of positions for the charged particles with very high precision, the TRT must continuously track the them with less precision allowed. The TRT is large compared to the Pixel Detector and SCT allowing the radius of the track to be measured. The TRT is also able to do PID (will be explained below) on electrons. They radiate transition photons in KeV range that react XE gas, creating distinct signals in the TRT.

1.2.3 *Calorimeter physics*

After passing the three sections of the ID, the particle will encounter the calorimeters. They consist of two parts, the electromagnetic calorimeter (ECAL), and the hadronic calorimeters (HCAL)[16]. The calorimeters are significantly larger than the ID (see figure 1.4) but have the same coverage in $|\eta| < 2.5$. In contrast to the ID, the focus of the calorimeters is measuring the energy of any hadrons, electrons, and photons entering the detector. They are constructed with as much material as possible to increase the interaction rate of the particles. Due to the hadrons sizes, they are difficult to *stop* compared to the electrons and photons, hence two separate calorimeters are required, one for measuring the energies of electrons and photons (the ECAL) and one for energies of hadrons (the HCAL). The calorimeters are composed of passive and active layers. An outgoing particle interacts with the passive layers and creates a shower of particles, which deposits its energy in the active layers, making it possible to measure the energy of individual particles of the shower. In the analysis, only measurements from the ID and ECAL will be used.

[16] An additional part is the forward calorimeter (FCAL), which is covering $2.47 < |\eta| < 4.9$. However this is beyond the scope of this thesis.

The Electromagnetic Calorimeter

Electrons and photons have unique properties compared to hadrons and muons. An important feature of the electron is *Bremsstrahlung*. When bent or decelerated, the electrons emit photons that convert into electron-positron pairs (e^+e^-), leading to a domino effect creating electromagnetic showers [20]. The incoming electrons and photons with energy E_0 cascade into more electrons, positrons, and photons, each containing only a fractions of E_0. The energy of the shower is collected by the ECAL cells and algorithms are used to reconstruct the original energy, E_0, of the electron or photon. The objective of this thesis is to improve this reconstruction of E_0.
The ECAL is constructed of two half barrels centered on the z-axis, both covering $0 < |\eta| < 1.475$, with the end-caps closing the cylinder and covering at $1.375 < |\eta| < 3.2$.
Both the barrel and end-caps consists of four layers. Three of the barrel layers can seen in figure 1.6 and all four in figure 1.7. As seen in figure 1.6, the internal layout of the layers is accordion-shaped, giving full coverage in the ϕ direction. The ECAL is a lead-liquid-argon (Lead-LAr) detector, lead being the passive material and Argon being the active one. A particle shower ionizes the liquid-argon, creating more e^+e^- pairs. The e^- travels to the copper electrodes submerged in the liquid, creating yet another current. From this current it is possible to estimate the position and amount of energy deposited by the shower particles.

Cells of the ECAL

As hinted in figure 1.6 and figure 1.7, the ECAL layers are composed of cells with varying granularity. The complete set of (η, ϕ) resolu-

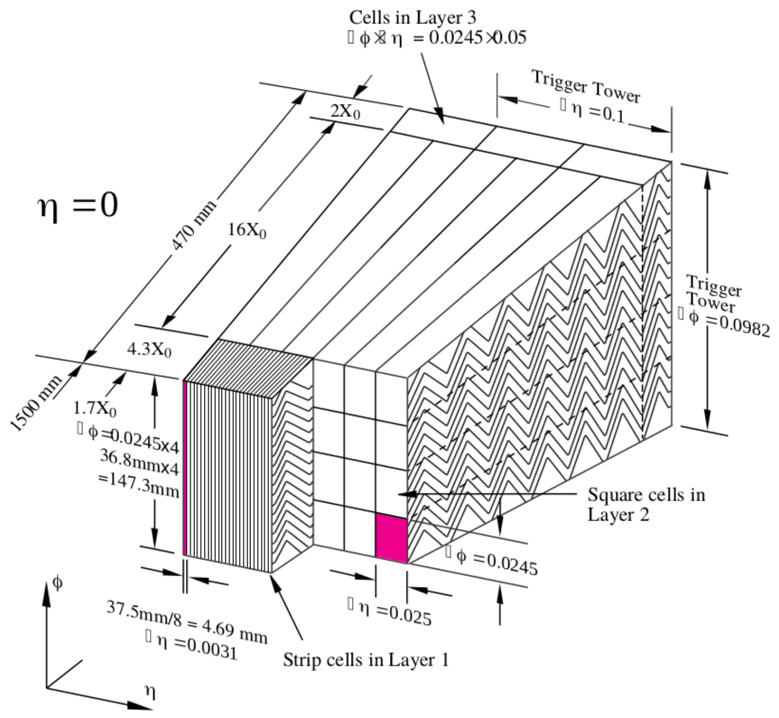


Figure 1.6: From [13]. The figure shows the accordion of the ECAL, namely layers 1,2 and 3. Layer 0 (pre-sampler) is not displayed on the figure and is before layer 1. The illustrates the $\eta \times \phi$ granularity of cells within each layer as well as the accordion shape of the ECAL.

tions in the layers can be seen in table 1.1. Each cell[17] measures two properties: the energy deposited in the cell and the *ATLAS time* of the measurement[18].

However, with bunch crossing every 25 ns and read-out time being finite, there might be small overlaps between the event energy in the pixels. This is corrected for in figure 1.8, with the energy of a pulse measured starting at $t_{start} = 450$ ns and ending at $t_{end} = 600$ ns. Afterwards, the additional energy from the next crossing is corrected for by underestimating the energy after $t = 600$ ns. This behavior should help correcting for pileup.

[17] The cells of the ECAL will also be referred to as pixels, as they make up the images used in the analysis.

[18] Additional information, such as noise and gain are also measured. However, these will not be used in the analysis.

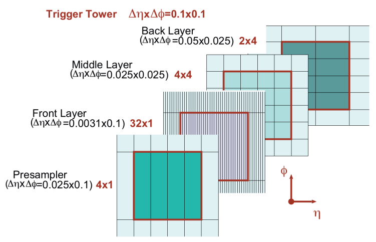


Figure 1.7: From [42]. The figure shows the (η, ϕ) resolution of the ECAL layers in the barrel.

The Hadronic Calorimeter

The ECAL is not designed to stop all particles. Hadrons, such as charged pions (π^+/π^-), are much more difficult to stop and only deposit a small fraction of their energy in the ECAL. Other hadrons, such as the neutron, with neural charge, will often not make any interactions in the ID or ECAL. Therefore, the ECAL is encapsulated by the Hadronic calorimeter (HCAL), which is designed to measure hadrons. The HCAL is significantly deeper than the ECAL and consists of three parts, the barrel, the extended barrel and an end-caps on each end. Both the barrel and the extended barrel can be seen in figure 1.4. The barrels are tile calorimeters using steel as an absorber and a scintillator[19] as its active material. When a hadron passes through the HCAL, it interacts with the steel, leading to a hadronic shower of charged hadrons. These charged hadrons causes the scintillator to radiate light, from which the energy and position of the hadrons can be determined.

[19] A scintillator radiates light, when exposed to a charge particle.

Layer	Granulartity in $\eta \times \phi$	$\|\eta\|$ coverage
	ECAL barrel	
Layer 0	0.025×0.1	$\|\eta\| \leq 1.52$
Layer 1	$0.025/8 \times 0.1$	$\|\eta\| \leq 1.40$
	0.025×0.025	$1.40 < \|\eta\| \leq 1.475$
Layer 2	0.025×0.025	$\|\eta\| \leq 1.40$
	0.075×0.025	$1.40 < \|\eta\| \leq 1.475$
Layer 3	0.050×0.025	$\|\eta\| \leq 1.35$
	ECAL end-caps	
Layer 0	0.025×0.1	$1.5 < \|\eta\| \leq 1.8$
Layer 1	0.05×0.1	$1.375\|\eta\| \leq 1.425$
	0.025×0.1	$1.425 < \|\eta\| \leq 1.5$
	$0.025/8 \times 0.1$	$1.5 < \|\eta\| \leq 1.8$
	$0.025/6 \times 0.1$	$1.8 < \|\eta\| \leq 2.0$
	$0.025/4 \times 0.1$	$2.0 < \|\eta\| \leq 2.4$
	0.25×0.1	$2.4 < \|\eta\| \leq 2.5$
Layer 2	0.05×0.025	$1.375\|\eta\| \leq 1.425$
	0.025×0.025	$1.425 < \|\eta\| \leq 2.5$
Layer 3	0.05×0.025	$1.5 < \|\eta\| \leq 2.5$
	HCAL LAr end-caps	
Layer 0,1,2,3	0.1×0.1	$1.5 < \|\eta\| \leq 2.5$
	HCAL tile gap	
Layer 1	0.1×0.1	$0.9 < \|\eta\| \leq 1.0$
Layer 2	0.1×0.1	$0.8 < \|\eta\| \leq 0.9$
Layer 3	0.1×0.1	$1.0 < \|\eta\| \leq 1.2$
	0.2×0.1	$1.2 < \|\eta\| \leq 1.6$
	HCAL tile barrel	
Layer 1	0.1×0.1	$\|\eta\| < 1.0$
Layer 2	0.1×0.1	$\|\eta\| < 0.9$
Layer 3	0.2×0.1	$\|\eta\| < 0.7$
	HCAL tile extended barrel	
Layer 1	0.1×0.1	$1.1 < \|\eta\| < 1.6$
Layer 2	0.1×0.1	$1.0 < \|\eta\| < 1.5$
Layer 3	0.2×0.1	$0.9 < \|\eta\| < 1.3$

Table 1.1: Granularities of the layers within the central ($|\eta| < 2.5$) calorimeters. It is important to note the different granularities, as this poses a central problem for the regression algorithm used in chapter 5 and 6. However, it will be solved in chapter 4.

The barrel covers $|\eta| < 1$, where the extended barrel covers $0.8 < |\eta| < 1.7$. The end-caps are different, as they use copper as the passive material and LAr as active. The end-caps have four layers with same resolution 0.1×0.1 in $\Delta\eta \times \Delta\phi$ for $1.5 < |\eta| < 2.5$ and 0.2×0.1 in $\Delta\eta \times \Delta\phi$ for $2.5 < |\eta| < 3.2$. The complete resolution of the layers in ECAL and HCAL can be seen in table 1.1.

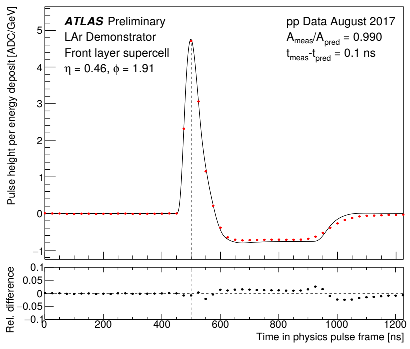


Figure 1.8: Signal read-out of cells [2]. The figure read-out behavior of a cell and the additional correct because of the fast bunch crossings.

Muon System

The last line of defense for detecting particles from the collision, is the muon chamber (MS). Only neutrinos and muons will reach the MS[20]. Both particles are very difficult to detect because of their low interaction rate. The MS is similar to the TRT in the ID, using a hollow rod filled with gas and a centered rod with voltage. A muon interacts with a gas, creating an electron that drifts to the centered rod. The drift can then be measured and used to estimate the path of the muon. The MS covers a region of $|\eta| < 2.7$. A toroid magnet at 3.5 Tesla is used to bend the trajectories of muons in ϕ in order to measure their momentum.

[20] However, neutrinos cannot be measured in the MS.

1.3 Reconstruction

After the collision data has been recorded, the real detective work begins. The reconstruction of particle tracks, the identification of particles (PID), energy reconstruction (ER), and matching the ECAL clusters to tracks have to be performed using reconstruction algorithms. These algorithms will be explained below.

1.3.1 Track reconstruction

Only charged particles with $p_T > 0.5$ and $|\eta| < 2.5$ are selected in the reconstruction. The reconstruction of tracks is set in three stages. An overview of the reconstruction will follow. For further information see section 10.2.1 in [13].

1. The pre-processing stage, where the raw data of the collision from the Pixel Detector and the SCT are converted into clusters. Afterwards the SCT clusters are transformed into space points.

2. The track finding stage, where track seeds are generated from the three space-points of the Pixel Detector with the hits from the first layer of the SCT. These seeds are extended out and used to find track candidates throughout the SCT. The candidates are then fitted and outlying clusters are removed. These fits are then extended into the TRT to account for the curvature[21] of the track. From this the p_T can be measured. For high p_T tracks the precision of the p_T is low, as the radius of the bend is very large and thus difficult to measure. At last, the tracks are re-fitted using the whole ID and compared to the silicon-only track candidates. Bad fits are again labeled as outliers.

[21] Because the magnetic field will bend the track of the charged particles.

3. The last part of the reconstruction is to trace back the track to their primary vertex close to the interaction point.

1.3.2 *Identification of particles*

The next important step is to identify the particle type and decay channel. In MC, the *TruthType*, *TruthOrigin* and *TruthPdgId* are generated, so the decay and particle type are known. However, in Data, this is not the case. To identify electrons in Data, ATLAS uses a likelihood algorithm to classify if its an electron or not. For photons, the identification is a cut-based method, making rectangular cuts in photons variables.
ATLAS divides their identifications into three cuts, *Tight*, *Medium* and *Loose*, with varying efficiencies. In this thesis, multiple criteria are used for selecting the correct events in Data. These are explained in chapter 3.

1.3.3 *Cluster matching*

We will only be focusing on cluster matching for ECAL. The clusters of energy in the ECAL have to be linked to the particles from the collision. Usually, particles deposit their energy in multiple cells. This is a two step process [12]. First, the size of the cluster is determined. Afterwards, a electron or photon is linked to the cluster.
To determine the number and size of the cluster, the algorithm starts by seeding initial clusters containing high energy cells in the ECAL. To become an electron cluster seed, it is required for the cluster to have $E_T \geq 1$ GeV and to match at least four hits in the SCT. For photon clusters the only requirement is $E_T \geq 1.5$ GeV. After selecting the cluster seeds, the algorithm will find satellite clusters, which is neighboring cells (within $\Delta\eta \times \Delta\phi = 0.075 \times 0.125$ of the barycentre) where the energy is (some σ) above the noise level[22] are attached to the seed, forming a SuperCluster. These satellites may be photons that radiated of the original electron, and therefore must be a part of the cluster. Afterwards, tracks from electrons and conversion vertices from photons are matched with the SuperClusters. However, this matching is not completely accurate and is vulnerable to mismatching. This will be explored in section 3.4.

[22] Additional requirements can be seen in [12].

1.3.4 *Energy reconstruction*

An important property of the reconstructed electrons and photons is their initial energy, E_0. It is not included in the track reconstruction described above. A good starting guess for E_0 is the deposited energy in the ECAL accordion, E_{acc}. However, as seen in figure 7.8, this is not the full story. The showers might not be isolated or some amount of energy might be deposited before the ECAL, so an energy correction is needed. In the following section, the calibration used by ATLAS, denoted $E\text{calib}^{(BDT)}$, will be explained[23]. Understanding

[23] For further information, see [11].

$E\text{calib}^{(BDT)}$ is important, as it is the main competition to the *DeepCalo* model.

Energy calibration at ATLAS

Energy calibration at ATLAS is separated into five selections meant to correct for the various biases that might appear during reconstruction.

- First is estimating the energy, E_0, of the particle. Here, a multivariate regression algorithm is used to approximate the energy using shower properties and the sum of energies deposited in the calorimeter. The multivariate regression algorithm used is a Boosted Decision Tree (BDT) with variables seen in table 1.2. It is trained on Monte Carlo (MC) simulated samples and is therefore dependent on the accuracy of the simulation.
- An adjustment is modeled to the relative energy scales of the different ECAL layers based on muon and electron studies. To extrapolate to the full energy range, an adjustment is applied as a correction before estimating the energy of Data.
- Correcting for non-uniformities in ECAL that might affect Data.
- Adjusting the overall energy scale of Data using $Z \to ee$ decay. The difference between MC and Data is also derived and corrected. This should be universal for photons and electrons.
- Lastly, an independent comparison between MC and Data in $J/\psi \to ee$ with low-energy electrons in focus. For photons, the Z boson decay is used.

DeepCalo is only relevant to step 1 and is proposed as an alternative to the BDT. All corrections mentioned above must to be revisited if *DeepCalo* were to replace $E\text{calib}^{(BDT)}$.

1.4 Motivation

The following section will motivate the analysis of this thesis. Note that the performance measurements used below will be explained in chapter 3. For now, σ_{CB} is the width of CB and this should be minimized.

Where to improve the reconstruction?

To reconstruct the invariant mass of the parent particle (see equation 1.1), the energies, ηs, and ϕs are needed from the byproduct of the decay.

Type	Name	Description
Energy	E_0	Energy deposited in the pre-sampler.
	E_{acc}	Energy deposited in the accordion $E_1 + E_2 + E_3$.
	E_1 / E_2	Ratio of energy between layer 1 and 2
Geometric	$\eta_{cluster}$	η impact point in the calorimeter
	ϕ_{mod}	$\lvert\phi_{cluster} - \phi_{cell}\rvert$, where ϕ_{cell} is the centre of the closest cell in layer 2 of the ECAL
	η_{mod}	$\lvert\eta_{cluster} - \eta_{cell}\rvert$, where η_{cell} is the centre of the closest cell in layer 2 of the ECAL
	ϕ_{TG3}	ϕ position in the scintillators in the tile gap.
Photon	r^{conv}	Estimated radius of the photon conversion in the transverse plane
	$R^{conv}_{E/p}$	E^{acc}_T / p^{conv}_T.
	$F^{conv}_{p_T}$	The amount of p^{conv}_T carried by the largest p_t track.

Table 1.2: List of variables used in the $Ecalib^{(BDT)}$ for energy reconstruction. List is from [11].

E_{truth}	$(\eta, \phi)_{truth}$	σ_{CB}	χ^2
X	X	0.06 ± 0.09	1.29
X		0.08 ± 0.09	1.27
	X	2.15 ± 0.02	2.21
		2.16 ± 0.02	2.22

Figure 1.9: MC sample for the decay channel, $Z \to ee$. σ_{CB} will be explained in chapter 4, but for now it is just a measure for how well the invariant mass is reconstructed, with $\sigma_{CB} = 0$ being best.

In table 1.9, the width, σ_{CB}, of the BW⊗CB on $Z \to ee$ in MC can be seen. As seen from the table, the energy resolution is the most important property to improve, compared to η and ϕ as $\sigma \approx 0$ when using only E_{truth} and $\sigma_{CB} \approx 2$ when using only $(\eta, \phi)_{truth}$. This can also be seen in equation 1.1. From the Higgs discovery in 2012 (figure 1.10), we see that the number of background photons is high and the resonance peak is wide. Improving The PID to remove background photons is not possible, so the only way to make the peak more distinct is to improve the energy reconstruction of the photons. This is important in $H \to \gamma\gamma$, but also in other Higgs channels, which are not as frequent, eg. $H \to Z\gamma$[1]. The ATLAS and CMS detectors designs were optimized with $H \to \gamma\gamma$ in mind. At the moment CMS has the most accurate energy reconstruction but is also the most expensive at a material price of 512 million euros[6], whereas material price of ATLAS is 390 million euros[30]. *DeepCalo* might close the performance gap between the two. *DeepCalo* will have its performance measured in the *control channels*, but the true motivation for improving the ER is because of the *motivation channels* that will benefit from a more accurate ER.

The channels can be seen in table 1.3. In figure 1.11, the energy ranges of particles from the channels in table 1.3 can be seen. Overlapping energy ranges between the *control channel* and *motivation channel* are very important, as the algorithms used to reconstruct the particles do not generalize well outside of its energy range.

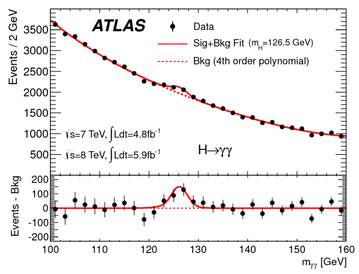


Figure 1.10: The figure shows the discovery of the Higgs boson from [9].

Table 1.3: The table shows the control and motivation channels. Control channels are where we will test improvement of the ER, and the motivation channels are channels that would benefit from an improvement.

Motivation channels	
$H \to H(\to \gamma\gamma)H(\to b\bar{b})$	Benefit from γ improvement
$H \to \gamma\gamma$	Benefit from γ improvement
$H \to Z(\to ee)\gamma$	Benefit from γ/e improvement
$H \to \gamma^*(\to ee)\gamma$	Benefit from γ/e improvement
Control channels	
$Z \to ee$	Test e improvement
$Z \to \mu\mu\gamma$	Test γ improvement

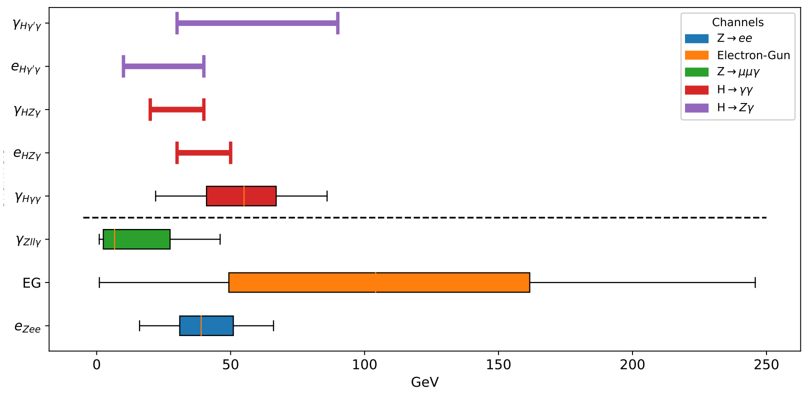


Figure 1.11: Energy range by-product of the decay of interest to this thesis. The black dashed line is the borderline between the motivation channels and the control channels. The box plots indicate energy ranges, where we have available Data or MC. See table 3.1 to find the file containers used in this thesis.

2 Machine Learning

This chapter will elaborate on the concept of Machine Learning (ML), its applications and limitations. If more information is desired on this subject, see [19] and [39]. As the previous chapter 1 laid the fundamental groundwork for the ATLAS experiment, the following chapter will introduce the reader to the concept of ML and why it will be advantageous compared to previously used methods. It will be assumed that the reader has a basic knowledge of linear algebra.

There are many different branches of ML. The ML algorithms applied in this thesis will be supervised, meaning the algorithms will be exposed to the data and the labels associated with the data, in contrast with unsupervised ML, where only the data is exposed but the labels might be unknown or kept secret. In the following section 2.1, the theoretical approach to ML will be explained. It will be followed by a section introducing the ML algorithm called Neural Networks (NN).

Note that throughout the thesis, the term optimization will be used in relation to the performance of an architecture, and the term minimize will refer to the minimizing the difference between the regression algorithm and the truth using a *Loss functions* (explained below).

Symbol	Definition
$\mathcal{X}$	Feature space ($D \geq 1$)
$\mathcal{Y}$	Label space ($D \geq 1$)
X	Single feature $X \in \mathcal{X}$
Y	Single label $Y \in \mathcal{Y}$
S	Sample space ($(X, Y) \in (\mathcal{X} \times \mathcal{Y})$)
$\mathcal{H}$	Hypothesis space ($D \geq 1$)
$l(Y', Y)$	Loss function between predictions
$p(X, Y)$	The joint probability distribution of S

Table 2.1: The table shows the symbols and definitions of properties in ML. D is the dimension.

2.1 Supervised Machine Learning

2.1.1 Terminology

Supervised ML is one of the most widespread machine learning techniques.

The aim of supervised machine learning is to *train*[1] an algorithm to map $\mathcal{X}$ to $\mathcal{Y}$. $\mathcal{Y}$ could be a cat or a dog, and $\mathcal{X}$ could be its height, weight, and color. Then, an algorithm will be trained in distinguishing between a cat and a dog by their height, weight, and color. The complete sample space is denoted $S = (X_1, Y_1), ..., (X_n, Y_n)$, where the S is assumed to be i.i.d[2]. When working in a complex S, multiple mappings are possible, each one called a hypothesis, given by h, and with $\mathcal{H}$ being the complete set of hypotheses. In ML, *loss* is a crucial concept to understand. It is a measure of how well a selected hypothesis maps $h(\mathcal{X}) \rightarrow \mathcal{Y}$ and its purpose is to force the algorithm to find the $h(\mathcal{X})$ that maps most accurately to $\mathcal{Y}$ and minimizes the expected

[1] The concept of *training* will be explained below.

[2] Independent and identically distributed random variables, meaning all pairs stem from the same joint probability distribution $p(X, Y)$.

loss $L(h)$. Loss is separated into two types, as seen in equation 2.1.

$$\begin{aligned}\text{Expected loss: } L(h) &= \mathbb{E}(l(h(X), Y)) \\ \text{Empirical loss: } \hat{L}(h, S) &= \frac{1}{n}\sum_{i=1}^{n} l(h(X_i), Y_i)\end{aligned} \tag{2.1}$$

Where $h(X) = Y'$ is the predicted label space. The star of the show is the expected loss $L(h)$, which, as the name suggests, is the expected error independent of S. However, the complete joint probability distribution $p(\mathcal{X}, \mathcal{Y})$ is unknown, so computing the expected loss is not possible. The solution is to draw i.i.d samples from S that approximate $p(\mathcal{X}, \mathcal{Y})$ and measure the empirical loss $\hat{L}(h, S)$ of a given h, which can measure how well the algorithm reconstructs $p(\mathcal{X}, \mathcal{Y})$.

2.1.2 *Over- and under-fitting*

Selecting the $h \in \mathcal{H}$ that minimizes the empirical loss in equation 2.1 (written as $\hat{h}^*_S = \text{argmin}_h \hat{L}(h, S)$), might not be the optimal model. Rather, S is a finite sample of the joint probability distribution and might contain impurities[3] from measured data, resulting in $\hat{h}^*_S$ being tailored or over-fitted to S and $\mathbb{E}\left[\hat{L}(\hat{h}^*_S, S)\right] \neq \mathbb{E}\left[L(\hat{h}^*_S)\right]$. This can partially be fixed by splitting S into S_{train}, S_{val} and S_{test}, where the algorithm is trained on the S_{train} but the hypothesis chosen is the one satisfying $\hat{h}^*_{S_{train}} = \text{argmin}_h \hat{L}(h, S_{val})$. This makes $\hat{h}^*_{S_{train}}$ less over-fitted towards S_{train}, as the unseen sample S_{val} is used to select h. This will in turn result in better generalization. Lastly, when an optimal $\hat{h}^*_{S_{train}}$ is found, the empirical loss can be measured using S_{test}. This might give the reader a good idea of why the terms *Big Data* and *Large-Scale data analysis* are present when talking about ML, as a large amount of data is needed to meaningfully construct $p(\mathcal{X}, \mathcal{Y})$, so $\hat{L}(h, S) \approx L(h)$. This can also be seen in the generalization bound found below (equation 2.3).

[3] Impurities may be statistical fluctuations or wrongly measured data.

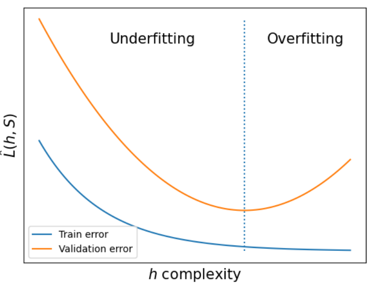


Figure 2.1: The figure shows the different scenarios and shortfalls that can appear when training a machine learning model, and why a validation sample is used to select the correct h. If training is terminated in the under-fit region, there is still generalization aspects to be found, which can be seen from the loss on both S_{train} and S_{val}. In the over-fit region, the loss from the S_{train} will decrease, however, the loss from the S_{val} will increase as the model over-fits to statistical fluctuations in S_{train}. The sweet spot is on the dotted line, where both have their minimum.

When a hypothesis h is selected, $\hat{L}(h, S_{test})$ can be measured, but S_{test} will only approximate $p(\mathcal{X}, \mathcal{Y})$, and a measure of the relation between the empirical loss and the excepted loss is needed. This relation is called the generalization bound $G(h)$, given by

$$G(h) = L(h) - \hat{L}(h, S), \tag{2.2}$$

where $G(h)$ must be minimized. As stated above, $\hat{L}(h, S)$ is tailored using S, so $L(h) \neq \hat{L}(h, S)$ as $\hat{L}(h, S)$ is biased towards S_{test}. The expected loss can be bounded using Hoeffding's inequality, see [39]. Using Hoeffding's inequality, a generalization bound of a finite number of hypotheses can be shown to be

$$L(h) \leq \hat{L}(h, S) + \sqrt{\frac{ln\left(\frac{M}{\delta}\right)}{2n}}, \tag{2.3}$$

where $M = |\mathcal{H}|$ is the size of the hypothesis, n is the size of S and δ is the confidence of the model. The objective is then to minimize the inequality, so the expected loss is strictly bounded. In ML, intuitively, the larger S, the more accurate the model. This can also be seen in equation 2.3 where the bound on the expected loss falls by $\sqrt{1/n}$. M is a complexity measurement of the model. With a highly complex model, the right-hand side will increase, giving a looser bound on the expected loss. Intuitively, a complex model might be tailored completely to S, resulting in *over-fitting* h_S^* to S. Over-fitting means that the model does not generalize to i.i.d S but rather fits a subset of S, making performance worse for unseen S. The model can also under-fit if there is still generalization to be found within S. Both phenomena can be seen in figure 2.1. The selected h should be at $\hat{h}^*_{S_{train}} = \text{argmin}_h \hat{L}(h, S_{val})$, which is the dotted line in figure 2.1.

2.1.3 *Classification and Regression*

The two main branches of supervised ML are classification and regression. Classification is where $h(X) \rightarrow Y'$ is a set of classes often defined by a discrete set of values, eg. this could be a binary classification $Y = \{0, 1\}$ between a cat or a dog. In regression $h(X) \rightarrow Y'$ predicts a continuous spectrum of values[4], eg. this could be predicting the weight of a cat or a dog.

[4] Classification algorithms can also predict a continuous spectrum of values between 0 and 1, which should be seen as probabilities of a given label. Probabilities or uncertainties are more difficult to obtain for regression algorithms. In chapter 4, we will propose a loss function that will help us determine the uncertainty in regression.

To give an intuitive understanding of classification and regression, we will continue with the cat and dog analogy. Regression could be predicting the weight of animals based on its height, age and type of animal (cat/dog). Here the number types of the feature space and label space are $\mathcal{X} = \mathbb{R} \times \mathbb{N} \times \{0, 1\}$ and $\mathcal{Y} = \mathbb{R}$, respectively. If the algorithm were to do binary classification between two species of animals (cat/dog) based on their weight, height and age, the feature space and label space will be $\mathcal{X} = \mathbb{R} \times \mathbb{R} \times \mathbb{N}$ and $\mathcal{Y} = \{0, 1\}$, respectively. The definition of number types are given in table 2.2.

Name	Symbol	Definition
Real numbers	$\mathbb{R}$	All number.
Rational number	$\mathbb{Q}$	Fractional.
Irrational number	$\mathbb{R}$-$\mathbb{Q}$	non-fractional.
Intergers	$\mathbb{Z}$	$\pm$ integers.
Natural numbers	$\mathbb{N}$	Positive integers.

Table 2.2: The table shows the main types of number, except for complex numbers $\mathbb{C}$, as these are not relevant in ML.

This thesis will mostly be focused on regression and its applications, but it will briefly look into classification as well.

2.1.4 *Loss function*

As of yet, the concept of *learning* has not been explained. It can vary between ML algorithms, yet a common denominator is the loss function. In section 2.1.2, the empirical and expected loss were introduced. The empirical loss is the loss measured by the ML algorithm and will simply be referred to as the loss unless stated otherwise. The loss function is used to measure the mistakes of an algorithm and can, in most cases, be viewed as the error of the predictions. Two common loss functions in regression are the square loss,

$$L(Y', Y) = (Y' - Y)^2, \quad (Y', Y) \in \mathbb{R}, \tag{2.4}$$

and the absolute loss,

$$L(Y', Y) = |Y' - Y|, \quad (Y', Y) \in \mathbb{R}. \tag{2.5}$$

When a ML algorithm is *learning*, it measures the loss $L(h(X), Y)$ of its current state in training and changes its trainable parameters accordingly to minimize the loss. The specific *learning* behavior of the ML algorithm used in this thesis will be introduced in the following section. Note that *learning* can take many forms dependent on the ML algorithms, but only *learning* for NN will be explained in the thesis. There are many different loss functions, all with different properties and behaviors. Looking at the two examples above, equation 2.4 weighs outliers heavier than equation[5] 2.5 because of its squared term. Thus, when summing the loss, the outliers will be weighted significantly heavier because of the squared term. This also means that the loss between two algorithms cannot be compared if the loss functions are different. An additional discussion of the loss functions and their behaviors can be seen in section 4.2.

[5] If $Y' - Y > 1$.

2.2 Neural Networks

In the following section, an elaboration on the basics of Neural Networks will take place, as it is the algorithm used in the analysis. The focus will be on a type of Feedforward[6] Network called a convolutional neural network (CNN). A CNN is able to handle structured[7] n-dimensional input space. Where the previous section was mostly focused on the theoretical elements of ML, the following section will introduce applied ML with a focus on networks[8].

[6] *Feedforward* in a network means the information will only flows forward through the network, resulting in the output Y'. No information is looped back into the network as is the case with Recurrent Neural Networks.

[7] Structured in the sense that the dimensions of the input must not be changed.

[8] In general "networks" refer to the many NN types, eg. recurrent neural networks (RNN), convolutional neural networks (CNN), graph neural networks (GNN) etc.

2.2.1 Deep Feedforward Networks

Neural Networks (NN) are inspired by neuroscience to behave similarly to neurons in a brain. The ambition of a NN is to approximate some function[9] f^* by mapping $\mathcal{X}$ to some function $f(\mathcal{X}; \boldsymbol{\theta})$, where $\boldsymbol{\theta}$ are the trainable parameters in the NN. To get an idea of the layout of a NN, a sketch overview of a NN can be seen in figure 2.2 with an input layer of 3 neurons, two hidden layers with 4 and 3 neurons, respectively, and an output layer consisting of 1 neuron. It is within these neurons the magic happens (it will be elaborated on in section 2.2.2).

[9] This can be thought of as the hypothesis h^* from before.

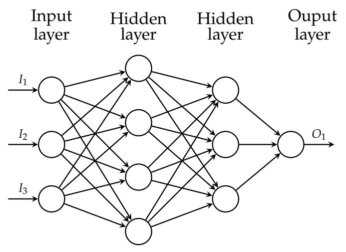


Figure 2.2: A sketch of a Neural Network. Each circle indicates a neuron with the arrow being where data is fed from.

Each layer in a NN can be viewed as its own function $f^{(l)}$. With the use of the chain rule, the whole network can be seen as being composited of the function from each layer,

$$f(\mathcal{X}) = \left(f^{(L)} \circ f^{(L-1)} \circ ... \circ f^{(2)} \circ f^{(1)} \right) (\mathcal{X}), \tag{2.6}$$

where $\mathcal{X}$ is the input vector of the neurons, $f^{(1)}$ is the first hidden layer and $f^{(L)}$ the output layer. This is the concept behind a NN. The way the layers are dependent on each other will be explained in further detail in section 2.2.3.

2.2.2 The Structure of Neurons

The neurons, within a NN, manipulates the input x_i by linear transformations. The parameters of the linear transformations are trained,

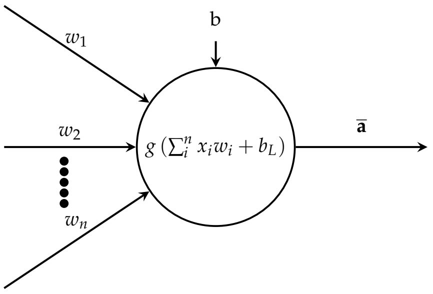


Figure 2.3: The sketch shows a single neuron with n connections, where $\bar{\mathbf{a}}$ is the vector output and g is a given activation function. For easy readability, the linear transformation is set up as a sum. However, using equation 2.7 will decrease computation time and is used in real-world algorithms.

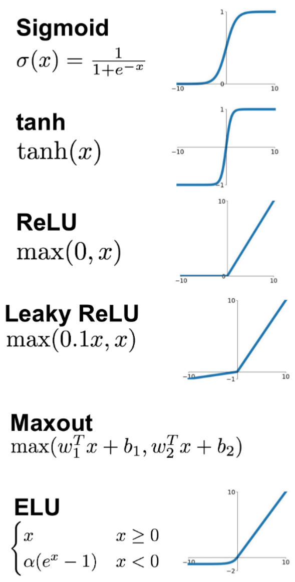


Figure 2.4: Possible activation functions. Figure from [22].

in collaboration with all neurons in the network, to map $f(\mathcal{X};\theta) \rightarrow \mathcal{Y}$. Each layer contains ≥ 1 neurons, with each neuron having a vector of weight, $\mathbf{W}$ for each connection to the neuron, and it has a single bias b unique for every neuron. The linear transformation within neurons can be written as

$$\mathbf{z} = \mathbf{W}^\top \mathbf{x} + \mathbf{b} \tag{2.7}$$

where $\mathbf{W}$ is the vector of the weights, $\mathbf{x}$ the input vector and $\mathbf{b}$ the bias ($\mathbf{b}$ is extended to a vector with the same number for each entry). This results in linear behavior for $\mathbf{z}$. A sketch of a single neuron can be seen in figure 2.3.

Activation function

Linear transformation of neurons is not the complete picture. In figure 2.3, a function, g, is wrapped around the linear transformation. This is the activation function, which adds non-linearity to the network by mapping the output of the linear transformation to an activation function, as seen in equation 2.8,

$$\mathbf{a} = g(\mathbf{z}), \tag{2.8}$$

where $\mathbf{a}$ is the final output of the neuron. Some examples of activation functions can be seen in figure 2.4. The two most commonly used activation functions are the *ReLU*, which is used within the network and in the output layer for regression networks, and *sigmoid*, which is used as an output function for classification networks. Expressions for both can be seen in equation 2.9

$$g_{ReLu}(x) = max(0, x), \quad g_{Sigmoid}(x) = \frac{e^x}{e^x + 1} \tag{2.9}$$

Choosing the right activation function is very important for the performance of the network. Selecting the wrong one can result in a vanishing gradient when back-propagating (explained below), meaning the network will not be *learning* anything from the data.

2.2.3 *Learning properties*

Back-propagation

As seen above, each neuron has a vector of $\mathbf{W}$ and $\mathbf{b}$ associated with it. These are trainable parameters and change according to the loss function (or cost, $C_\#$, for the algorithm). The amount of change to each trainable parameter is calculated by the back-propagation algorithm. In forward feeding neural networks, the inut$\mathcal{X}$ flows forward in the network, producing an output Y'. Y' is evaluated against Y in the loss function. A back-propagation algorithm is a general algorithm able to calculate the gradient of a function f, $(\nabla_x f(\mathbf{x}, \mathbf{y}))$, and, as shown in equation 2.6, NN is a collection of functions. After the cost is measured, the information is sent back through the network, computing the gradient of the loss function, and an *optimizer* algorithm is thereafter responsible for changing the trainable parameters. To compute the gradient backwards through the network, back-propagation takes advantage of the chain rule in calculus,

$$f(g(x))' = f'(g(x))g'(x) \text{ can be view as } \frac{\partial f}{\partial x} = \frac{\partial f}{\partial g}\frac{\partial g}{\partial x}. \tag{2.10}$$

Back-propagation will give a measure of how to change the weights and biases of the network to minimize the loss function. In figure 2.5, a sketch of the output layer can be seen. Here, the activation output from the previous layer is combined with the weight and bias of the output neuron (see equation 2.7). Afterwards, the output is mapped to an activation function, resulting in $a^{(L)}$. Finally, the loss is calculated by comparing $a^{(L)}$ and Y. Now, back-propagation must calculate $\partial C_{\#}/\partial \omega^{(L)}$. This is computed using the chain rule,

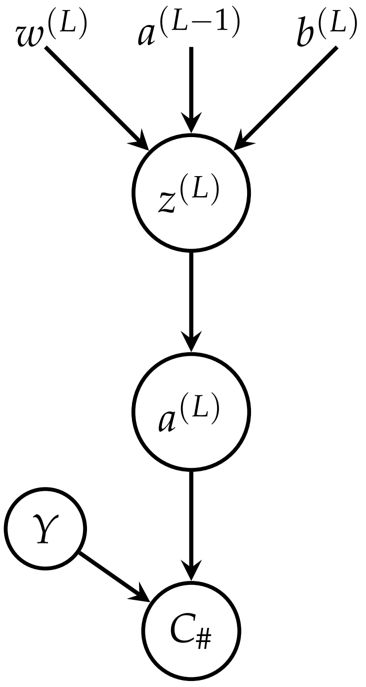


Figure 2.5: The sketch shows an example of an output layer of a NN. $C_{\#}$ is the cost of the #th batch ($\# \in \mathbb{N}$).

$$\frac{\partial C_{\#}}{\partial w^{(L)}} = \frac{\partial C_{\#}}{\partial a^{(L)}}\frac{\partial a^{(L)}}{\partial z^{(L)}}\frac{\partial z^{(L)}}{\partial w^{(L)}} \tag{2.11}$$

If MSE is applied as cost, the back-propagation of the simple output layer in figure 2.5 is

$$\frac{\partial C_{\#}}{\partial a^{(L)}} = 2(a^{(L)} - Y), \quad \frac{\partial a^{(L)}}{\partial z^{(L)}} = g'(z^{(L)}) \text{ and } \frac{\partial z^{(L)}}{\partial w^{(L)}} = a^{(L-1)}$$
$$\text{will result in } \frac{\partial C_{\#}}{\partial w} = 2(a^{(L)} - Y)g'(z^{(L)})a^{(L-1)}. \tag{2.12}$$

For calculating the gradient of the bias parameter in the network, $\frac{\partial z^{(L)}}{\partial b^{(L)}} = 1$ is used instead. Generalizing each layer of calculation to f^L, the back-propagation through the network can be defined as

$$\frac{\partial C_{\#}}{\partial w^{(L)}} = \frac{\partial C_{\#}}{\partial f^{(L)}}\frac{\partial f^{(L)}}{\partial f^{(L-1)}}\cdots\frac{\partial f^{(1)}}{\partial w^{(L)}}. \tag{2.13}$$

Batch averaging is used to compute the gradient over the entire batch (batch will be explained below). The complete vector gradient of a network is given by

$$\nabla C_{\#} = \begin{bmatrix} \frac{\partial C}{\partial w^{(1)}} \\ \frac{\partial C}{\partial b^{(1)}} \\ \vdots \\ \frac{\partial C}{\partial w^{(L)}} \\ \frac{\partial C}{\partial b^{(L)}} \end{bmatrix}. \tag{2.14}$$

When $\nabla C_{\#}$ is found, the second stage of training a NN starts. Note that the calculation example is for a single neuron at layer L(see figure 2.5). With multiple neurons an additional subscript is required for the weights and biases (see [39]). Now, $\nabla C_{\#}$ is be applied in the network to change the trainable parameters. For this, an optimizer algorithm is used for applying ∇C. The commonly used optimizer is the Stochastic gradient descent (SGD) [38], which is responsible for splitting S into batches. The SGD can be seen in equation 2.15.

$$\omega_{\#+1} = \omega_{\#} - \lambda \nabla C_{\#} \tag{2.15}$$

Note that $w \neq \omega$, with ω being the collections of **W** and **b** in the network. $\lambda =]0, \infty[$ is the learning rate[10] and regulates how much ω is changed after each iteration, which translates to how much is learned in each epoch. The optimizer used in this thesis is explained in section 2.3.4.

[10] It is a hyperparameter and has to be tuned for the optimal convergence of the network. While $\lambda > 1$ is possible, it is never done in practice. Commonly, $\lambda << 1$. Hyperparameter will be explained below.

This procedure is repeated a number of iterations, given by n. The number of iterations that have passed the complete dataset into the network is called an epoch. The training iterations of a network are usually measured in epochs, eg. if the dataset has 10000 entries and batch size 1000, one epoch will consist of 10 iterations of 1000 entries.

2.3 *Convolutional neural networks*

The actual network algorithm used in the thesis is a Convolutional Neural Network. In recent years, there has been a steep increase in the accuracy of image recognition algorithms and the common factor is the use of Convolutional Neural Networks (CNNs). The idea behind CNNs has been around for more than 30 years [27], but due to the Big Data aspect of ML, it was not widely used. Note that most of the math in the following section is from [39], and, for a more in-depth description of the fundamentals of ML, see [39].

At its core, CNNs are about mapping a weight matrix called a *kernel* to the $\mathcal{X}$. A kernel can be seen as an $n \times m$ neuron[11]. Using a regular image as $\mathcal{X}$, the input is a $H \times W \times D$ tensor, where D gives the number of channels in the image[12] and H and W give the height and width of the image. The expression for 2D convolutions (conv2d) is given as

[11] In practice, a square of size $n \times n$ is usually chosen.

[12] In standard images the channels are red-green-blue (RGB), so $D = 3$. RGB will give perception of most colors when working in an 8 bits colorspace.

$$Z(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i-m, j-n)K(m,n) \qquad (2.16)$$

where (i,j) is the dimensional position on the image, I is the input image, K is the kernel, and Z is the resulting feature map. Due to the nature of convolutions, the kernel is flip diagonally, but equation 2.16 is commutative [13]. Consequently, the kernel can be flipped relative to the input as (m,n) increases, making it more intuitive to multiply the kernel with the image. Therefore, most ML software libraries that implement CNNs actually do not implement convolutions, but rather cross-correlations[14], given by

[13] This means $S(i,j) = (I * K)(i,j) = (K * I)(i,j) = S(j,i)$.

[14] Including Tensorflow and Keras, which is the software used in this thesis.

$$Z(i,j) = (I \star K)(i,j) = \sum_m \sum_n I(i+m, j+n)K(m,n), \qquad (2.17)$$

Compared to equation 2.16, the sign within I has changed, so, when convoluting, the kernel will not be flipped. For simplicity reasons it will still be referred to as convolutions.
This is of course easiest to understand visually. Figure 2.6 illustrates conv2d on a 2D input using a 2 kernel[15]. The input is $H \times W \times D = 3 \times 4 \times 1$ and the kernel size[16] is 2×2. The image is in *gray-scale* as only a single channel depth is used. To draw parallels to NN, the

[15] Note that the kernel is not being flipped during convolution, so it is actually implementing cross-correlation.

[16] 3d kernels are also possible, they will do 3d convolution on tensors, however, it was tested in [18] and did not yield any additional performance. Thus, only conv2d will be used in this thesis.

individual kernel weights can be seen as the neurons in a hidden layer of a NN. The kernel weights are trainable parameters and an activation function is applied to the feature map Z (not shown in the figure). Kernels are also typically smaller than the image, meaning it is not fully connected. This increases statistical efficiency, because each trainable parameter is applied to multiple pixels in the image, but it also decreases memory usage as less parameters need to be stored.

In figure 2.6, only a single kernel is used. However, larger CNN's use multiple kernels, and the collection of kernels make up a filter ($k \times n \times m$, where k is the number of kernels).
Regarding the number of trainable parameter for a conv2d is given ((width of kernel · height of kernel · number of channels in previous layer+1) · number of filters), the $+1$ is for the bias and could be removed. As an example, a conv2d with a kernel size of $(3, 3)$ and 32 filters on a $(96, 96, 1)$ image will result in $(3 \cdot 3 \cdot 1) + 1) \cdot 32 = 320$ trainable parameters.

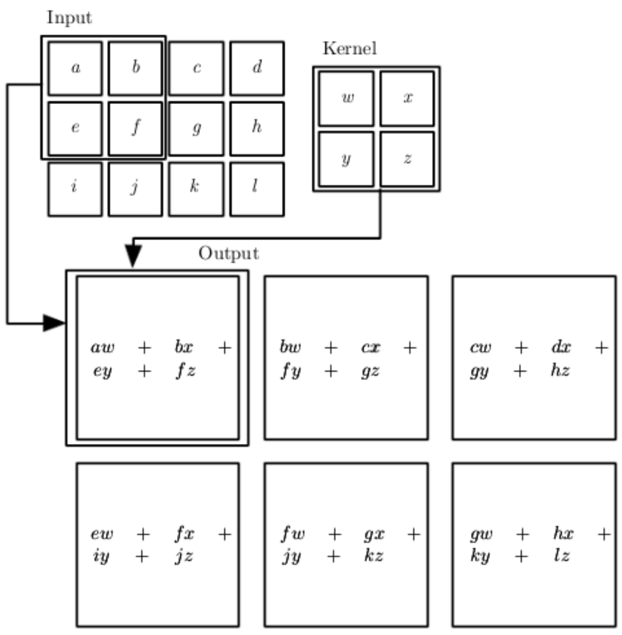


Figure 2.6: Sketch of cross-correlation (equation 2.17) between a input image $(3 \times 4 \times 1)$ and a kernel (2×2). For simplicity during the thesis, cross correlation will be referred to as convolutions. Figure from [19].

2.3.1 *Padding- and pooling- layers*

In figure 2.6, the input dimension changes from $3 \times 4 \rightarrow 2 \times 3$, meaning a reduction in both $H \times W$ after one convolution. Usually, CNNs tends to be deep, with multiple convolution layers, resulting in an unintended reduction in dimension. However, *zero padding* can account for this. *Zero padding* can be applied to increase or preserve the input dimension of a matrix by adding rows and/or columns of zeros as pixel-values to the matrix. *Zero padding* is applied before the conv2d layer and is illustrated visually in figure 2.7, where the dimensions of the matrix are kept constant. Sometimes down-

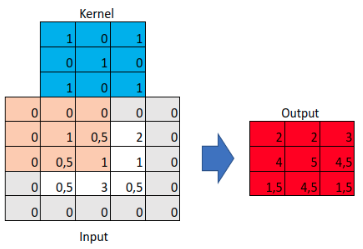


Figure 2.7: *Zero padding* has been performed on a 3×3 image. Afterwards, a 3×3 kernel is used. This results in an image that is 3×3, meaning the dimensions of the image is kept constant.

sampling is desired within a network and is usually reserved for *pooling* layers. An example of *max-pooling* can be seen in figure 2.8. Here, the *pooling* layer takes the maximum value of a pixel within the four same-colored squares. The matrix is reduced from a 4×4 to a 2×2. The two most common pooling layers are *max-pooling* (explained above) and *average-pooling*, where the average pixel-values are calculated within the pooling matrix. Pooling helps increase generalization as it introduces invariance to small translations of the input, meaning if the input is shifted by a small amount, most of the pooled outputs do not change. The computational cost through the network will also be reduced due to down-sampling of the matrices.

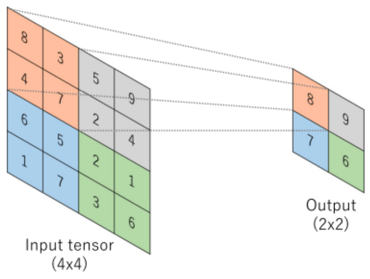


Figure 2.8: 2×2 *max-pooling* applied on a 4×4 matrix, resulting in a 2×2 matrix. Figure from [50].

In addition to the *pooling* layers, there is a single *GlobalPooling* layer after the last conv2d layer. This layer prepares the output of the CNN to be the input of a NN that uses the features extrapolated by the convolutions to estimate of Y, eg. whether the object in the image is a cat or a dog. However, NNs can only utilize vector data, so the tensor output of the CNN has to be re-scaled. By using a *GlobalPooling* layer, the tensor can be re-scaled to a vector using either a *max-* or *average- pooling* of the channels or *flattening* of the tensor. Say the output of a CNN is $3 \times 3 \times 256$. Applying *max-* or *average-pooling* (3×3) will tehn result in an output of 256, whereas applying *flattening* will result in an output of $3 \cdot 3 \cdot 256 = 2304$.

2.3.2 *Feature-wise Linear Modulation*

Convolutional neural networks can be very powerful for detecting patterns in images. However, one area where an ordinary CNN architecture lacks precision is adding context[17] to an image. Inputs are usually in the form of 2D or 3D matrices of pixel-values, but an important feature to an image might be a scalar variable in the form of the place the image was taken or the date of the image, etc. This could give better context to the image. In our case, in a detector environment, this might be $\langle \mu \rangle$, the average number of interactions per crossing, which might indicate the amount of pileup in an image.

[17] Context meaning that the algorithm is supposed to learn the importance of the image channels and weight them accordingly. These weights are dependent on variables related to the image.

This information is added to the channels of the CNN using a *Feature-wise Linear Modulation* layer, also called *FiLM*. Proposed by [17], a *FiLM* adds the ability to use scalar variables for adding context to an image. The context is in the form of an affine transformation of the channels after convolution. First, a *FiLM generator* (FiLM gen. is essentially a NN) is used to modulate f and h

$$\gamma_{i,c} = f_c(\mathbf{x}_i) \quad \text{and} \quad \beta_{i,c} = h_c(\mathbf{x}_i) \tag{2.18}$$

where $f = h$ since the same FiLM gen. is used. The output is $\gamma_{i,c}$ and $\beta_{i,c}$ for the ith input of the cth channel. The FiLM layer between the conv2d and activation layers applies affine transformation that scales and shifts the channels in the CNN, given by

$$FiLM(x_i) = \gamma_{i,c} z_{i,c} + \beta_{i,c}, \tag{2.19}$$

with $z_{i,c}$ being the output of the ith conv2d and the cth channel. In figure2.9, the affine transformation of the channels can be seen. The FiLM also has the ability to turn a channel completely off by just setting the ith and cth γ and β values to zero. FiLM gen. is integrated into the back-propagation, giving it the ability to learn the affine transformation that best maps to the desired output. This makes FiLM a powerful algorithm for adding context in images.

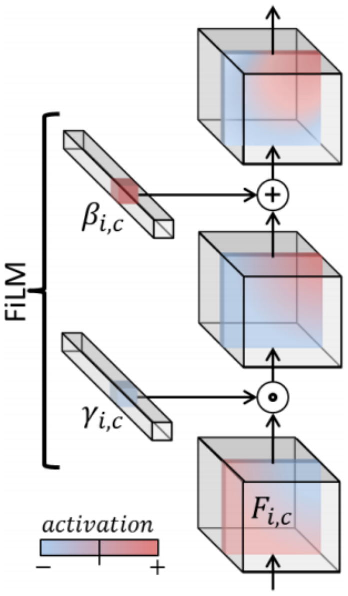


Figure 2.9: $\gamma_{i,c}$ and $\beta_{i,c}$ are created from *FiLM gen*, and then the *FiLM* layer within the CNN applies the transformations to the channels. $F_{i,c}$ is the cth channel of ith convolution. Figure from [17].

2.3.3 *Regularization - The eternal fight for generalization*

As mentioned above, in section 2.1.2, the eternal struggle is the generalize of ML algorithms to S, so the difference between empirical and expected loss is minimized. Due to the size and customizability of CNNs, they can be prone to over-fit. In this section, some key components in forcing CNNs to generalize, avoid over-fitting, and techniques to speed up the evaluation of a CNN will be introduced.

Dropout

Dropout layers are a technique to force neurons to generalize more by turning off neurons in each epoch. When an epoch starts, Dropout selects (given some probability density function a set of neurons. These selected neurons are neutralized, meaning they are turned off and not being used in the following epoch. Afterwards, a new random set is neutralized and the previous are activated. This causes the network to become more robust and the neurons to generalize, since neurons that might recognize specific patterns could be neutralized in the following epoch and now this pattern has to be modeled by the existing neurons. Furthermore, it also helps to prevent the network from over-fitting to S.

Early Stopping and ModelCheckpoint

Early stopping and ModelCheckpoint are a very simple and useful tools against over-fitting and endless convergence time. ModelCheckpoint ensures that when a minimum of a model, h^*, is found (see figure 2.1), the parameters of the model is saved.
The Early Stopping tool compared the validation score of the currently each epoch to the previous validation scores. If the new score is lower [18] than the previous, the ModelCheckpoint algorithm saves the parameter and training continues. However, if the new score is higher than the previous, a counter starts. If the score has not decreased given n number of epochs[19], the Early Stopping algorithm will terminate the training and select the previous checkpoint. Stopping the training after the validation score has stopped decreasing will help speed up the algorithm as it does not train endlessly, eg. if the minimum is found after 50 epochs, it should not continue until 1000 epochs.

[18] Assuming minimization problem.

[19] This is chosen by the user.

Batch Normalization

Batch normalization [21] layers (BN) are a clever way to improve the performance, speed, and stability of a NN by standardizing the output of each neuron using

$$BN(x_i) = \gamma_i \cdot \frac{x_i - \mu_i}{\sigma_i} + \beta_i, \tag{2.20}$$

with x_i being the output of the layer, μ_i and σ_i the mean and standard deviation of x_i (non-trainable) over a mini-batch. The γ_i and β_i are trainable parameters of BN used to scale and shift the normalized values. The reasoning behind BN can be found in back-propagation (section 2.2.3) and activation functions (figure 2.4), as the back-propagation uses the gradient of the activation function, a very subtle gradient is preferred. Looking at figure 2.4, most of the gradient within the activation functions are at $x \approx 0$ and, due to the use of gradients in the back-propagation, networks will prefer the output of neurons to be normalized values close to $x \approx 0$.

Hyperparameters

Hyperparameter are used to regulate the behavior of an ML algorithm. They control the learning process of the algorithms. Contrary to the trainable parameter of a network, the hyperparameters are user-define and are typically constant throughout the training process. For NN the important hyperparameter to consider is often number of hidden layer, number of neurons in each hidden layer, the activation function used in each layer, learning rate used in training, the batch size or the loss function. When optimizing an architecture, optimizing the configuration of hyperparameter is often what is meant as different configuration of hyperparameter will have a big influence of the performance of a network.

2.3.4 *Optimizing learning rate*

The learning rate $\lambda =]0, \infty[$ (normally $0 < \lambda < 1$) was quick addressed in section 2.2.3 and will be described in more detail below. The learning rate is the hyperparameter responsible for regulating the amount learned after each iteration, see equation 2.15. This will in terms affect the convergence time and sensibility to small local minima. Too low of a learning rate ($\lambda \approx 0$) will lead to the model being slow and highly sensitive to local minima and too large of a learning rate ($\lambda \approx 1$) might result in the model diverging from the global minima and not be able to converge to a minimum. Both behaviors can be seen in figure 2.10. Due to the importance of λ and the complexity of the loss function space, multiple algorithms have been proposed to tackle this problem. In general, there are two types, *Optimizers* and *Learning-Rate-Schedulers* (LRS). An *Optimizer* affects how learning is applied to the gradient descent. Rather than being applied linearly, as in equation 2.15, an *Optimizer* transforms the gradient term. In

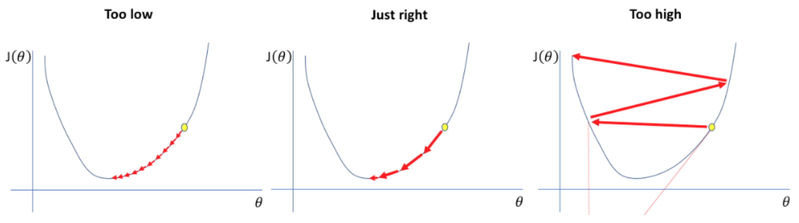


Figure 2.10: θ is the parameters of the network and $J(\theta)$ is the loss function. While $J(\theta)$ is very simple, an increase of the complexity of θ will lead to an increase of the complexity of $J(\theta)$ as well. Figure from [23]

this thesis, Nadam [16] is used as the *Optimizer*. It incorporates momentum[20]. A LRS controls the size of λ. Usually, a range of λ is set, and the LRS then changes the λ after each iteration[21], helping the algorithm convergence to a minimum.

[20] Taking previous gradients into account.
[21] Usually from a high λ to a low λ.

Optimizers

Optimizers were shortly mentioned in section 2.2.3, where it was shown how they are used in the learning process of a network. The SGD was shown, which is a very simple optimizer, which only takes the recent back-propagation into account. However, the previous gradients from back-propagation might also be of importance to the learning network. Hence, many optimizers also add *momentum* to their loss function, given by

$$\omega_{\#+1} = \omega_{\#} - \lambda \nabla C_{\#} - m_{\#}, \quad m_{\#} = \gamma \nabla C_{\#-1} \tag{2.21}$$

where $m_{\#} = \gamma \nabla C_{\#-1}$ is the *momentum*, which is the fraction of the past gradients. $\gamma =]0, \infty[$ regulates the amount the previous gradient is used, usually $\gamma < 1$. *Momentum* has many benefits. It reduces the convergence time as well as the oscillation of the loss function.
There exists many kinds of optimizers specialized for different problems. An overview of the most commonly used can be seen in [38], which also elaborates further on the inner workings of many optimizers. The optimizer used in this thesis is Nadam, as it was found to be the most accurate in [18].

NADAM

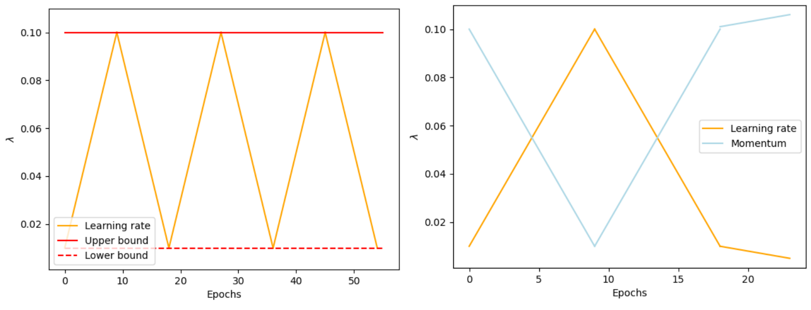
As it is the optimizer used, we will give a short introduction into the fundamental mechanics of Nadam. Nadam is an acronym for Nesterov-accelerated Adaptive Moment Estimation, which is a combination of Adam and Nesterov accelerated gradients with a modified momentum term. Nadam is given by

$$\omega_{\#+1} = \omega_{\#} - \frac{\lambda}{\sqrt{\hat{v}_{\#}} + \epsilon} \left(\beta_1 \hat{m}_{\#} + \frac{(1-\beta_1)\nabla C_{\#}}{1-\beta_1^{\#}} \right), \tag{2.22}$$

This optimizer can be a little confusing, as it is a combination of two optimizers. However, it works as follows. The $\beta \approx 1$ and $\epsilon = 10^{-8}$ are estimated empirically [38]. Both $\hat{v}_{\#}$ and $\hat{m}_{\#}$ are dependent on the past $\nabla C_{\#}$ and $(\nabla C_{\#})^2$, respectively. In essence, this optimizer is working well with momentum, as $\hat{m}_{\#}$ is a exponentially decaying average of previous gradients and $\hat{v}_{\#}$ is $\hat{m}_{\#}^{2}$. The NAG adds a persistence to the momentum, so the optimizer slow down when the momentum becomes too large, which can be a problem for Adam.

Learning Rate Schedulers

As mentioned above, a LRS affects the learning rate of a network by changing the value of λ after each iteration (see equation 2.15), which will affect the amount learned after each iteration. There are many variation of LRS, and they do not all simply decrease λ for convergence to a specific minimum in the loss space. The two LRS we will be focusing on are Cyclical Learning Rates [41], where λ both decreases and increases. The reasoning is to insure that the network is not converging to a local minima, and can therefore "jump" out of the minima due to the large λ. The two types of LRS can be seen in figure 2.11.



(a) Cyclic learning rate (CLR) **(b)** 1Cycle Learning rate (oneCLR)

Figure 2.11: Example of 1cycle LRS and cyclical LRS. The cyclical LRS has additional hyperparameters that can change the behavior of the learning rate eg. the upper bound can exponentially decay as a function of iterations.

For CLR, an upper bound, lower bound and step size are defined. The bounds control the range of the learning rate, where the step size controls the number of iterations until the bounds. Additional functions can be added to CLR, eg. making the upper bound exponentially decay. oneCLR is similar to CLR, but instead of running for multiple step_sizes, oneCLR only runs two step sizes from its lower to upper bound and back again. It concludes by reducing its lower bound with $1/10$ for n number of iterations. The result is that oneCLR converges quicker, after ≈ 50 epochs, where CLR might need > 100 epochs. However, this also means that step_size is an import hyperparameter for oneCLR as it is not meant to continue after two steps.

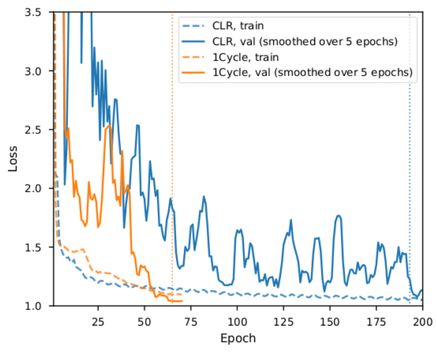


Figure 2.12: Figure from [18]. The figure illustrates the number of epochs needed for CLR and oneCLR to convergence to the same loss. It can be seen that oneCLR is many epochs faster than CLR.

3 Pre-processing of data

Obtaining clean and readable data plays a crucial role in every Machine Learning pipeline. Due to the amount of data produced in a particle collision, ATLAS has constructed their own file format called `.root`, which has different levels dependent on the needs of the analysis. In this thesis, we will be working with xAOD and DxAOD, which contain both raw and reconstructed particle properties from multiple collisions. They require a pre-processing framework before they can be processed by *Python*. In this chapter, we will go through the different pre-processing stages to produce a sample optimized for *Tensorflow* in *Python*.

To access the files used in this thesis, access to *Rucio* [10], a database maintained by *CERN*, is required. The framework used to process xAOD and DxAOD[1] is built from the *Athena* framework[14] developed by the CERN Collaboration. The specific framework used in this thesis was developed by Lukas Ehrke and Daniel Nielsen, with additional adjustments made by us. The framework is called *NTupleProduction* and can be found at [3]. All the data files used in this thesis are available in xAOD or DxAOD format on *Rucio*, and the

[1] xAOD and DxAOD are file formats downloaded from *Rucio*.

Decay channel	Container	Type	Deviation
$Z \to ee$	mc16_13TeV.361106.PowhegPythia8EvtGen_AZNLOCTEQ6L1_Zee.deriv. DxAOD_EGAM1.e3601_e5984_s3126_r10201_r10210_p4089	MC	DxAOD
	data16_13TeV.periodAllYear.physics_Main.PhysCont .DxAOD_EGAM1.grp16_v01_p4088	Data	DxAOD
Electron Gun	mc16_13TeV.423000.ParticleGun_single_electron_ egammaET.recon.AOD.e3566_s3113_r9364	MC	xAOD
$Z \to \mu\mu\gamma$	mc16_13TeV.301536.Sherpa_CT10_mumugammaPt10_35.deriv. DxAOD_EGAM4.e3952_s3126_r10201_r10210_p3956	MC	DxAOD
	mc16_13TeV.301903.Sherpa_CT10_mumugammaPt70_140.deriv. DxAOD_EGAM4.e3952_s3126_r10201_r10210_p3956	MC	DxAOD
	data16_13TeV.periodAllYear.physics_Main.PhysCont. DAOD_EGAM4.grp16_v01_p3948	Data	DxAOD
$H \to \gamma\gamma$	mc16_13TeV.346214.PowhegPy8EG_NNPDF30_AZNLOCTEQ6L1_ VBFH125_gamgam.merge.AOD.e6970_e5984_s3126_r10724_r10726	MC	xAOD
	mc16_13TeV.343981.PowhegPythia8EvtGen_NNLOPS_nnlo_30_ ggH125_gamgam.merge.AOD.e5607_e5984_s3126_r10724_r10726	MC	xAOD

Table 3.1: The table lists the containers used in the analysis. Decay channel, name of the container, type of data, and at which deviation level the files are in can all be seen in the table. Each container consists of multiple xAOD or DxAOD files.

files used can be seen in table 3.1.

3.1 *xAOD to DxAOD*

Many petabytes of data are recorded when running the ATLAS experiment. However, depending on the physics analysis, only a fraction of the data is of interest, so removing the unused events can reduce the sample size tremendously. As seen in table 3.1, some data files are in the xAOD format and have to be converged to the DxAOD format. ATLAS has developed a framework called *DerivationFramework* (DF)[5] within *Athena* to perform conversions of xAOD to DxAOD. We will be using it to create DxAOD from xAOD. DF has four key operations

- Skimming: Removing whole events
- Thinning: Removing whole objects from within an event but keeping the rest of the event
- Slimming: Removing information from within objects but keeping the rest of the object
- Augmentation: Adding data not found in the input data

By default, the physics analysis groups at ATLAS have their own derivations, which is a combination of the four operations above. Some derivations can be seen in table 3.2. The two derivations used in this thesis are EGAM1 and EGAM4. Minor changes have been added to the EGAM4 template to proper handle $H \rightarrow \gamma\gamma$. The pipeline of xAOD to DxAOD can be seen in figure 3.1. Both xAOD and DxAOD are in *.root* format. $Z \rightarrow ee$ and $Z \rightarrow \mu\mu\gamma$ are already in the DxAOD format but $H \rightarrow \gamma\gamma$ and the Electron-Gun (EG) are not. Thus, the DF will be used to create $H \rightarrow \gamma\gamma$ in EGAM4 and EG in EGAM1. DxAOD for both $H \rightarrow \gamma\gamma$ and $H \rightarrow Z\gamma$ can be found in *Rucio*, but they are in the HIGG1D1 and HIGG1D2 deviations that do not contain the necessary information for the analysis, namely the ECAL image information is missing. The EGAM is therefore used instead[2].

Name	Selection
EGAM1	$Z \rightarrow ee$, central eletrons.
EGAM2	$J/\psi \rightarrow ee$
EGAM3	$Z \rightarrow eee$, $Z \rightarrow ee\gamma$
EGAM4	$Z \rightarrow \mu\mu$, $Z \rightarrow \mu\mu\gamma$
EGAM5	$W \rightarrow ev$
EGAM6	$Z \rightarrow ee$, more loose than EGAM1
EGAM7	Fake electron sample
EGAM8	$Z \rightarrow ee$ with least one forward electron
EGAM9	γ trigger efficiency measurements
HIGG1D1	$H \rightarrow \gamma\gamma$
HIGG1D2	$H \rightarrow Z\gamma$

Table 3.2: The table shows a summary of [26] and [29]. $Z \rightarrow ee$ and $Z \rightarrow \mu\mu\gamma$ are using EGAM1 and EGAM4, respectively. For $H \rightarrow \gamma\gamma$, EGAM4 is used, because HIGG1D1 and HIGG1D2 do not contain images.

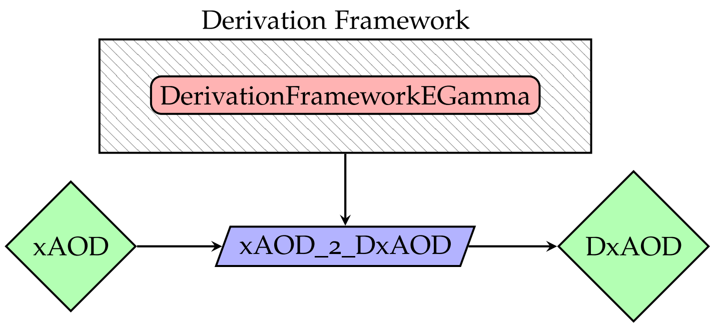


Figure 3.1: The pipeline from xAOD to DxAOD. Green symbolizes a collection of data files, red are C++ files and blue is the folder that combines the files. The striped grid indicates that the algorithm is a part of *Athena* [14].

[2] HIGG1D1 or HIGG1D2 could be used by removing slimming of the necessary information for images.

3.2 *DxAOD to h5*

DxAOD is the most common starting format for physics analyses. It has a predefined derivation, like EGAM1. However, the format still needs further tinkering before it can be used by *Python*. Figure 3.2 illustrates the DxAOD pipeline from `.root`→`.tfrecord`. The figure displays the key points of the pipeline. The `.tfrecord` format is used as it is optimized for High-performance-computing (HPC) with fast transfer speed and low memory consumption. First, the DxAOD files are selected from *Rucio* in `.root.1` format by the user. The files selected are then used as input in the *NTupleProduction* framework. It extracts and constructs the necessary variables from the `.root.1` and removes most of the *branch* structure of the file. It up-samples the ECAL images data to the correct size, and applies thinning and slimming to remove information not of interest to the analysis. This converts the files into an Ntuples (`.root`) format. Next, root2hdf5 constructs *Python*-friendly files from `.root` files. The root2hdf5 framework formats the `.root` file to `.h5` of *numpy* arrays, but all rows are still event-based (Event-based is where all particles from one event is in the same row.). The framework also selects events that might be of interest using *pdgID* and *truthOrigin* for MC and ATLAS Likelihood *Loose* for Data.

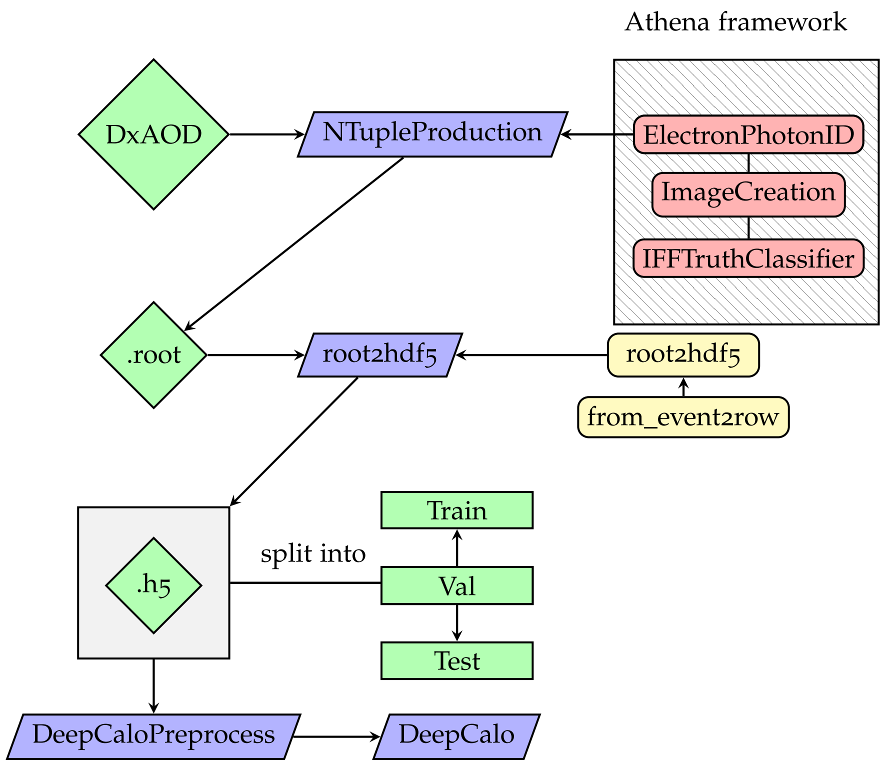


Figure 3.2: The figure shows the pipeline from DxAOD to `.h5`. Green symbolizes data files, red are C++ files, yellow are *Python* files, blue are the folders that combines the scripts. The striped grid indicates that the algorithm is a part of *Athena* [14].

Afterwards, from_event2row *flattens* the event-based samples into a row-based sample, where each row represents a particle and its variables. The images are also reshaped from column vectors to matrices, the barrel and end-cap images are added together and the tile gap cells are summed into a scalar variable. The events are also separated into train, test, and validation samples.

Finally, the DeepCaloPreprocess uses the `.h5` files to construct `.tfrecord` files. Here, the variables are normalized and an additional event selection is used to cross check if the events are correctly selected. When the `.tfrecord` files have be obtained, they are ready for *DeepCalo*. For more information on `.tfrecord`, see section 4.1. The root2hdf5 and DeepCaloPreprocess frameworks have been developed in this thesis.

3.3 *Image format and data types*

Above, we described the process of constructing the correct file format. Below, we will first describe the type of variables used in *DeepCalo* and elaborate on the standardization of variables in *DeepCaloPreprocess* from figure 3.2.
One of the great advantages of a NN is its flexibility towards input types. While it is not limitless, it has strong applications when it comes to image and scalar input (see section 2.2). This allows the network to utilize the many collision properties measured in the detector. The variables used by *DeepCalo* are the scalar properties of the event, the track properties from tracks within $\Delta R < 0.4$ of the event, and the ECAL images from all four layers.

3.3.1 *Scalar properties*

Scalar properties selected for *DeepCalo* indicate the cell position and energy size of the event in the detector. The scalar properties are mostly information related to the ECAL and could add context to the images. Table 3.3 gives a short description of each scalar and figure 3.3 illustrates their distributions in both MC and Data. Some properties can only be measured for charged particles, and thus these are not present for photons. This is stated in table 3.3. Note that pileup ($\langle\mu\rangle$) is not re-weighted because the re-weighting files could not be found. Further, due to the large misalignment between MC and Data, pileup will not be used in the final model, but will likely be very important variables as an indicator of noise related to the images. The scalar variables selected are the same as in [18], where experiments using BDTs were trained on many scalar variables, and by using SHAP [28] 16 scalar variables were selected.

3.3.2 *Track properties*

The track variables, which are sequence data and could help to indicate the amount of pileup present in the images. *pp* collisions are messy, and when a particle deposits energy in a region, it might not be isolated from other particles. Often, multiple track particles close to the target will deposit energy within the same region and increase the amount of energy present in the images. Therefore, is it impor-

Type	Name	Description
Energy	E_{acc}	Energy deposit in layer 1-3 of ECAL.
	η_{index}	η cell index of cluster of layer 2.
	$f0_{cluster}$	Ratio of energy between layer 0 and E_{acc} in $\|\eta\| < 1.8$ (end of layer 0).
	$R12$	Ratio of energy between layer 1 and 2 in the ECAL.
	p_t^{track}	p_T estimated from tracking for the particle (only e).
	E_{TG3}	Ratio between the energy in the crack scintillators and E_{acc} within $1.4 < \|\eta\| < 1.6$.
	$E_{tile-gap}$	Sum of the energy deposited in the tile-gap.
Geometric	η	Pseudorapidity of the particle.
	$\Delta\phi_2^{rescaled}$	Difference between ϕ, as extrapolated by tracking, use for ECAL momentum estimation and ϕ of the ECAL cluster.
	η_{ModCalo}	Relative η position w.r.t. the cell edge of layer 2 in the ECAL*.
	$\Delta\eta_2$	Difference between η, as extrapolated by tracking, use for ECAL momentum estimation and η of the ECAL cluster (only e).
	$poscs_2$	Relative position of η within cell in layer 2 in ECAL. $2(\eta_{cluster} - \eta_{maxEcell})/0.025 - 1$, $\eta_{cluster}$ is η of the barycenter of the cluster and $\eta_{maxEcell}$ is η of the most energetic cell of the cluster.
	$\Delta\phi_{TH3}$	Relative position in ϕ in a cell. $\text{mod}(2\pi + \phi, \pi/32) - \pi/32$.
Misc.	$\langle\mu\rangle$	Average proton-proton interaction per bunch crossing.
	n_{tracks}	# of tracks assigned (only e).
	$n_{vertexReco}$	Number of reconstructed vertices.

Table 3.3: The table shows the scalar variables that will be used in *DeepCalo*. Some variables are only measured for the electron. This is indicated using "only e" in the table. $*$ assumes a constant cell size in η of 0.025.

tant for the network to know the properties of track particles present in its region in order to correct for the pileup effect. The track properties are therefore added to the network. A description of the variables can be seen in table 3.4 and, to understand the geometric behind the variables, see figure 9.20. Only tracks within $\Delta R < 0.4$ are selected. The number of tracks will vary between events and are sorted by

$$\text{Iso}_{\text{TP}} = \sum p_T^a \cdot e^{-b\frac{\Delta R}{\sigma_R}}, \quad b = a = 1, \tag{3.1}$$

where a and b indicate the importance of momentum or ΔR. If sorting by distance to the target particle, ΔR must be more important and therefore $a < b$ and vice versa for momentum. The selection of a constant number of tracks is explained in chapter 4.

Table 3.4: The table shows the track variables used in *DeepCalo*. The variables should help the network understand how the pileup contributes to the energy in the ECAL images. Tracks are only left by charged particles.

Type	Name	Description
Energy	$p_{t,track}/q_{track}$	Transverse momentum of track divided by its charge q
Geometric	d_0/σ_{d0}	$d0$ is the signed transverse distance between the point of closest approach and the z-axis where σ_{d0} is its uncertainty
	ΔR	$\Delta R = \sqrt{(\phi_0 - \phi)^2 + (\eta_0 - \eta)^2}$
	$\text{vertex}_{\text{track}}$	Reconstructed vertex of the track
	z_0	Longitudinal distance between the point of closest approach and the z-axis.
	η_{track}	Reconstructed $\|\eta\|$ of tracks.
	ϕ_{track}	Reconstructed ϕ of tracks.
Misc.	n_{pixel}	Number of hits in the pixel detector
	n_{SCT}	Number of hits in the SCT
	n_{TRT}	Number of hits in the TRT

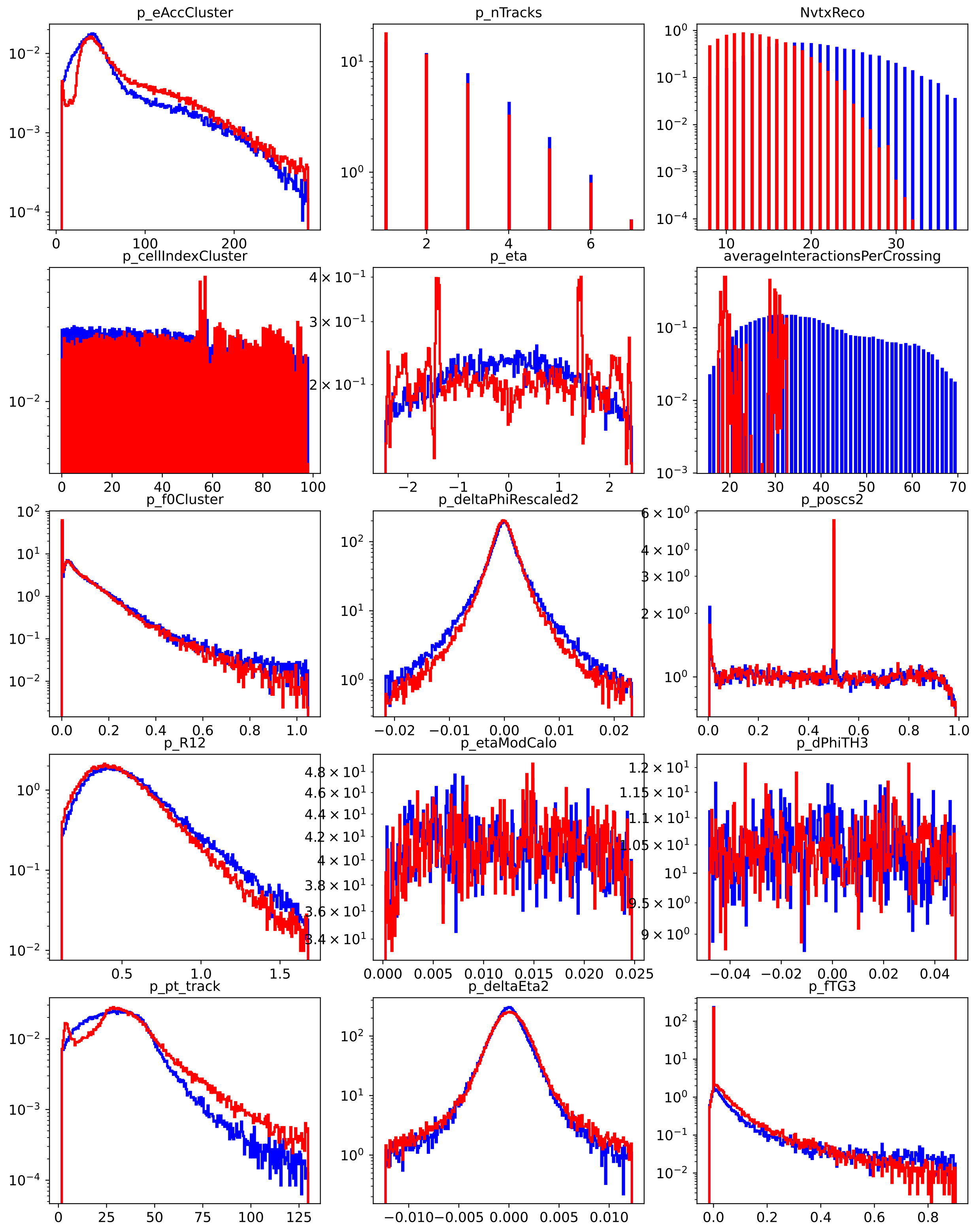


Figure 3.3: The figures show the distributions of scalar variables in MC and Data. The scalar distributions belong to the $Z \to ee$ channel. The blue-colored histogram is MC and the red one is Data.

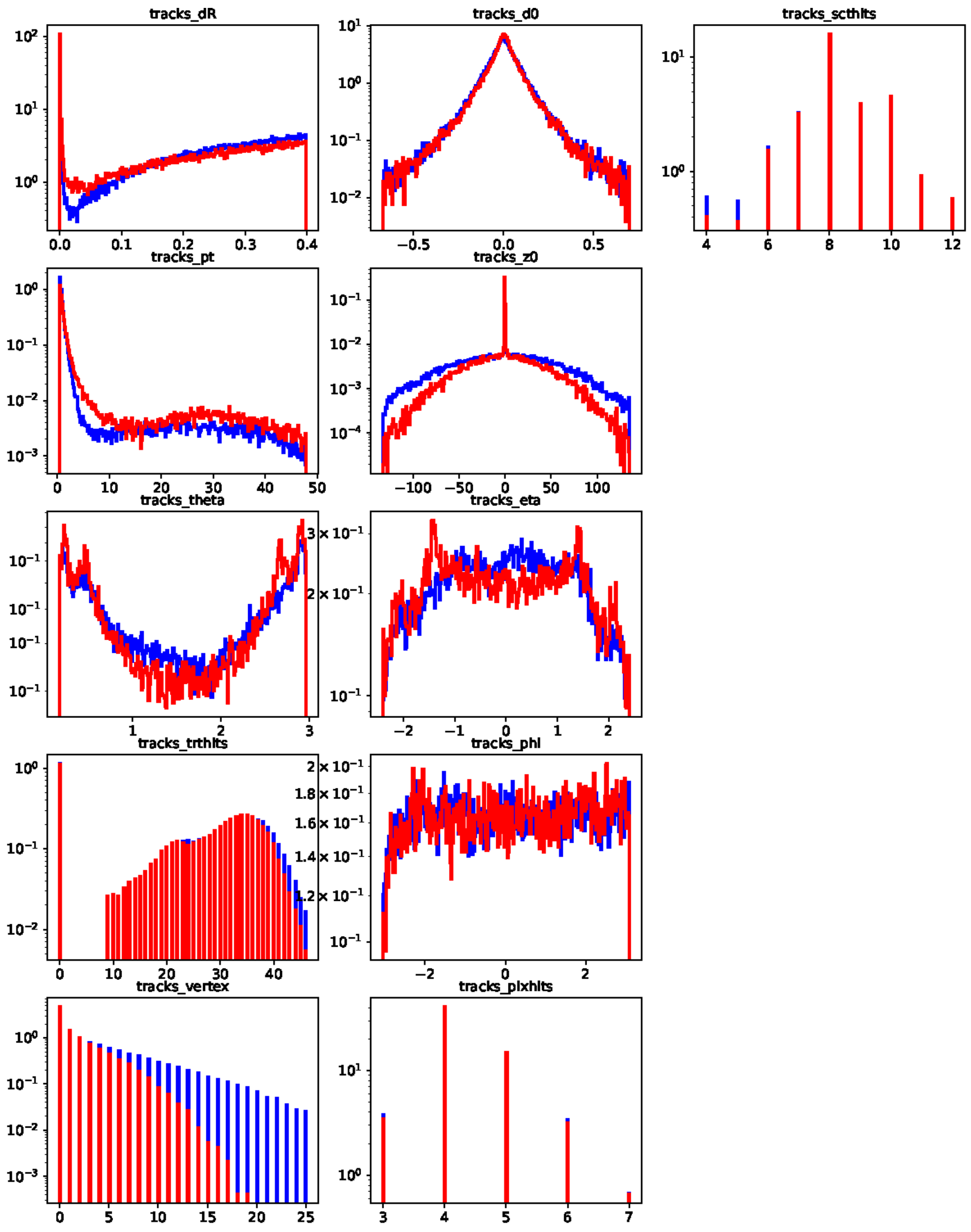


Figure 3.4: The figures show the distributions of track variables in MC and Data. The track distributions belong to the $Z \to ee$ channel. The blue-colored histogram is MC and the red one is Data.

3.3.3 Images

The information within the ECAL images is the basis for constructing a CNN to reconstruct the energy of particles. We assume that the raw ECAL cell contains more information than the scalar variables used in $Ecalib^{(BDT)}$as a CNN may be able to estimate attributes to the energy better with the ECAL images.
From table 1.1 and figure 1.7, we see the cell size in $(\Delta\eta_l \times \Delta\phi_l)$ between layers $l \in [1, 2, 3, 4]$ in the ECAL will vary, resulting in a varying resolution between the layers. This will pose a problem for a CNN as the $(\Delta\eta \times \Delta\phi)$ dimensions have to be identical between layers, as seen in section 2. However, this can be solved by up-sampling the images to an identical resolution.

However, cells must be selected first. Cells are obtained by using a fixed window centered at the barycentre of the event. The ATLAS e/γ group recommends a window size of $(\eta, \phi) = (7 \times 11)$ with respect to layer 2 (with cell size $(\Delta\eta_2 \times \Delta\phi_2) = 0.025 \times 0.025$). This should encapsulate most of the cluster without captioning to much pileup. In figure 3.7, an example of the cell selection procedure can be seen with the additional up-sampling.

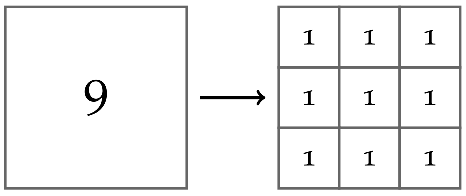


Figure 3.5: Up-sampling works by dividing the cell into small subcells with the total sum and size of the original cell. Here, the large cell with energy 9 is scaled to 9 smaller cells with size and energy 1/9 of the large cell.

The composition of window sizes with respect to layer 2 can be seen in table 3.5, which also reveals the up-sampling of ϕ and η compared to layer 2. The common resolution between the layers is chosen to be $\Delta\eta_1 \times \Delta\phi_2 = 56 \times 11$ as we see the highest resolution in ϕ in layer 2 and in layer 1 for η. Thus, no information will be removed from the image. Cells may extend beyond the 7×11 window, if they do the additional cells beyond will be removed after up-sampling.

Each cell of layers i must be sampled to achieve the common resolution. The amount of scale is controlled by $w_i = \Delta\eta_i/\Delta\eta_1$ and $h_i = \Delta\phi_i/\Delta\phi_2$. To conserve the pixel value of the cell, each resulting cell receives a value of $E_c/(w_i h_i)$, where E_c is the energy of the undivided cell. An example of the up-sampling can be seen in figure 3.5, where we see a uniform up-sampling between η and ϕ that is not the case for the ECAL images.

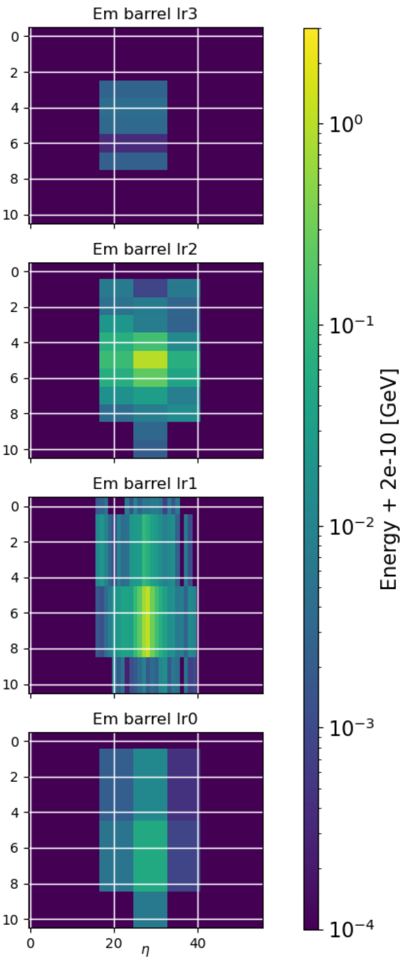


Figure 3.6: The figures illustrates the ECAL energy images in MC for $Z \to ee$ after upsampling. The time images from the same event can be seen in figure 3.8. Note that the color-map is in logimatic scale and the unit is GeV. The previous resolution of the layers are still noticeable.

After a common resolution has been achieved, we are able to fill in the pixel values. Up-sampling the images has the advantage of centering the barycentre more closely to the center of the image. The result can be seen in figures 3.6 and 3.8 for energy and time, respectively.

The up-sampling algorithm is a part of the *NTupleProduction* framework (see figure 3.2), with up-sampling performed by the *ImageCreation* script. This framework constructs an up-samled column vector, as `.root` files are not able to store matrices. The *root2hdf5* constructs matrices from the column vectors. The ECAL images have a barrel, an end-cap, and a crack region of the detector. To reduce the complexity of handling the many image channels , the barrel and endcap images are added together while the crack region is summed to a scalar variable called $E_{tile-gap}$. The end result is images with dimensions $(56, 11, 4)$. It is not only energy that is measured from the cell; additional properties like *time*, *gain* and *noise* are also a part of the ECAL image. *Gain* is the re-scaling of the energy size of the ECAL

	Presampler	Layer 1	Layer 2	Layer 3
Height (ϕ)	$\frac{11}{4}$	$\frac{11}{4}$	11	11
Width (η)	7	56	7	$\frac{7}{2}$
$\Delta\eta \times \Delta\phi$	0.025×0.1	$\frac{0.025}{8} \times 0.1$	0.025×0.025	0.05×0.025

Table 3.5: Dimensions of the layers in ECAL with respect to layer 2. This can be used to illustrate how each ECAL layer images must be up-sampled so they all have the same dimension.

images, and will mostly contain $[1, 2, 3]$ as values. *Noise* is the amount of noise related to each cell in the ECAL. It measures the background noise of the ECAL cells when no collision occurs. Both variables will not be used in this thesis.

However, *time* will be used. *time* is the time registered when a cell is activated, relative to the ATLAS clock. At time $t = 0$ the interaction occurs and an imagined sphere propagates outwards from the interaction point at the speed of light. Everything within the sphere has a negative time and everything outside has a positive time. Due to slow read-out time compared to the fast bunch crossings, a buffer is added to the cell time. The time between bunches is 25 ns, so cells activated significantly far from $t = 0$ should be considered out-of-time pileup and should not be processed as a part of the event. Making a decisive cut for the out-of-time pileup can be difficult but may be *learned* by *DeepCalo*.

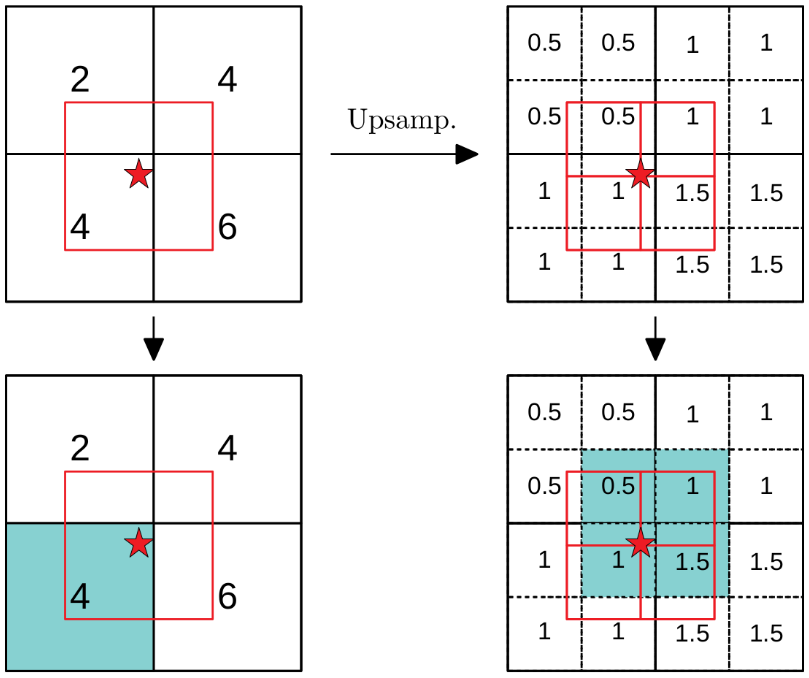


Figure 3.7: Figure from [18]. The figure illustrates the reason cells are selected after re-sampled, showing the results from cell selection with and without up-sampling. The red star indicates the barycentre, where the particle hit. We when wish to construct a 1×1 image indicated by the red square. The left columns shows the cells selected if the image was not up-sampled. It can be seen that the barycentre is not centered in the image. In the right columns, the image has been up-sampled before selecting the cell of the 1×1 image. It can be seen that the barycentre is now more centered than in the previous selection.

3.4 *Pre-processing*

The next vital step is pre-processing the data. *"Garbage in, garba out"* is a common phrase used to state that ML models trained c flawed data will give flawed predictions. Many of these flaws ca be fixed during the pre-processing phase. In the sections above, v have explored the data measured at ATLAS in its relatively raw stat In the following, we will elaborate on the transformations used c the samples to prepare them for the *DeepCalo* model. The re-sampl of images is explained above and no further pre-processing will fc lowNote that some ECAL images might come in MeV and should l scaled to GeV..

Standardizing variables

The standardizing of variables are performed by the *DeepCaloPreprocess*. In chapter 2, we saw that neurons in the hidden layers perform linear transformations on the input and map it to an activation function

$$\mathbf{h} = g(\mathbf{W}^T\mathbf{x} + \mathbf{b}). \tag{3.2}$$

This makes weights in the neurons scale-dependent. Recall back-propagation (in equation 2.13 and 2.15), where gradients are dependent on the trainable parameters. Thus, if the magnitudes of the inputs are not at the same scale, the neurons weigh inputs with high variance heavier and these will thus update much faster than inputs with lower variance. Therefore, standardizing input variables is important for the performance of the network. In addition to the gradient, most activation functions (See figure 2.4) will only have curvature[3] close to the origin, so the network will learn most (for back-propagation) close to the origin.

There are many types of standardizing, all with different behaviors of handling distributions. A discussion in standardization performance for *DeepCalo* took place in [18], where the conclusion was to apply a combination of the QuantileTransformer[4] to handle heavy-tailed distributions and apply the RobustScaler for the rest. For a more detailed description of these algorithms, see section 9.6.3.

Looking at figures 3.3 and 3.4, we see multiple heavy-tailed features, which will be standardized using the QuantileTransformer. The transformed distribution used in *DeepCalo* can be seen in figures 9.24 and 9.25. Additionally, in figures 9.22 and 9.23, only the RobustScaler has been applied for standardization. Here, it is clear that outliers and tails are still present.

Target energies

The particles deposit most of their energy in the ECAL. Consequently, a good approximation for E_{truth} is E_{acc}. Thus, instead of having *DeepCalo* predict E_{truth} with a large range, it predicts r_{truth}, as seen in equation

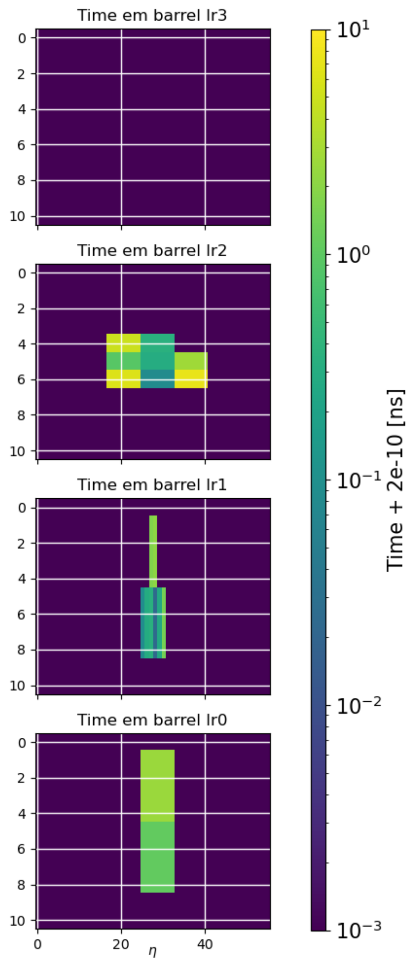


Figure 3.8: An example of absolute time images from ECAL in MC for $Z \to ee$ after the up-sampling. The illustrated time images are from the same event as figure 3.6. Note that the color-map is in logimatic scale and the unit is *ns*.

[3] Curvature is the gradient of the gradient and measures how much the gradient changes at a given point.

[4] The QuantileTransformer is mostly used to handle outliers/long tails in a distribution at the cost of changing the linearity between distributions.

3.3.

$$r_{truth} = \frac{E_{acc}}{E_{truth}}. \quad \text{The transformation apply: } E_{acc} \cdot \hat{r} = \hat{y}, \tag{3.3}$$

where $\hat{r}$ is the output from *DeepCalo* and $\hat{y}$ is the final prediction used in $\mathcal{L}(y, \hat{y})$. This transformation will result in a decrease in convergence time and might also lead to a small accuracy improvement due to an increase in numerical stability.

Mislabeled events in MC

Mislabeled events can confuse a CNN and might increase convergence time and decrease performance, so they will be removed. From inspecting the MC sample in table 3.1, we found two types of discrepancies between Ecalib$^{(BDT)}$and E_{truth}. The first type of discrepancy is a mislabeling, where the particle pair both have the same E_{truth}. These pairs are easy to remove, as we just need to check if the tag and probe share E_{truth} and, if they do, the pair is removed.

The second one is slightly more complex. For some events in MC, we see a large difference between E_{truth} and Ecalib$^{(BDT)}$(or E_{acc}). E_{acc} is assumed to be close to E_{truth} for all events. This is likely due to a mismatch between the SuperCluster and track/vertex of the particle (see section 1.3.3). However, it is difficult to determine if it is a poor reconstruction from Ecalib$^{(BDT)}$or a mismatch of the cluster. Nevertheless, equation 3.4 will be applied, where k is a hyperparameter that can vary as it is unknown when the discrepancy starts. The mislabeled and selection of events at various k values are illustrated in figure 3.9 with table 3.6 showing the ratio of events left after the cut.

Cuts (k)	Event ratios after cut
0.6	0.61
0.4	0.58
0.2	0.54
0.1	0.48

Table 3.6: The table shows the different k cuts from figure 3.9 and the ratio of events left. Selecting $k = 0.6$, 60% of the sample remains.

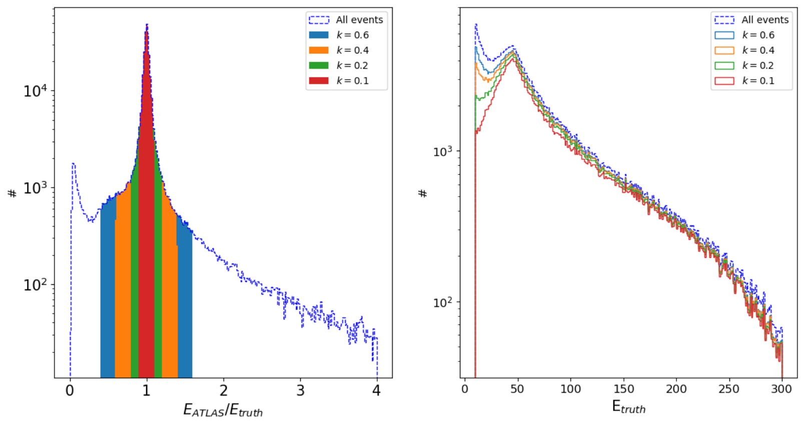


Figure 3.9: *Left figure* shows the distribution of Ecalib$^{(BDT)}/E_{truth}$ with four different values of k applied to display the selection of events. Events close to 1 indicate Ecalib$^{(BDT)}\approx$ E_{truth}. The *right figure* illustrates the distribution of E_{truth}. The distribution suggests events are uniformly removed, except for low E_{truth} values, which means many mislabeled event have a low E_{truth}. The table 3.6, we can see the ratios of events left after cutting.

$$\text{keep\_event}: \quad \left| 1 - \frac{\text{Ecalib}^{(BDT)}}{E_{\text{truth}}} \right| < k \tag{3.4}$$

where k can be seen as a hyperparameter determining different levels of strictness. Too strict of a k will result in *DeepCalo* not being trained on events that the Ecalib$^{(BDT)}$has found difficult to reconstruct, which will result in a performance decrease for *DeepCalo*.

However, it will improve convergence time as mislabeled events are removed. Looking at figure 3.9, $k = 0.6$ is selected. $k = 0.6$ will select all events within the peak centered at one, remove the peak at ≈ 0.1 and the long-tailed values > 1.6. This will select events within $0.4 < Ecalib^{(BDT)}/E_{truth} < 1.6$. In the analysis chapter, the use of k will also help the $Ecalib^{(BDT)}$in MC in comparison with the *DeepCalo*, as some poorly reconstructed events will be removed.

3.5 *Selecting candidates from Data*

Selecting events from Data can be difficult. A clean sample is desired, and when performance is measured using the BW⊗CB, the particles must be associated with the correct decay channel. Data will also be used in training, and here a clean sample is also obligatory because E_{truth} in Data is constructed using the associated decay channel. The selection in Data can be seen in table 3.7. These are the requirement for the particles in $Z \to ee$ and $Z \to \mu\mu\gamma$.

$Z \to \mu\mu\gamma$		$Z \to ee$
$\mu\mu$	γ	ee
> 9.5GeV	> 9.5GeV	> 9.5GeV
Loose	Loose	Loose
$\sum Q = 0$	• $N_\gamma = 1$	$\sum Q = 0$
• $N_{\mu\mu} = 1$	Tight	• $N_{ee} = 1$
Trig	• $N_\gamma = 1$	**Event dropped**
• $N_{\mu\mu} = 1$	**Event dropped**	
$m_{\mu\mu} < 82$ GeV		
$m_{\mu\mu} > 20$ GeV		
• $N_{\mu\mu} = 1$		
Loose vs. Tight		
• $N_{\mu\mu} = 1$		
Event dropped		

Table 3.7: This table shows the selection of events in Data for the different decay channels. • indicates a check (or if statement), and if the statement is satisfied, the remaining particles are selected. However, if not, additional cuts below are needed. If the last • is not satisfied, the event is dropped. The $Z \to \mu\mu\gamma$ cuts are connected, as the channel requires two muons and a single photon.

4 Model Architecture

In the following chapter, we will introduce the concept of *High-Performance Computing* (HPC) and the basics behind computers. We will not elaborate on the fundamentals of computers, such as transistors and machine code, but limit the introduction to the interact between components within a computer, and focus on optimizing the run-time of ML models. As such, this chapter might serve as an introduction for people venturing into Deep Learning. For further reading on computing fundamentals, see [15] and [48]. Afterwards, the performance measures and loss functions for MC and Data in relation to *DeepCalo* will be introduced. Lastly, an in-depth description of the *DeepCalo* architecture will be given, where the sub-modules and optimization of *DeepCalo* will be presented. Much of this work is done with [18] as its foundation.

4.1 High-performance Computing

With increasing complexity and data availability, the data size and convergence time of models will skyrocket, making a tedious process of training and optimizing the hyperparameters of the models. However, with advanced algorithms and hardware, the issue can be solved. In the following paragraphs, some modern techniques to optimize algorithms and hardware will be explored, focusing mostly on the graphic processing unit (GPU) due to its parallel computing capabilities.
For this thesis, only NVIDIA GPUs were available, so only NVIDIA GPUs and their software will be discussed. Although for many years, the majority of computational speed-ups have been through cramming more transistors into the hardware following Moore's law [31]. Nowadays, the focus has shifted to software optimization of the hardware algorithms in the attempt to cut corners in the communication between different components. While speeding up an algorithm has always been limited to hardware in some capacity, it is important to have equilibrium between hardware so the computing devices are constantly fully utilized. Otherwise, performance can be left on the table. When a computing device is limited by another device in the system, it is referred to as a *bottleneck*.

Hardware

The core computing components of a system are the CPUs and GPUs[1].

[1] While most motherboard can have multiple GPUs installed for a single CPU, some enterprise grade motherboard can have multiple CPUs installed as well.

While other components will affect the performance, none of them are computing units.

The CPU is the brain of the operation as it communicates with all the components connected to the motherboard. CPUs have a varying number of cores and vary in frequency from $\approx$ 1.0-5.0 GHz[2]. The CPU cores have a large amount of internal memory that is storing information about the application at hand. The CPU used in this thesis is a Threadripper 3970x with 32 cores (64 threads) at 3.7GHz base clock (w/ boost at 4.5 GHz).

[2] Hz meaning operation per second. CPUs also have a base clock and a boost clock, being able to boost to a higher clock before getting too hot.

CPUs process the advanced commands coming from the user and can in short bursts run single applications very fast due to its high operation frequency but lack performance in heavy parallel processes. An important characteristic for the CPU is its memory management as most communication goes through the CPU[3]. It has an advanced Memory Controller Chip (MCC) that is optimized for handling the multiple levels of CPU memory . Embedded directly into the CPU is its cache or SRAM (with 3 different types of memory L1, L2, L3), which has a size of several MBs but is extremely fast and insures that the CPU does not run out of applications to run internally[4]. Bandwidth[5] of cache can vary dependent on CPU, however, a common theoretical bandwidth of L3 cache is 175 GB/s. The next level of memory is DRAM (dynamic random-access memory or RAM), which is slower than the cache but has dynamic storage and is module-based, such that additional memory can be installed into the motherboard as long as the CPU supports it. RAM contains information about the task at hand or previously run tasks. RAM speed is measured in MHz and Latency. Without going into too much detail about RAM specifications, MHz control how quickly data can be read or written to the RAM and Latency is a measure for the delay between request and execution of a command, and should be minimized. RAM at 3200 MHz has a transfer rate of 25.6 GB/s. This will limit the complete bandwidth of the system, as RAM controls the transfer rate to the CPU. If a CPU has four memory channels the total bandwidth of the CPU is $4 \cdot 25.6\,\text{GB/s} = 102.4$ GB/s of bandwidth to other components in the system. CPU manufacturers will usually indicate the number of memory channels to indicate the bandwidth of a CPU, as this is very important if many components have to be connected to the motherboard. This can also be seen as a PCIe bandwidth. Examples will be given below.

[3] New technologies bypass the CPU. They will be explained below as they will be a game-changer for the *HPC*.

[4] The amount of cache will increase as the number of cores increase, as the embedded memory is required to an increasing number of tasks for the increasing number of cores.

[5] Rate at which the CPU can read and write data back and forth to components.

Place	Name	Memory Size
Chip	Core	$0.256 - 16$ kB
Chip	$\text{L}1 - L3$	$2 - 256$ MB
Mo	Ram	< 2 TB
Mo	HDD (swap)	Limit to Mo

Table 4.1: The table lists the place and size of the different memory options. Mo refers to the motherboard.

The last memory level is the storage using the *NVMe* or *SATA* [48] technologies, which can store large amounts of data in the petabyte size. However, storage is very slow compared to other memory types and is severely limited by the CPUs PCIe bandwidth and its own read or write speed. Some of the fastest *NVMe* available a have transfer speed of 2GB/s. However, the CPU has a bigger, clumsier brother that is dependent on the CPU, namely the GPU. It runs slower than the CPU, with frequencies between 1-2 GHz, has less embedded memory and less advanced cores compared to those of the CPU and has non-swappable RAM. Nonetheless, what it does not have in speed and complexity, it makes up in raw horse-power. NVIDIA GPUs have in the neighborhood of 2.000-10.000 CUDA cores that are optimized for parallel computing[6]. Think of the CPU as being a speedboat that quickly can get from A to B but carrying only a few, whereas the GPU is a cruise ship that is slow from A to B but can carry thousands. The cruise ship, however, is dependent on the speedboat to scout ahead. The GPU has its own RAM called VRAM that is non-swappable. It is not directly connected to the storage or RAM of the PC, so data has to flow between the CPU and GPU. The GPU and CPU are connected using a PCIe port, which has a transfer speed[7] of 32 GiB/s one-way. A schematic of the GPUs cores and memory size compared to the CPU can be seen in figure 4.1. Here, it is visible that the CPU contains large amount of internal memory compared to the GPU. Due this amount of internal memory, the CPU is used for running applications, as it is able to cache much of the information from programs so it can quickly accessed again, where the GPU has considerably more, less advanced cores, designed for parallel tasks.

[6] GPU cores are not the same as CPU cores, but when it comes to simple linear algebra in ML, they behave the same. GPUs also have Tensor Cores, which are vectorized cores that able to compute faster linearly with the drawback of reducing precision.

[7] 1000^3 bytes is 1 GB, whereas 1024^3 bytes is 1 GiB.

One last important thing is regarding compatible hardware. With the new and improved PCIe 4.0, transfer speeds have increased from 16GiB/s to 32GiB/s one-way for PCIe, meaning transfer speeds between storage[8]-CPU and CPU-GPU has double the bandwidth. However, both CPU, GPU, motherboard and storage have to be compatible with PCIe 4.0, otherwise the hardware will run at PCIe 3.0 speeds.

[8] Using PCIe 4.0.

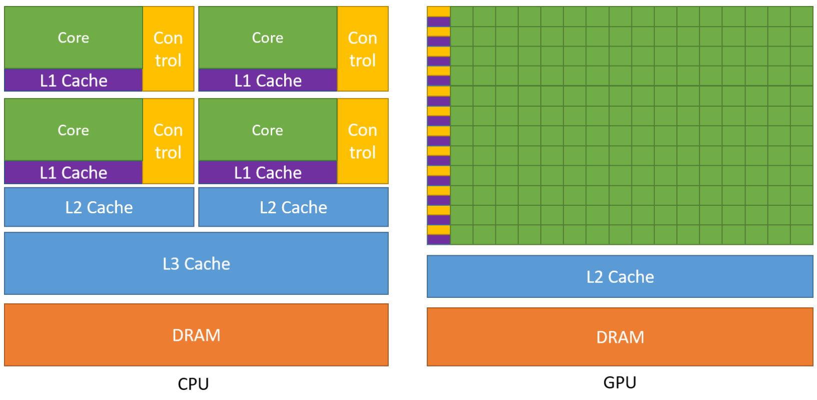


Figure 4.1: From [32]. The schematic shows the memory and core levels of the CPU and GPU. The elements are color-coded so the L1 cache, core and control can also be seen on the GPU figure. Storage could also be a part of the figure, but it is linguistically usually not referred to as memory.

Bottlenecks

If the speedboat is not fast enough, the cruise ship will have to wait. This results in the GPU waiting for the CPU to load/transfer the data, resulting in an idle GPU, and convergence time will increase. A CPU has a maximum memory bandwidth, which is the maximum amount of data a CPU can transfer. The CPU used in this thesis has a maximum memory bandwidth of 95.37 GiB/s (or 102.4 GB/s) and PCIe 4.0 can transfer 32 GiB/s one-way, meaning if two GPUs with PCIe 4.0 are connected to the CPU[9] and both could be fully utilized

[9] Which is the case.

in both directions[10], the CPU will not be able to transmit enough data to the GPUs to be fully utilized, leading to a bottleneck. Additional import of data from storage device also requires bandwidth, making the system even more crowded.
To identify a bottleneck when training a ML algorithm, one would need to run one model where the data is cached in RAM and one without[11]. This will result in three scenarios.

1. If the GPU utilization in both cases are close to 100%, there is no bottleneck.
2. If the GPU utilization is high, when data is cached, but otherwise not. The bottleneck between CPU/RAM and GPU.
3. If the GPU utilization is always low. The bottleneck between RAM and GPU.

[10] $2 \cdot 2 \cdot 32$ GiB/s = 128 GiB/s.

[11] To see utilization (see the amount of watts drawn) of the GPU run `watch -n 0,001 nvidia-smi`.

Solutions

Some possible solutions will be given below. In scenario 1, there is no bottleneck in the system. If an increase is desired, it can be obtained by investing in new hardware that will increase the computing power of the system. In scenario 2, the bottleneck is likely an I/O[12] or pipeline problem. Both problems can be solved by parallelizing the data transfer and pipeline.
In scenario 3, the bandwidth between the GPU and CPU is filled[13]. Bandwidth cannot simply be increased[14], so the only solution is to minimize the bandwidth of other components or the bandwidth of the data. Decreasing bandwidth used by the input data will improve convergence time in all three scenarios, and could be accomplished by reducing the precision from `float64` to `float32`[15], which would take up less bandwidth and increase data flow. The data format could also be changed to binary, which could drastically reduce loading time. The ML software *Tensorflow* has its own data format for binary called *TFRecords*[45], which decreases loading times and memory footprint. Depending on the file, changing the format to binary might also decrease the file size.

[12] Input/Output and refers to communication with external storage.

[13] It is also possible to change the data format of input data. This will be explained below.

[14] Faster RAM could be bought to increase bandwidth. However, it is limited to the memory bandwidth of the CPU.

[15] `float64` has a memory size of 64 bits with a total number of combinations at 2^{64}, whereas `float32` uses 32 bits and `float16` uses 16 bits.

One solution also just comes down to system architecture. Due to the swappable modular design of the X86 architecture[16], most information has to flow through the CPU, and this highway through the CPU can be crowded, forcing components to wait. However, newer CPU architecture, such as *ARM*, where the GPU is embedded into the CPU and where they share RAM, removes the overhead between the CPU and GPU and speeds up the process[17].

[16] The most common CPU architecture.

[17] This is the CPU architecture of the new Apple computer, it is called an M1 chip.

Software

Newer software also occasionally decreases computation time. Where the solutions above reduce transfer time between the storage and GPU, a software update could increase the computation speed of the algorithm, optimizing the calculation within a compute unit. A useful tool to classify bottlenecks in a pipeline is *Tensorboard* [44], which is a software developed by *TensorFlow* that is able to benchmark ML architectures and their pipelines to find possible improvements. *Tensorboard* can also be used to display hyper-parameter optimization, and this part of the software will be used in the optimization (see section 4.4). Software is also being developed to increase computation speed and minimize bandwidth usage by optimizing the computation in the cores and connection between components in the system. *Tensorflow*, in collaboration with Nvidia and Google, have developed XLA [46] [18] and Mixed precision [47] [19], which can greatly decrease the computation time of models, but in the Mixed precision case, the precision of the model will also decrease slightly. We will not go in-depth with their inner workings as it is beyond the scope of this thesis. NVIDIA has also announced *DirectStorage* for computer games, software to bypass the CPU and feed data directly into the GPU to minimize latency and bandwidth problems. This could remove any bandwidth in HPC and decrease computation time by removing the CPU from the equation.

[18] It is an acronym for Accelerated Linear Algebra and pre-compiles the computational code by fusing mathematical expressions together.

[19] Mixed precision algorithm will utilize the Tensor Cores for increased speed. However, Tensor Cores compute with a lower precision (`float16`) compared to cuda cores, but the algorithm will switch between cuda and Tensor Cores for the most optimal computation speed and model precision.

Lastly, one could prune the architecture of a NN after x-number of iterations, removing nodes with low weights. However, this will most likely decrease the accuracy of the network depending on how rough the pruning is.

4.2 *Optimizing networks*

Now that the fundamentals of HPC have be established, we will pivot back to the reconstruction of particles and the *DeepCalo* architecture. The following section will introduce the loss function used to train *DeepCalo* and the evaluation metrics used to compare the performance between *DeepCalo* and $E\text{calib}^{(BDT)}$.

Loss in Monte Carlo

Choosing the correct loss function is a crucial step in ML as it governs the optimization and behavior of the algorithms. In this thesis, the chosen loss function is the logarithm of the hyperbolic cosine (`logcosh`) seen in equation 4.1 and figure 4.2.

$$\mathcal{L}(y, \hat{y}) = log(cosh(y - \hat{y})). \tag{4.1}$$

`logcosh` combines two of the most popular loss functions, `MAE` and `MSE`. The disadvantage of the `MAE` is its fork function gradient[20]

[20] Note that the calculations of the gradient of the `MAE` and `MSE` are not completely accurate, as we disregard the sum and $1/n$. However, the behavior of the gradient is the same, and that is the focus.

$$\text{MAE}'(y, \hat{y}) = \begin{cases} 1 & \text{if } y \neq \hat{y} \\ 0 & \text{if } y = \hat{y} \end{cases}.$$

This means that all predictions will receive the same gradient, except for $y = \hat{y}$ (very rare in regression), the network otherwise will not be rewarded for correctness of its predictions. MSE solves this by squaring the error, so the gradient evolves like

$$\text{MSE}'(y, \hat{y}) = 2(y - \hat{y}).$$

For almost correct predictions, the gradient becomes very small, however, as the loss scales squarely, the outliers will receive a very high gradient that will overshadow others.

logcosh solves these disadvantages by combining MAE and MSE. At low differences, the distribution behaves like MSE. However, at larger differences the logcosh behaves like MAE, see figure 4.2.

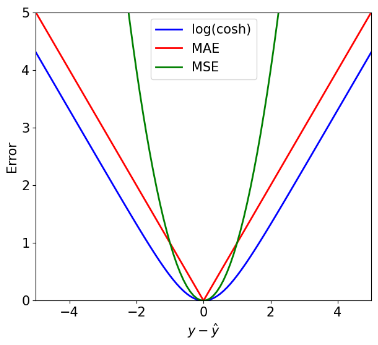


Figure 4.2: The figure illustrates the behavior of three possible loss functions MAE, MSE and logcosh. The x-axis gives the error between $\hat{y}$ and y.

Data

Optimizing the network for Data is even more complicated due to the missing *truth* label. However, a loss function and setup of the network can be constructed, unlocking the ability to train on Data. We have developed two methods for training on Data with a focus on $Z \to ee$[21]. This will be a brief in introduction into the methods, but a more thorough introduction will be given in chapter 5.

[21] It is also possible to work in the $H \to \gamma\gamma$ and $Z \to \mu\mu\gamma$ channels. However, the $Z \to \mu\mu\gamma$ Data sample is not large enough for convergence, and the $H \to \gamma\gamma$ Data does not contain the ECAL image.

Regression to the mean

We create a loss function that calculates the invariant mass of the electron-pair using the *DeepCalo* predictions. The invariant mass is then compared with the Z-boson mass at 91.1876 GeV. The logcosh is implemented to compare the predicted invariant mass to the determined mass of the Z-boson (see equation 4.2).

$$\mathcal{L}(\hat{y}) = log(cosh(91.19^2 - \hat{M}_{ee}^2)) \tag{4.2}$$

Only events between $86\text{GeV} \leq M_{ee} \leq 97\text{GeV}$ have been selected using the $Ecalib^{(BDT)}$, so this loss function has a minor dependence on the ATLAS reconstruction.

Constructing E_{truth} from E_{ATLAS}

The other method to train on Data is by constructing an approximation of E_{truth} from $Ecalib^{(BDT)}$ for Data called $E_{label,data}$. Using equation 1.1, $E_{T,1}$ is isolated and M is set to $M = 91.19$ GeV. Then, a single electron in the electron pair is selected, and $Ecalib^{(BDT)}$ is inserted as $E_{T,2}$. The result is the $E_{T,1}$ of the electron if the invariant mass of the event was exactly 91.19 GeV. Recall that the invariant mass of $Z \to ee$ is a BW distribution, so using $M_Z = 91.19$ as a constant mass is not correct and is an approximation. However, by selecting events within $86\text{GeV} \leq M_{ee} \leq 97\text{GeV}$, we believe the approximation is sufficient enough to use as E_{truth} in training. This method also has a dependency on the ATLAS reconstruction.

4.2.1 *Uncertainty on predictions*

For more details on aleatoric and epistemic uncertainties in ML, we highly recommend [24], which the following section is based on.

Uncertainties are an important part of every estimation. In ML, we distinguish between two types of uncertainties, an aleatoric uncertainty[22] and epistemic uncertainty[23]. Epistemic uncertainties rise from the lack of knowledge in an experiment and are often not present in large-scale data analysis due to the amount of data. Therefore, we will focus on aleatoric uncertainties. A simple change in the architecture gives *DeepCalo* the ability to predict the energy and its aleatoric uncertainty using the negative log-likelihood as a loss function.

[22] Also known as statistical uncertainty, which is the noise on the observed sample.

[23] Also known as systematic uncertainty, which captures the ignorance of the model based on the observations fed to the model. Epistemic uncertainty can also be referred to as model uncertainty.

$$\mathcal{L}(y,\hat{y},\sigma) = \frac{1}{2}\left(\frac{(y-\hat{y})^2}{\sigma^2} + \log(\sigma^2)\right) \tag{4.3}$$

The loss function consists of two parts, a likelihood term relating the predicted energy, truth energy, and σ together. The second is a regularization term on the uncertainty to guarantee that $\sigma \to \infty$. However, a problem arises from equation 4.3, as the loss is undefined at $\sigma = 0$ and *DeepCalo* uses ReLU$= max(0,x)$ as its last activation function, making $\sigma = 0$ possible as an output. There are two solutions to this problem. The first is selecting an activation function that is > 0 everywhere. This could be Softplus/SmoothReLU $= ln(1+e^x)$ bound within $]0,\infty[$, however, it might cause numerical instabilities due to underflow when using `float32/64`. The second solution is rewriting equation 4.3 by defining $\sigma^2 = e^{\sigma'}$

$$\mathcal{L}(y,\hat{y},\sigma) = \frac{1}{2}\left(\frac{(y-\hat{y})^2}{e^{\sigma'}} + \log(e^{\sigma'})\right) \Rightarrow \frac{(y-\hat{y})^2}{2e^{\sigma'}} + \frac{1}{2}\sigma'. \tag{4.4}$$

DeepCalo will then predict y and σ', and if $\sigma' = 0$, the loss will reduce to `MSE`.

It should be noted that due to the transformation, the absolute size of σ should not directly be related to the uncertainty of the energy prediction, as it is possible to have[24] $\sigma' = 0$. It should be seen as a confidence indicator and is used as a measure of how confident the network is in its prediction. However, due to its nature, we will refer to it as a confidence or uncertainty.

[24] Due to the transformation of σ, we have a minimum bound on $\sigma = \sqrt{e^{\sigma'}}$, $\sqrt{e^0} = 1$.

Nonetheless, σ can be carried over to the invariant mass of the parent particle, estimating a uncertainty on its invariant mass. The equation to calculate the invariant mass of a two particle decay can be seen in section 9.1.2. This uncertainty can in turn be used to remove poorly reconstructed events.

Metric	Expression
MAE	$\frac{1}{n}\sum_{i=0}^{n}\lvert y_i - \hat{y}_i\rvert$
MSE	$\frac{1}{n}\sum_{i=0}^{n}(y_i - \hat{y}_i)^2$
RMSE	$\sqrt{\frac{1}{n}\sum_{i=0}^{n}(y_i - \hat{y}_i)^2}$

Table 4.2: The table shows possible performance metrics. MAE gives equal weight to every prediction. MSE will focus on outliers, which will weigh heavily in the average. RMSE also weights outliers heavily, but attempts to make the error more relatable similar to MAE by applying the square root.

4.2.2 *Performance measure*

The main purpose of this thesis is to outperform the Ecalib$^{(BDT)}$. Therefore, it is necessary to construct metrics that fairly evaluate *DeepCalo* against Ecalib$^{(BDT)}$. In the following section, we will focus on constructing the evaluation methods for MC and Data that fairly measures the performance between Ecalib$^{(BDT)}$and *DeepCalo*.

Monte Carlo

In MC, the evaluation between Ecalib$^{(BDT)}$and *DeepCalo* is quite simple. As it is simulated data, the E_{truth} is known and can be evaluated directly against the predictions. This provides an opportunity to use numerous supervised metrics to evaluate performance. Some examples can be seen in table 4.2. The metric used is this thesis takes inspiration from [11], where the relative error (RE) is calculated[25]

$$RE = \frac{E_{calib}}{E_{truth}}, \tag{4.5}$$

[25] In textbooks, the relative error is calculated by $\frac{measure-truth}{truth} = \frac{measure}{truth} - 1$. However, in this thesis, RE will be referring to equation 4.5, as the only difference to the textbook version is a shift.

where E_{calib} can be either *DeepCalo* or Ecalib$^{(BDT)}$ predictions. Afterwards, the effect interquartile range (eIQR) is used to measure the narrowness of the RE,

$$eIQR = \frac{P_{75}(RE) - P_{25}}{1.349}(RE), \tag{4.6}$$

where P_{75} and P_{25} are the upper and lower quantiles. The 1.349 is a normalization term, as 50% of the area under a Gaussian distribution lies 1.349 from the mean. The RE will create a Gaussian distribution centered at 1, where the aim is to reduce the width of the distribution, which is measured by eIQR. Performance between models can then be compared using equation 4.7.

$$\text{Relative metric} = 1 - \frac{\text{model metric}}{\text{benchmark metric}} \tag{4.7}$$

This is the measure[26] that will be used in chapter 5 and 6, as well as for finding the optimal architecture in section 4.4. For MC, eIQR will be used as a metric, where the ratio between eIQRs (reIQR) will be used to compare models $re\text{IQR}_n = 1 - \frac{e\text{IQR}_n^{\text{model}}}{e\text{IQR}_n^{\text{benchmark}}}$.

[26] This measure is actually not fair, but rather biased towards the *benchmark metric*, eg. $eIQR_{model} = 1$ and $eIQR_{benchmark} = 2$. Using equation 4.7, we see $1 - 1/2 = 0.5$ on an increase of 50%. However, if we change the fraction $2/1 - 1 = 1$, we see correctly improvement of 100% increase in performance. However, equation 4.7 was chosen as it was used in [18], so performance can be compared.

Data

Due to the missing E_{truth} for Data, the evaluation of model performance is not as easily obtainable as for MC. However, due to the assumption of particle resonance explained in chapter 1, we can estimate the performance from the width of the invariant mass distribution by using a fit. The fit will have a small variation dependent on the decay channel. These will be described below.

- $Z \to ee$: The resonance distribution will be fitted using a Breit–Wigner distribution convoluted with a Crystal Ball function (BW⊗CB), where the parameters of the BW are set to $\mu_{BW} = 91.1876$ GeV and $\sigma_{BW} = 2.4952$ GeV (see [43]). The parameters of the CB are then fitted. The measure of performance will be the additional width from the CB (σ_{CB}). The CB should measure the error arising from the energy reconstruction.

- $Z \to \mu\mu\gamma$: For $Z \to \mu\mu\gamma$ it is almost the same as $Z \to ee$. However, due to the number of background events, the fit function required an additional term for the background distribution. The background function used is a wide Gaussian that is fitted outside the Z peak.

- $H \to \gamma\gamma$: We do not have $H \to \gamma\gamma$ for Data. Thus, metrics for evaluating Data are not needed. However, some fitting methods have been tested on the invariant mass distribution of the Higgs for MC. The one that achieved the best results was a double Gaussian (Note that 125 GeV is the mass of the Higgs [43]).

$$G(f_g, G) = (1 - f_g) \cdot G(\mu = 125, \sigma = 3) + f_g \cdot G(\mu, \sigma) \tag{4.8}$$

These fitting methods will be used as the benchmark for Data. As a proof of concept, figure 9.1 illustrates the invariant mass distribution of $Z \to ee$ using all the MC *truth* variables. The distribution has then been fitted using a BW and a BW⊗CB. It can be seen that the BW alone is not able to represent the distribution accurately, but when the BW⊗CB fit is applied, the $\sigma_{CB} \approxeq 0$, as expected when the E_{truth} is used.

4.3 *Model components*

This section contains a detailed description of the *DeepCalo* architecture as well as the optimization applied to achieve the most accurate model. A flowchart overview of *DeepCalo* and its sub-models can be

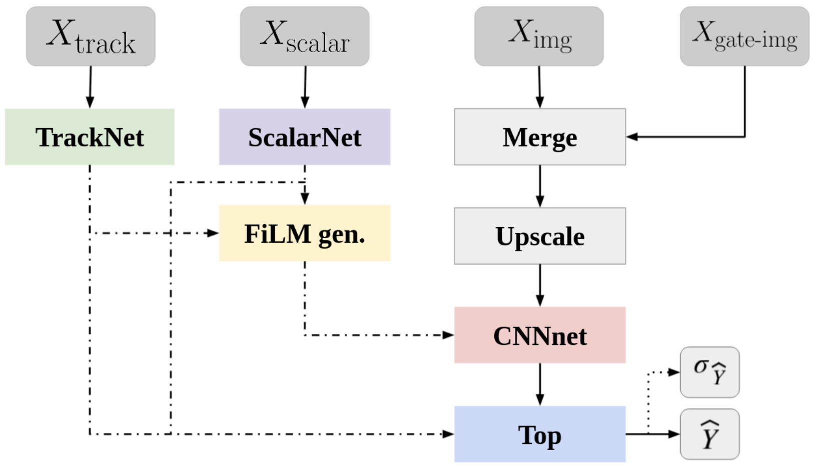


Figure 4.3: The figure illustrates the *DeepCalo* architecture, where colored boxes are modules and the gray ones are input. The dashed lines indicate the possible connections to the main network, namely *CNNnet* and *Top* layers. The network outputs $\hat{Y}$, but can also output the uncertainty $\sigma_{\hat{Y}}$. Images are also up-sampled in *DeepCalo* (by Upscale) to save on file size of the data and *Merge* is used to concatenate the Gate and ECAl images.

seen in figure 4.3. The figure must be read from top to bottom with the arrows indicating data flow. At the top of the chart, the potential inputs are listed: X_{track} are the track variables (see table 3.4), X_{scalar} are the scalar variables (see table 3.3), X_{img} is the ECAL image (see figure 3.5) and, lastly, the $X_{gate-img}$ can be additional information in image format, which could be time, gain or noise. The main modules of the network are *CNNnet* and *Top*, which will be a part of every iteration of the network. The network has three additional sub-modules, namely the *TrackNet*, *ScalarNet*, and *FiLM gen*. The connection of modules can be seen in figure 4.3, where the dash-dotted lines are connections that can be deactivated. As a rule of thumb, the architecture predicts the energy related to the input data.However, we will show examples where the architecture predicts the uncertainty associated with its energy prediction, as well as classifying between MC and Data events.

The gray *Merge* box in figure 4.3 will concatenate X_{img} of size $56 \times 11 \times 4$ with $X_{gate-img}$ of size $56 \times 11 \times 4 \cdot d$, where N is the number of gate variables, eq. time, gain and/or noise. This will result in an image of size $56 \times 11 \times 4 + 4 \cdot d$. Afterwards, the images are upsampled in the *Upscale* box to a squared resolution of $56 \times 55 \times 4 + 4 \cdot d$. The colored sub-modules will be explained in detail in the following sections.

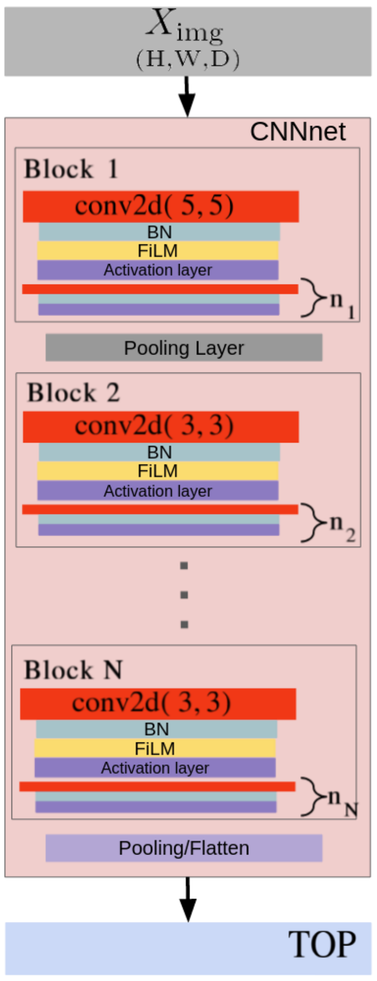


Figure 4.4: The figure shows the internal layout of the *CNNnet*.

CNNnet

The central module of the architecture is the *CNNnet*. The general setup is inspired by [18]. As input it takes images of size $(H, W, D) = (56, 11, D)$, where the depth, D, is dependent on the amount of features[27] added to the images. The images first encounters a conv2d used to deduct the relation between pixels in the images. The architecture of the *CNNnet* can be seen in figure 4.4. The *CNNnet* is designed in CNN Blocks ranging from $i \in \{1, ..., N\}$ (i being the Block number), where each Block contains a *conv2d* of 3×3 or 5×5, a *batch normalization* (BN) layer, a *FiLM* layer, and an *activation* layer, followed by $n_N \times$sub-blocks consisting of *conv2d*, BN and activation layers. Located between each Block is a *pooling* layer set to 2, which reduces the images in each pooling layer by $(H/(2i), W/(2i))$. Each *conv2d* outputs $2^{i-1}d_0$, where d_0 is the depth of X_{img}. After N Blocks, the *CNNnet* final layer is a *GlobalPooling* or *Flatting* layer reducing the output to a vector, ready for the *Top* module. The *CNNnet* architecture was tested in [18], where numerous variations of many architectures were tested on MC. This will not take place in this thesis due to the time constraints. However, optimization of the layers within the Blocks will take place.

[27] By features, we mean energy, time, gain. Each feature has $D = 4$, one for each layer in the ECAL.

ScalarNet

ScalarNet is a simple neural network. It functions as an *upscaler* for the scalar variables before they go into the *FiLM* or *Top*. It is similar to a Dense Block, as shown in figure 4.7, and outputs the number of variables equal to the number of neurons in the last layer.

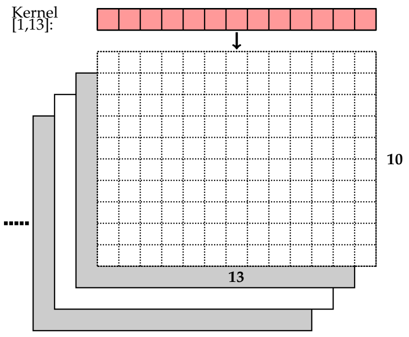


Figure 4.5: The figure shows *conv1d* performed on the sequence data from the tracks. The kernel size and number of filters can vary depending on preference. In the figure, the number of track variables is 13, however, in table 3.4 there are only 11. The number of tracks in the figure is 10, however, the number of tracks selected in each event will be determined in the *TrackNet* section.

TrackNet

X_{track} can be a strong indicator of the pileup present in the ECAL images, as the track particles can deposit their energy within the ECAL images of the particle in question. Therefore, it is important to take this behavior into consideration. Figure 4.9 illustrates the number of tracks per event. The track variables can be seen in table 3.4. Handling tracks can pose a challenge as a single event has multiple tracks associated, with each track having 11 variables recorded (called sequence data). A standard forward-feeding network is not able to handle this format of data, but convolutional neural networks and recurrent neural networks are. To be consistent throughout the network, a 1D convolutional layer (*conv1d*) is selected to handle the

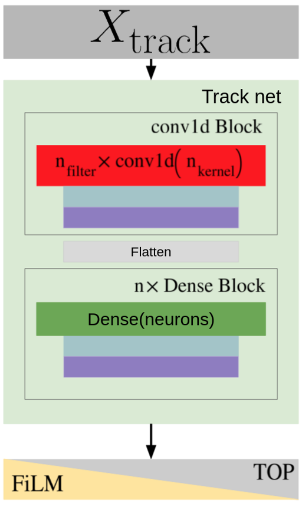


Figure 4.6: The figure shows the internal layout of *TrackNet*.

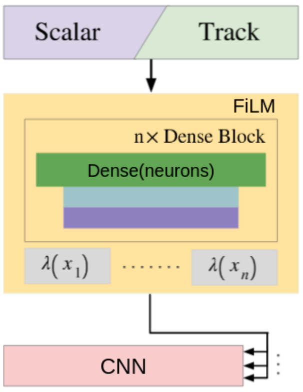


Figure 4.7: The figure shows the internal layout of the *FiLM gen.*. The *FiLM gen.* outputs a vector λ with scale and shift parameters for the channels in the *CNNnet*.

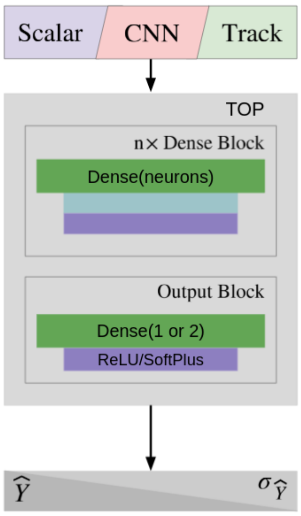


Figure 4.8: The figure illustrates the architecture of the *Top* module. It is a simple NN with a single or double output.

sequence data. The *TrackNet* architecture is illustrated in figure 4.6. It consists of a single *conv1d* block with *kernel size* and *# of filters* as hyper-parameters. Afterwards, a *Flatting* layer creates a vector as input to n number of Dense Blocks. The output from *TrackNet* serves as input to *FiLM* and/or *Top* modules.

FiLM

The feature-wise linear modulation (FiLM) [17] sub-module can be seen in figure 4.7. It consists of n Dense Blocks, where the number of neurons can vary. The output of the Dense Blocks are n feature-wise affine transformations $\lambda(x_n)$ corresponding to the number of *FiLM* layer in *CNNnet* (see figure 4.4). The expression for $\lambda(x_n)$ can be seen in section 2.3.2. Intuitively, the *FiLM gen.* must be able to re-weight channels it deems to be pileup (see figure 2.9) and provide a more accurate prediction of E_{truth} by removing pileup contributions.

The output is dependent on the Block number and kernel sizes of the *CNNnet*. As previously mentioned in the description for *CNNnet*, the ith Block outputs a new depth of $2^{i-1}d_0$, and, since the *FiLM* applies linear transforms on each feature map (channel) using γ and β, it outputs

$$|\lambda_n| = 2 \cdot d_0 \left(\sum_{i=1}^{N} 2^{i-1} \right) = 2 \cdot d_0 \left(2^N - 1 \right) \tag{4.9}$$

number of parameters, where λ_n is the vector used by the nth *FiLM* layers in the *CNNnet*. It consists of parameters, namely γs and βs, which are used for scaling and shifting the image channels. The $2 \cdot d_0$ is the amount of γs and βs needed at each convolution to scale and shift the channel, and $\sum_{i=1}^{N} 2^{i-1}$ is the evolution in the number of channels after a convolution. The *FiLM* optimization took place in [18], and will therefore not be touched on in this thesis.

Top

The last sub-module is the *Top*, which is a part of the main *DeepCalo* network. Here, the information from the previous modules is collected and used as input for the Dense *Top*. The *Top* architecture can be seen in figure 4.8. As *TrackNet* and *FiLM*, it consists of Dense Blocks with the output Block consisting of a single dense layer with either one or two neurons using either ReLU or SoftPlus as an activation function. With this final layer, the network can predict positive real numbers $\mathbb{R}$, and this could be the energy of a particle and its uncertainty (for two output neurons). The output format will be explained in section 4.2.

4.4 *Optimization*

Due to the size of the network, each sub-module is optimized individually by testing their performance in predicting E_{truth} for $Z \rightarrow ee$

MC events. Lastly, we will optimize the connections between the modules. Possible connections can be seen in figure 4.3. *Hparams* and *Tensorboard* will be used for the hyperparameter optimization and displaying of model performance.

TrackNet

TrackNet will be optimized alone. It will predict an energy by itself and the best performing model will be connected to *DeepCalo*. The amount of tracks will vary between events, but when using *conv1d*, the size of inputs must be the same. In figure 4.9, the number of tracks per event is shown (on $Z \rightarrow ee$). It is similar to a Poisson distribution with $\lambda \approx 6$. Selecting 15 tracks per event is approximately $\approx 98\%$ of the tracks and should be a suitable number of tracks indicating pileup. For events not containing 15 tracks, zero-padding was applied. However, some events have more than 15 tracks, so the most important tracks must be selected. Here, the tracks will be sorted by equation 3.1 and the top 15 will be selected. Afterwards, the architecture of *TrackNet* must be optimized. *Conv1d*, kernel size and number of filters will be optimized. To reduce the number of permutations, the kernel size and number of filters will be set to the same value $[1, ..., 5]$. The number and size of the Dense Blocks can also be optimized, and various sizes will be tested. A GridSearch[28] will be applied, using *logcosh* (see equation 4.1) to evaluate performance. The performance of *TrackNet* can be seen in figure 4.10, where the lime highlighted line is the best performing architecture.

After the optimization of *TrackNet*, we will test the influence of the variables available to the *TrackNet* (see table 3.4). To measure the importance, we will use *Permutation Feature Importance*[29]. The results are illustrated in figure 4.11. We see that each variable contributes positively to the performance of *TrackNet* as the performance decreases if any variables are shuffled. We also see a large variance of η and θ, which is likely due to η being a transformation of θ, so they contain the same information and the model switches between using one or the other.

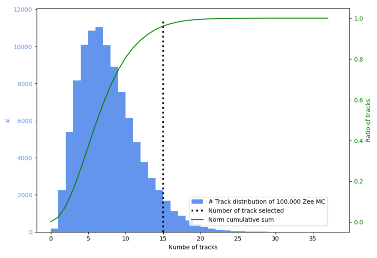


Figure 4.9: The blue histogram shows the number of tracks in 100.000 $Z \rightarrow ee$ events. The green line is the CDF of the histogram and the black vertical line indicates the number of tracks that are selected for *DeepCalo*. $\approx 98\%$ of present tracks are selected.

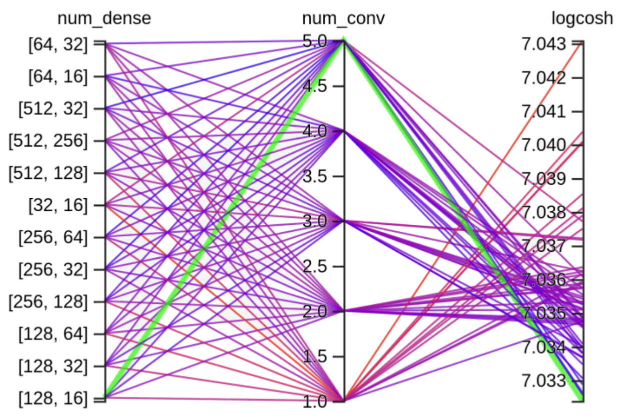


Figure 4.10: Tensorflows *HParams* has been applied to find the most optimal *TrackNet*. Each iteration has been trained five times with random initial weights. The lime colored line shows the most optimal architecture. *Tensorboard* is used to display the image. *num_dense* is the number and size of the Dense block and follows 2^i, eg. $[64, 16]$ means 3 hidden layers with 64, 32 and 16 neurons in each layer, respectively. The number of filters and kernel sizes are combined in one hyper-parameter called *num_conv*.

[28] GridSearch will test all possible permutations of the network one by one to find the most optimal one.

[29] At the beginning of each model, a feature is randomly shuffled to break down the relationship between the feature and target.

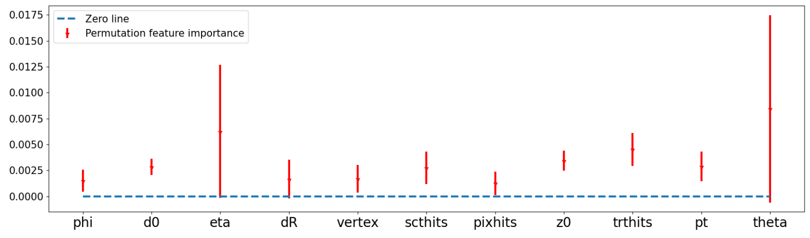


Figure 4.11: Permutation Feature Importance has been run five times for each variable giving uncertainties. If a feature is not contributing to the performance of the network, it should be negative.

Hyperparameter	Space
Number of block	**3**, 4, 5
Size of blocks	2, **4**, 6
Pooling layer	**maxpool**, avgpool
Global Pooling	**maxpool**, avgpool, flatten

Table 4.3: The hyperparameters chosen in the GridSearch of *CNNnet*. The parameters in bold are the parameters of the most accurate model.

CNNnet

Optimizing the *CNNnet* is challenging, as *conv2d* contains many hyperparameters. Selecting the most meaningful ones, the evaluation time can be reduced, yet still take multiple weeks.

Using figure 4.4 as a starting point, the hyperparameters chosen to optimize for can be seen in table 4.3, and the result from the GridSearch can be seen in figure 4.12 with the most accurate model colored lime. The *CNNnet*s output will serve as an input to the *Top*, where the *Top* consists of 3 hidden layers with $[512, 512, 1]$ neurons in each layer.

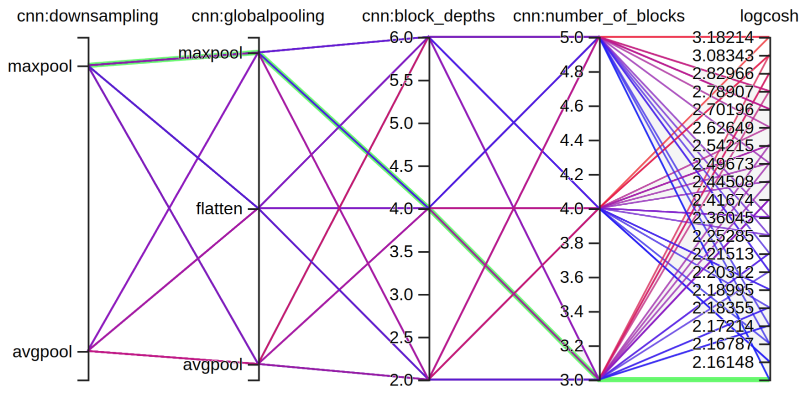


Figure 4.12: Tensorflows *HParams* have been applied to obtain the most optimal *CNNnet*. All samples were loaded into memory to minimize latency, however, due to memory constraints, the complete set of data was not used. The sample sizes were $S_{train} = 2.000.000$, $S_{val} = 750.000$ and $S_{test} = 500.000$, with early-stopping at 15 epochs. *Logcosh* is the performance measure, and the lime-colored line shows the best-performing one.

Each iteration in figure 4.12 was run 4 times to ensure some convergence stability, which resulted in a running time of about five weeks on a single NVIDIA RTX 3090. The lime-colored was the best performing model and these parameters are then selected.

ScalarNet, FiLM gen. and Top

Due to time constraints, optimization of *ScalarNet*, *FiLM gen.* and *Top* were not possible. However, all sub-modules were optimized in [18], so the parameters found there will be used in this thesis.

4.4.1 *Selecting sub-modules*

We have two modules that can change position, namely *ScalarNet* and *TrackNet*, and they can be connected to *FiLM gen.* and *Top*, resulting in a total of 12 permutations[30] with an additional for the basic network (just *CNNnet* and *Top*). The optimal model will be selected using MC

[30] Mistakenly, there are actually 15 permutations, but three were *lost in translation*.

and Data evaluation explained in section 4.2.2, where the Data metric will be weighted highest as its performance in Data, we wish to improve[31]. Each permutation will be run twice to ensure similar convergence. The results can be seen in table 4.4. The row marked

[31] Only a small sample of Data has been used to test the accuracy.

	reIQR75	reIQR95	$\sigma_{CB,mc}$	$\sigma_{CB,data}$
Basic	-0.121	-0.025	2.677	2.488
FiLM: scalar	0.229	0.257	1.871	2.108
FiLM: scalar - top: scalar	0.229	0.252	1.844	2.132
FiLM: scalar - top: scalar track	0.223	0.251	1.850	2.107
FiLM: scalar - top: track	**0.226**	**0.264**	**1.794**	**1.969**
FiLM: scalar track	0.228	0.265	1.810	2.119
FiLM: scalar track - top: scalar track	0.210	0.262	1.779	2.035
FiLM: track - top: scalar	-0.042	-0.067	2.517	2.297
FiLM: track - top: track	0.140	0.149	2.057	2.204
top: scalar	-0.154	-0.131	2.327	2.377
top: scalar track	0.213	0.233	1.924	2.158
top: track	0.136	0.164	2.051	2.172

Table 4.4: Performance of the various sub-module combinations. Each model has been trained twice on MC $Z \rightarrow ee$. Three permutations are missing, and we unfortunately did not have time to test them.

as bold is the best performing one, with *ScalarNet* in *FiLM gen.* and *TrackNet* in the *Top*. Table 4.4 also provides us a small fraction of insight into the BlackBox of *DeepCalo* by indicating the importance of the variables and sub-modules of the network. First of, adding *ScalarNet* into the *Top* does not affect performance much, but it is certainly very important in the *FiLM gen.*. This may be due to *ScalarNet* containing spacial and general energy information of the cells, and this will help the *FiLM* re-weight the channels. The *TrackNet* contains information about the possible noise in the image and could also work well for the *FiLM gen.*. However, this is likely used as a re-weighting parameter in the *Top*.

4.4.2 *Learning Rate Scheduler*

LRS

After the *DeepCalo* architecture has been selected, it is time to look at the hyperparameters that are not a part of the architecture, namely learning rate and batch size. Choosing the correct learning rate becomes less crucial when applying an LRS, however, the correct LRS, as well as its behavior and range, still needs to be chosen. From figure 2.12 and [18], we see that oneCLR provides the desired speed-up compared to the CLR with the same accuracy level. oneCLR has also been used in the optimization of the network above due to the speed-up, so we will continue to use oneCLR.

Next, it is important to choose the correct range for oneCLR. This can be tested by varying the learning rate during training, see figure 4.13. The selection of λ_{max} and λ_{min} is in the regions where the learning flattens. In figure 4.13, the black dashed lines indicate the regions of λ_{max} and λ_{min} resulting in $\lambda = [0.0001, 0.04]$ for *DeepCalo*.

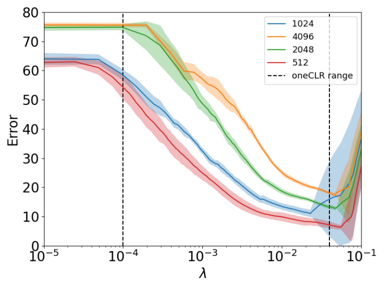


Figure 4.13: Determining the oneCLR learning rate range. The *DeepCalo* model has been trained for one epoch five times to measure the mean and standard deviation of the loss. The learning rate has then been gradually changed during the epoch to measure learning rate at which the network learns the most.

Batch-size

Due to the memory size of the *NVIDIA* RTX 3090 (24 GB), the batch-size for the input data can be quite large, ranging from [512, 4096] in input size[32]. In figure 4.13, the convergence of *DeepCalo* at different batch-sizes is shown. The batch-size desired is the one with the steepest slope when changing the learning rate. It can be seen in figure 4.13 that the batch-size of 2048 has the steepest decline in the loss, so it is chosen as the batch-size.

[32] Batch-size could be lower, but this would result in poor generalization.

DeepCalo summary

A quick summary of the parameters found from the optimization above and in [18] is given table 4.5. This architecture has a total of 2.202.797 parameters[33]. Before hyper-parameter optimization, the *DeepCalo*s total number of parameters was ≈ 3.600.000. The reduction in parameters is appreciated and attributes to a small speed-up in training.

[33] The parameters consists of 2.194.851 trainable parameters and 7.946 non-trainable.

Table 4.5: The table shows a summary of the important parameters selected for *DeepCalo*. Note that the activation function used throughout the network is *Leaky ReLU*, except for the output layer that uses *ReLU*.

Hyperparameter	Parameter
`TrackNet`	
Units	(128, 64, 32,16)
Normalization	Batch
Kernel size & filters	5
Connected to	[Top]
`ScalarNet`	
Units	(256)
Normalization	Batch
Connected to	[FiLM]
`FiLM gen.`	
Units	(512, 1024)
Normalization	Batch
`CNNnet`	
Down-sampling	MaxPool
Globalpooling	MaxPool
Number of blocks	3
Depth of blocks	4
`Top`	
Units	(512, 512, 1)
Output activation	ReLU

Analysis

The following two chapters will explore the performance of the *DeepCalo* model discussed in chapter 4. *DeepCalo* will be trained and evaluated on four separate channels. All channels have MC samples available, but only $Z \rightarrow ee$ and $Z \rightarrow \mu\mu\gamma$ have samples of Data (see table 4.6). To keep track of the different *DeepCalo* models, we will add a subscript to each model. The subscript will indicate the channel and the type of data the model is trained on, eg. $DeepCalo_{Zee,mc}$ is the $Z \rightarrow ee$ model trained on MC samples.

In the evaluation of the models, multiple reconstruction properties have to be investigated before a conclusion can be drawn. We will have to investigate

- the general performance increase for both MC and Data,
- the uniform performance increase over the entire $|\eta|$, $\langle\mu\rangle$ and E_T range.

All the *DeepCalo* models will be evaluated on these properties. Unless stated otherwise, every model has been run four times to ensure measure consistency and achieve an uncertainty on the performance.

Channel	MC	Data
$Z \rightarrow ee$	1.000.000	450.000
$Z \rightarrow \mu\mu\gamma$	350.000	400.000
$H \rightarrow \gamma\gamma$	310.000	No data available
Electron Gun	1.100.000	No data available

Table 4.6: The table displays the channels that will be explored and the number of events in the *test* sample for both MC and Data.

5 Electron Reconstruction

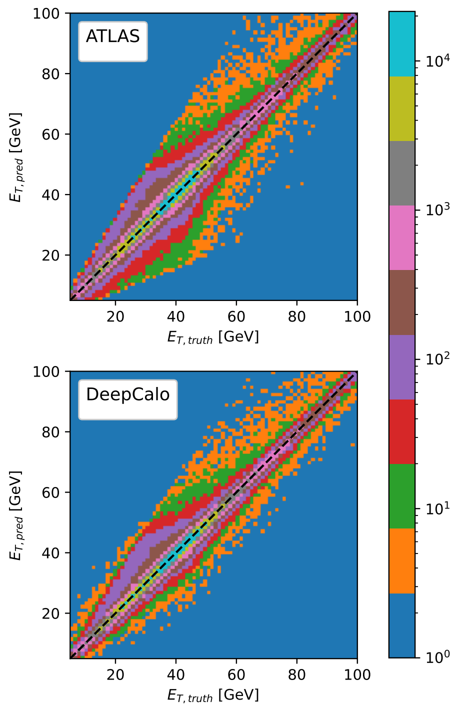


Figure 5.1: Predictions are on MC. The figure shows a correlation plot between $E_{T,truth}$ and transverse predictions from Ecalib$^{(BDT)}$or *DeepCalo*$_{Zee,mc}$. The black dashed line is a linear line ($f(x) = x$). The closer to the black line the better. The color indicates the number of events and the color-scale is logarithmic. We see *DeepCalo* being placed closer to the dashed line than Ecalib$^{(BDT)}$.

In this chapter, we will explore the electron reconstruction performance of *DeepCalo* in the $Z \to ee$ channel. The chapter will contain five *DeepCalo* models. First a MC trained model called *DeepCalo*$_{Zee,mc}$, where performance will be measured for both MC and Data. Afterwards, *DeepCalo*$_{Zee,data}$, a Data trained model, will be evaluated. Next is the ensemble model *DeepCalo*$_{Zee,mc/data}$, where the two previous models have been combined. This model is trained on both MC and Data. Afterwards, the *DeepCalo*$_{EG}$ model will be evaluated. EG stands for *Electron Gun* and is an electron sample that has no decay channel but a large E_T range. The last section will be on the *additional properties of DeepCalo*, where a *DeepCalo*$_\sigma$ model will be assessed, as it is able to estimate uncertainties on its predictions, thus removing poorly reconstructed events.

5.1 Analysis of *DeepCalo*$_{Zee,mc}$

In the following section, the performance of the *DeepCalo* model trained for $Z \to ee$on MC, *DeepCalo*$_{Zee,mc}$, will be explored.

5.1.1 Monte Carlo

The *DeepCalo*$_{Zee,mc}$ performance in MC can easily be evaluated, as we are able to directly compare its predictions with the E_{truth}. The evaluation methods for both MC and Data are explained in section 4.2.2. The MC samples will also be evaluated using the evaluation method applied to Data so a direct comparison between MC and Data performance is possible.

Energy evaluation

We compare *DeepCalo*$_{Zee,mc}$ and Ecalib$^{(BDT)}$to E_{truth} using the *e*IQR and *re*IQR metrics described in section 4.2.2. The *re*IQR results and their uncertainties can be seen in equation 5.1.

$$\begin{aligned} \langle reIQR_{75}^{DeepCalo} \rangle &= 22.4 \pm 0.7\% \\ \langle reIQR_{95}^{DeepCalo} \rangle &= 25.9 \pm 0.9\% \end{aligned} \tag{5.1}$$

From equation 5.1, we see a performance gain of $> 20\%$ and consistency in the convergence, as the uncertainties are low at $\approx 1\%$. An example of the RE distribution of a *DeepCalo*$_{Zee,mc}$ model and the

$Ecalib^{(BDT)}$reconstruction can be seen in figure 5.3. The $DeepCalo_{Zee,mc}$ distribution in the figure is clearly more narrow than that of $Ecalib^{(BDT)}$, indicating an improvement in reconstruction. To ensure unbiased and pattern-free predictions, the correlation between E_{pred} and E_{truth} can be seen in figure 5 for both *DeepCalo* and $Ecalib^{(BDT)}$. This will be referred to as a correlation plot. The correlation plot for the $Ecalib^{(BDT)}$ prediction reveals no biases, as most events follow the black dashed line. However, looking at the outliers, they have an upper and a lower bound (cone-like shape) with a slight tendency towards underestimating at low energies. This tendency, however, is diminished at $E_T > 50$ GeV. This cone-like shape will be discussed in chapter 7. Looking at the correlation plot of $DeepCalo_{Zee,mc}$, we see a considerable improvement in removing the underestimated outliers compared with $Ecalib^{(BDT)}$. The overestimated outliers do not look quite as dense as for $Ecalib^{(BDT)}$, however, it is difficult to determine from the figure. In chapter 7, we will inspect the behavior of the outliers. As mentioned in the introduction to this chapter, the aim is a uniform performance increase independent of $|\eta|$, $\langle\mu\rangle$ and E_T. By separating the events into their respective $|\eta|$ vs E_T and $\langle\mu\rangle$ bins, we can evaluate the performance of each bin.

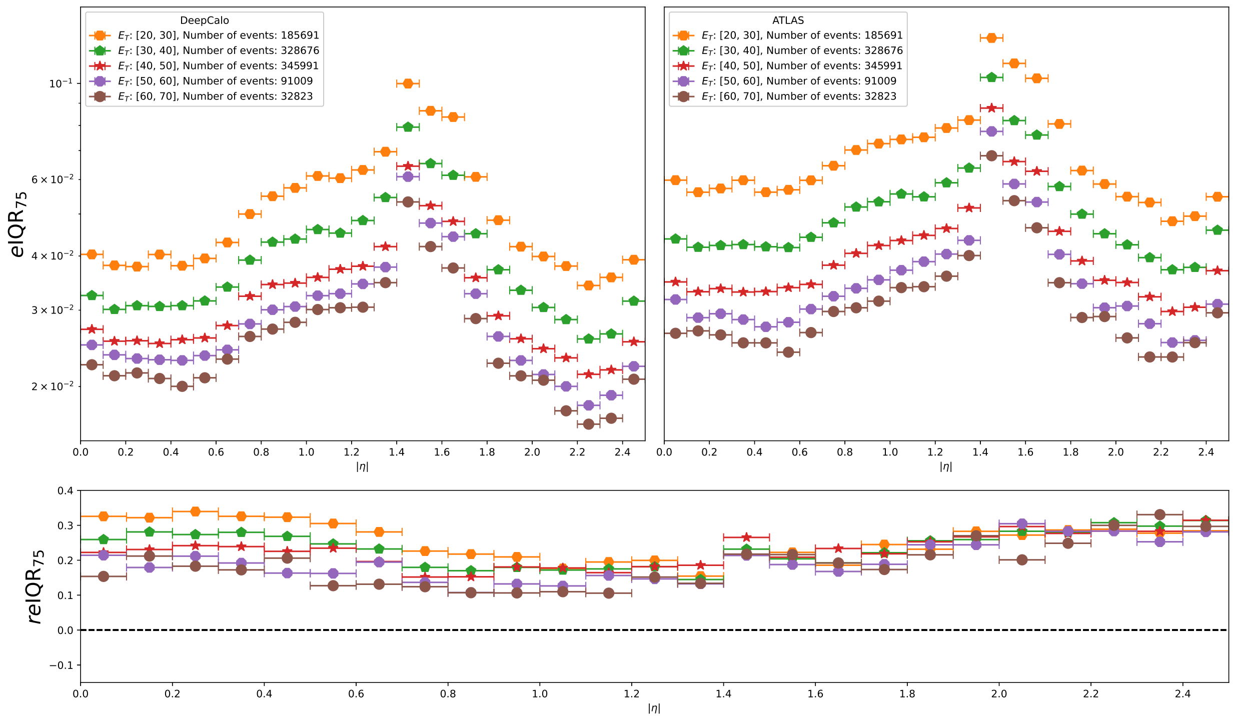


Figure 5.2: Predictions are on MC. *Top figures* shows the $eIQR_{75}$ of RE as a function of $|\eta|$ and E, with bin size of 0.1 in $|\eta|$ for $Z \rightarrow ee$. They share a *y axis* and the legend shows the number of events in each E_T bin. *Bottom figure* show a comparison between the two model using $re\text{IQR}_{75}$ for all the $|\eta|$ and E_T bins.

The $|\eta|$ vs E_T performance can be seen in figure 5.2 using the $e\text{IQR}_{75}$ measure and comparing the models using $re\text{IQR}_{75}$. We see consistent 10-30% performance improvements in the $|\eta|$ vs E_T bins. For more detailed figures on the $|\eta|$ vs E_T performance, see section 9.5.4, where the number of events in each bins are also shown. The $\langle\mu\rangle$ performance on MC will follow shortly.

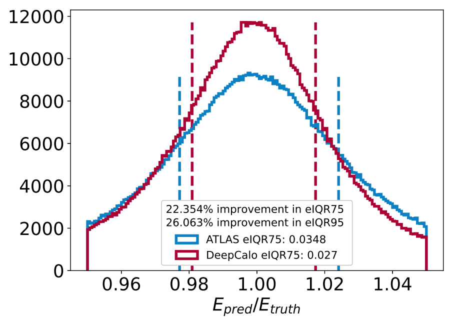
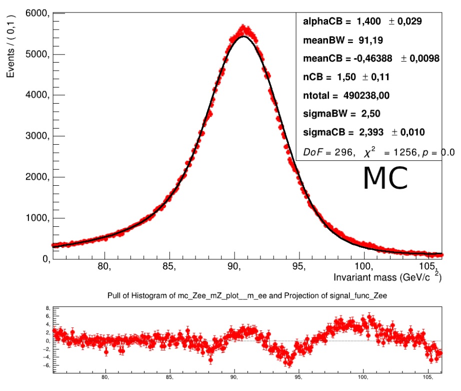
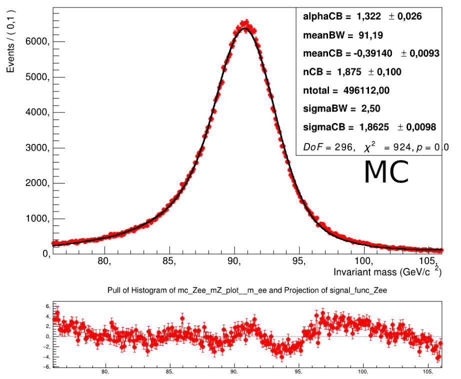


Figure 5.3: Predictions are on MC. The figure shows the RE between E_{truth}, $Ecalib^{(BDT)}$and $DeepCalo_{Zee,mc}$. The vertical dashed lines are the IQR_{75} and IQR_{25} of the distributions with the $re\text{IQR}_{75}$ and $re\text{IQR}_{95}$ performance in the legend text.



(a) Energy reconstruction from Ecalib$^{(BDT)}$



(b) Energy reconstruction from $DeepCalo_{Zee,mc}$

Figure 5.4: The figure shows the BW⊗CB fit to the energy estimations from $DeepCalo_{Zee,mc}$ and Ecalib$^{(BDT)}$in MC. Only true electrons have been used in the fit. The best performing model has the lowest σ_{CB}.

Mass evaluation

The $>20\%$ performance increase seems promising for the the *DeepCalo* Data performance. However, due to the discrepancies between MC and Data, the performance in Data is expected to decrease. By using the Data metric explained in section 4.2.2, we will be able to compare the performance between $DeepCalo_{Zee,mc}$ on MC and Data. An example of the invariant mass fit to MC can be seen in figure 5.4 with the result from the four models compared to Ecalib$^{(BDT)}$in equation 5.2. The expressions used to calculate the average and the uncertainties can be seen in chapter 9.

$$\langle 1-\frac{\sigma_{CB}^{DeepCalo}}{\sigma_{CB}^{ATLAS}}\rangle = 1-\frac{1.8310\pm 0.006}{2.393\pm 0.01} = 23.5\pm 0.4\% \tag{5.2}$$

We see a $23.5\pm 0.4\%$ improvement in the width of the BW⊗CBfit over Ecalib$^{(BDT)}$, which translates to a more accurate energy reconstruction.
To see direct comparisons between the invariant masses of $DeepCalo_{Zee,mc}$, Ecalib$^{(BDT)}$and E_{truth}, see figure 9.6, where the decrease in width is more prominent.

5.1.2 Data

As performance has been measured on MC, we will now evaluate it in Data. The $DeepCalo_{Zee,mc}$ model provides a significant improvement over Ecalib$^{(BDT)}$ in energy reconstruction for MC. However, as shown (See section 3.3) and discussed earlier, there are notable discrepancies between MC and Data that will lead to a decrease in *DeepCalo* performance. In $Z\rightarrow ee$ Data, the performance metric will always be σ_{CB} of the BW⊗CB fit.

Mass evaluation

The BW⊗CB fit for one of the four models can be seen in figure 5.6, where the selection criteria from table 3.7 have been applied. The performance on Data of $DeepCalo_{Zee,mc}$ compared to Ecalib$^{(BDT)}$can be seen in equation 5.3.

$$\langle 1-\frac{\sigma_{CB}^{DeepCalo}}{\sigma_{CB}^{ATLAS}}\rangle = 1-\frac{2.058\pm 0.010}{2.271\pm 0.019} = 9.4\pm 0.9\%. \tag{5.3}$$

While we see an improvement in the Data performance compared to Ecalib$^{(BDT)}$, it is a significant decrease from the previous MC performance. The $|\eta|$ performance of the $DeepCalo_{Zee,mc}$ model will follow in section 5.6, where we compute the resolution of $|\eta|$ in both MC and Data.

Pileup performance for MC and Data

With the upcoming upgrades due at the LHC, an increase in luminosity and thereby pileup is expected. An analysis of the pileup

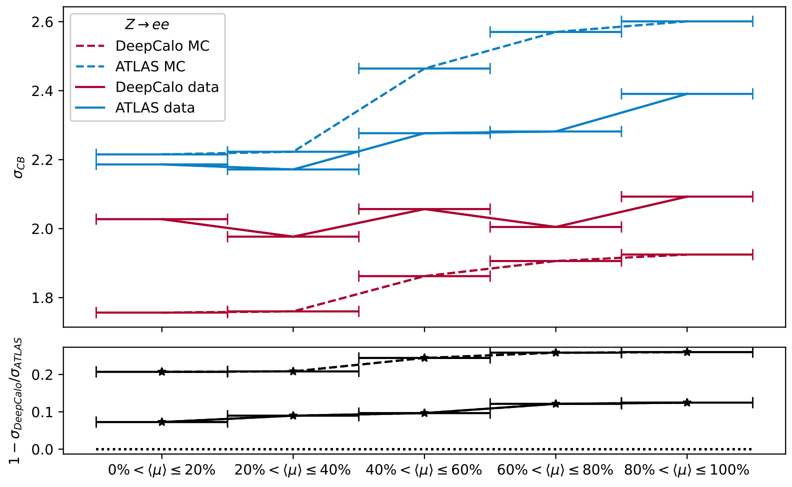
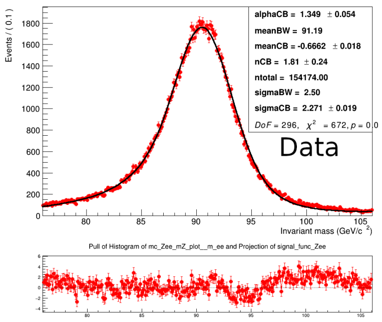


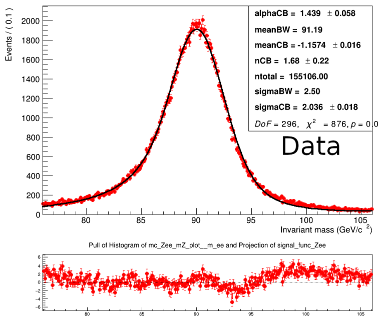
Figure 5.5: *Top figure* shows the pileup ($\langle\mu\rangle$) performance of $DeepCalo_{Zee,mc}$ for both MC (dashed line) and Data (solid line). The pileup range between MC and Data are not the same, so both ranges have been divided into five equally sized bins. The performance measure is σ_{CB} from the BW⊗CBfit. *Bottom figure* shows the comparison between *DeepCalo* and Ecalib$^{(BDT)}$ using $1 - \sigma_{DeepCalo}/\sigma_{ATLAS}$. The pileup range for MC is $[10, 80]$ and for Data it is $[15, 40]$.

performance of *DeepCalo* is important, as pileup can challenge the reconstruction. Therefore, it is important that *DeepCalo* achieves a better energy reconstruction than Ecalib$^{(BDT)}$at high pileup, as it is the future of LHC. It should be noted that pileup for MC is not re-weighted, as the files could not be found. Thus, the pileup range between MC and Data are not comparable, as the pileup range differs between MC and Data. The pileup ranges can be seen in figure 3.3.

The pileup performance for both MC and Data can be seen in figure 5.5 with the $1 - \frac{\sigma_{DeepCalo}}{\sigma_{ATLAS}}$ at the bottom. The figure shows that Ecalib$^{(BDT)}$and $DeepCalo_{Zee,mc}$ both suffer from increasing pileup in MC, but from the bottom plot we see that Ecalib$^{(BDT)}$ suffers more from pileup than *DeepCalo* as the *re*IQR increases. For Data, we see the same behavior, with the *DeepCalo* performance remaining fairly constant, while the performance of Ecalib$^{(BDT)}$ decreases with increasing pileup. Looking at the $1 - \frac{\sigma_{DeepCalo}}{\sigma_{ATLAS}}$ ratios, *DeepCalo* achieves an improvement of 20% and $\approx$ 9% on MC and Data, respectively, when compared to Ecalib$^{(BDT)}$ at low pileup. With increasing pileup, the ratio increases to $\approx$ 28% and $\approx$ 12% for MC and Data, respectively, which looks promising for when LHC undergoes upgrades.



(a) Energy reconstruction from Ecalib$^{(BDT)}$



(b) Energy reconstruction from $DeepCalo_{Zee,mc}$

Figure 5.6: The figure show the BW⊗CB fit to the energy estimations from $DeepCalo_{Zee,mc}$ and Ecalib$^{(BDT)}$on Data. The criteria from table 3.7 have been used to select events in Data.

5.2 *Electron gun*

From the analysis above, we saw improvements both in MC and Data throughout pileup over the entire $|\eta|$ range and energies within $E_T \approx [10, 70]$.

However, electrons from $Z \rightarrow ee$ only have a limited energy range, and performance needs to be tested at higher energies. This is essential, as it might be required for the discovery of new physics at higher energies. The EG MC samples will be used for testing this[1], as they have electrons at very high energies (see figure 9.11). In addition, the EG evaulation will also examine the bias of the*DeepCalo* towards

[1] The EG samples contain roughly 7.500.000 events with $0.6, 0.15, 0.25$ ratios for train, test and validation samples, respectively.

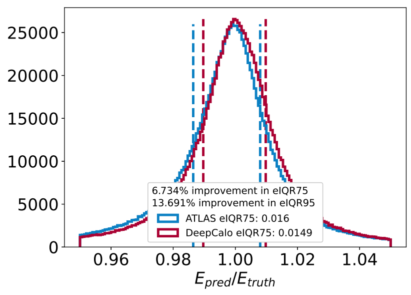


Figure 5.7: Predictions are on MC. The histogram shows the RE of *DeepCalo*$_{EG}$ and Ecalib$^{(BDT)}$ with the *e*IQR improvement to *DeepCalo* in the legend. The vertical dashed line are the IQR$_{75}$ and IQR$_{25}$ of both distributions.

decay channels, as the algorithm might be biased for a specific invariant mass. The *DeepCalo*$_{EG}$ has only been run twice. An example of the RE distribution of one of the *DeepCalo*$_{EG}$ models can be seen in figure 5.2 and the general performance can be seen in equation 5.4.

$$\begin{aligned} \langle reIQR_{75}^{DeepCalo} \rangle &= 6.811 \pm 0.077\% \\ \langle reIQR_{95}^{DeepCalo} \rangle &= 13.81 \pm 0.12\% \end{aligned} \tag{5.4}$$

We see a $6.811 \pm 0.077\%$ and $13.81 \pm 0.12\%$ improvement in *e*IQR$_{75}$ and *e*IQR$_{95}$, respectively. This is a decrease compared with the previous improvement of $> 20\%$. This is likely due to the decrease in the concentration of events, as the energies span $E_T = [5\text{->}\ 1000$ GeV, with roughly 4.5 millions events for training. In figure 5.8, the η vs E_T bins are illustrated (similar to figure 5.2).
Here, we see fluctuations between the increasing and decreasing of performance compared with Ecalib$^{(BDT)}$. *DeepCalo* especially struggles at very high energies. However, in and beyond the crack region ($|\eta| > 1.4$), *DeepCalo* exceeds Ecalib$^{(BDT)}$consistently, except at very high energies.
Nevertheless, the performance of *DeepCalo* on a single electron is a bit disappointing, as we had hoped for better generalization at high energies. Yet, the performance might increase with large MC samples.

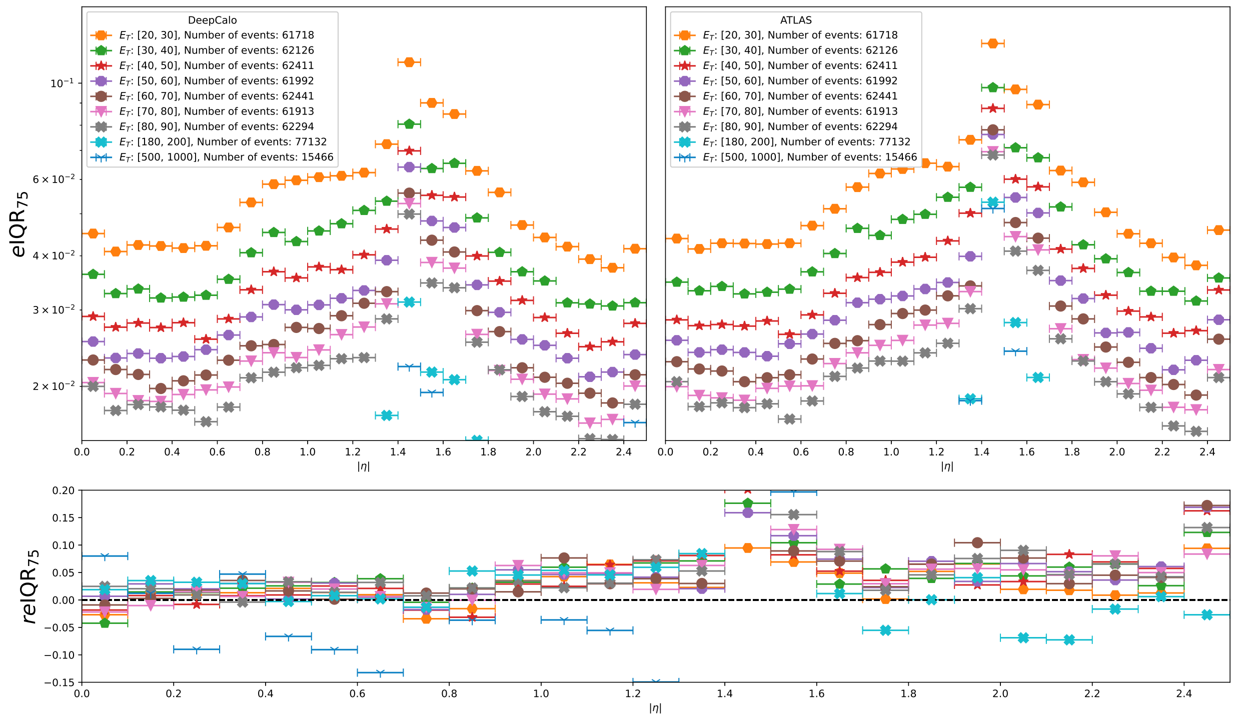


Figure 5.8: Predictions are on MC. *Top figures* show the *eIQR*$_{75}$ of RE as a function of $|\eta|$ and E_T with a bin size of 0.1 in $|\eta|$ for the EG samples. They share a *y axis* and the legend shows the number of events in each E_T bin. *Bottom figure* shows a comparison between the two models using *re*IQR$_{75}$ for all the $|\eta|$ and E_T bins.

5.3 Analysis of $DeepCalo_{Zee,data}$

Discrepancies between MC and Data are mostly accounted for when looking at the 1D and 2D variable distributions, yet higher order discrepancies in $N > 2$ variable space are not corrected and are prominent when comparing $DeepCalo_{Zee,mc}$ performance for MC and Data. Variations in high dimensions are nearly impossible to account for. However, we might be able to help *DeepCalo* understand these discrepancies by training directly on Data. As was briefly explained in chapter 4, we have developed two methods for training on Data, which will be explained in greater detail in the coming sections. Both methods involve $E\text{calib}^{(BDT)}$, as only events within 86 GeV$s \leq M_{ee} \leq$ 97 GeV are selected, with $E\text{calib}^{(BDT)}$ being used for calculating M_{ee}.

Regression to the mean

The first method we tested was constructing the invariant mass of the particle pairs in the loss function and evaluating it to the invariant mass of the parent particle. The functions used in the loss function can be seen in equation 5.5.

$$\begin{aligned} M^2 &= 2p_{T,1}p_{T,2}(\cosh(\eta_1 - \eta_2) - \cos(\phi_1 - \phi_2)) \\ \mathcal{L}(\hat{y}) &= \log(\cosh(91.19^2 - M^2)) \quad \text{for } Z \to ee. \end{aligned} \tag{5.5}$$

However, this loss function has multiple drawbacks. It reduces the number of evaluations by half, and it can be a problem if too few events are within 86 GeV$s \leq M \leq$ 97 GeV. η, ϕ and $EventNumber$ are required to calculate the invariant mass, so they have to be added in the output layer. While not a problem for η and ϕ, this poses a problem for *DeepCalo*, as it outputs `float32`, leading to the $EventNumber$ overflowing, as it is `int64` precision. This can be solved by sorting the events beforehand, not shuffling during training, and only using a single GPU so the data is separated. The loss function was not vectorized either, making it very slow. Lastly, the method is dependent on the batch sizes to correct for the approximation[2] and achieve a meaningful gradient from the loss function. Here, a large batch size is required, meaning a large amount of memory on the GPU is necessary. Due to these drawbacks, this methods will not be used in the further analysis. The next method, however, will be.

[2] The approximation referring to the assumption that the invariant mass is not a distribution but a fixed mass.

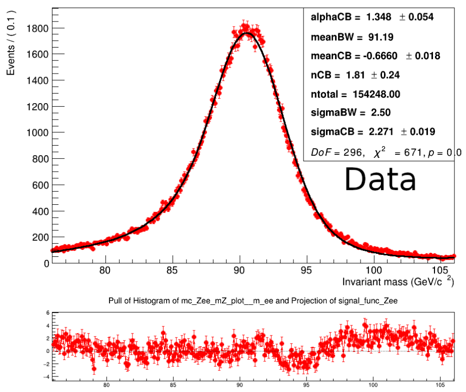
Constructing $E_{label,data}$

This method will be applied when training directly on Data. This method involves constructing an estimate of $E_{truth,data}$ from $E\text{calib}^{(BDT)}$. Thus, no change in loss function is required, resulting in none of the drawbacks previously mentioned. However, it would still be a approximation. Using equation 5.6, we can construct $E_{truth,data}$ using

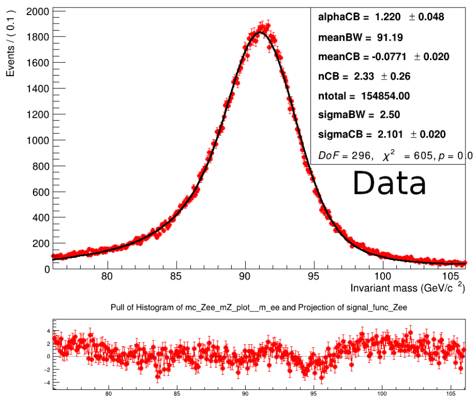
Ecalib$^{(BDT)}$and setting M^2 to the mean value of the resonance peak.

$$M^2 = 2p_{T,1}p_{T,2}(\cosh(\eta_1-\eta_2)-\cos(\phi_1-\phi_2)), \quad p_T = E_T \updownarrow$$
$$E_{label,data} = \frac{M^2}{2E_{T,2}(\cosh(\eta_1-\eta_2)-\cos(\phi_1-\phi_2))}, \tag{5.6}$$
$$\text{with } E_{T,2} = \text{Ecalib}^{(BDT)} \text{ and } M^2 = 91.19^2$$

Afterwards, the $E_{label,data}$ can be used directly in training without any additional changes. An important property to evaluate is the accuracy of $E_{truth,data}$. This will be assessed in chapter 7.



(a) Energy estimates from Ecalib$^{(BDT)}$



(b) Energy estimates from $DeepCalo_{Zee,data}$

Figure 5.9: The figure show the BW⊗CB fit to the energy estimations from $DeepCalo_{Zee,data}$ and Ecalib$^{(BDT)}$on Data. The criteria from table 3.7 have been used to select events in Data.

Results from training in Data

Figure 5.9 illustrates the predictions from $DeepCalo_{Zee,data}$ trained using the *Reconstruct* $E_{truth,data}$ method. To ensuring comparability, the test sample is the same as used for $DeepCalo_{Zee,mc}$. Looking at the performance for σ_{CB}, we see a increase of

$$\langle 1 - \frac{\sigma_{CB}^{DeepCalo}}{\sigma_{CB}^{ATLAS}} \rangle = 5.9 \pm 0.9\% \tag{5.7}$$

in performance when compared with Ecalib$^{(BDT)}$. This is a decrease from the $DeepCalo_{Zee,mc}$ model of about $\approx 3\%$, which might lead us to discard this method, but this method has important features other than reducing the width of the BW⊗CB fit. As explained in chapter 1, corrections are made to Ecalib$^{(BDT)}$ to account for discrepancies between MC and Data, and to ensure that the μ_{BW} is not shifted in Data. Looking at μ_{CB} in figure 5.6 and 5.9, we see $\mu_{CB} = 0.073$ GeV for the $DeepCalo_{Zee,data}$ and $\mu_{CB} = -0.5659$ GeV for the $DeepCalo_{Zee,mc}$ model, indicating that when training on Data, the network will learn the corrections to Data. The average shift of $DeepCalo_{Zee,mc}$ and $DeepCalo_{Zee,data}$ for Data can be seen in equation 5.8 with uncertainties.

$$\begin{aligned}\langle \mu_{mc,CB} \rangle &= -1.159 \pm 0.009 \text{ GeV} \\ \langle \mu_{data,CB} \rangle &= -0.004 \pm 0.009 \text{ GeV}\end{aligned} \tag{5.8}$$

$\langle \mu_{mc,CB} \rangle$ is from $DeepCalo_{Zee,mc}$ and $\langle \mu_{data,CB} \rangle$ is from $DeepCalo_{Zee,data}$. As seen in equation 5.8, the $DeepCalo_{Zee,data}$ model reduces the shift of the BW⊗CB fit and determines the μ of the distribution very close to the Z mass. Note that to reconstruct $E_{truth,data}$, the correct mass of the parent particle is needed, so if the mass of the parent particle is unknown, we can not apply this method. Nevertheless, the properties of $DeepCalo_{Zee,data}$ will be useful for the next *DeepCalo* model below.

5.4 *Ensemble method*

As seen above, we were able to train the *DeepCalo* model on MC and Data, each with its own benefits and drawbacks. By combining them, we might be able to take advantage of the benefits without the drawbacks. We have therefore developed a $DeepCalo_{ensemble}$ method, where we mix training on MC and Data, giving the model the possibility of learning the discrepancies between MC and Data and correct for them. As defined previously in section 4.4, when E_{truth} is known, the *logcosh* from equation 4.1 can be applied. Thus, it would be possible to extend the ensemble model of *DeepCalo* to be trained with the ensemble loss function seen in equation 5.9.

$$\begin{aligned}\mathcal{L}(y, \hat{y}) = \mathcal{L}(y_{(\text{Zee, MC})}, \hat{y}_{(\text{Zee, MC})}) + \mathcal{L}(y_{(\text{Zee, Data})}, \hat{y}_{(\text{Zee, Data})}) + \\ \mathcal{L}(y_{(\text{Z}\mu\mu\gamma\text{, MC})}, \hat{y}_{(\text{Z}\mu\mu\gamma\text{, MC})}) + \mathcal{L}(y_{(\text{Z}\mu\mu\gamma\text{, Data})}, \hat{y}_{(\text{Z}\mu\mu\gamma\text{, Data})}) + \\ \mathcal{L}(y_{(H\gamma\gamma\text{, MC})}, \hat{y}_{(H\gamma\gamma\text{, MC})})\end{aligned} \tag{5.9}$$

This will solve the four main problems of the previous models,

1. Non-channel-specific: The same model will be able to reconstruct both electrons and photons from $Z \to ee$, $Z \to \mu\mu\gamma$ and $H \to \gamma\gamma$.
2. Correction to Data made during training.
3. Non-sample-specific: By training on MC and Data, the Data sample might assist the model in the misalignment between MC and Data.
4. Assist $H \to \gamma\gamma$ in Data: The amount of background photons in the $H \to \gamma\gamma$ channel prohibits training on Data. However, by combining $H \to \gamma\gamma$ MC with $Z \to \mu\mu\gamma$ MC/Data, the model might learn the patterns of γ in Data.

For the ensemble model, the *Regression to the mean* method cannot be used, as it calculates the M^2_{inv} in the loss function. This poses a problem, as the scale for the MC and Data loss will not be identical. To test the ensemble method, we will in the following section show the results for the $DeepCalo_{Zee,mc/data}$ ensemble model. Here, *DeepCalo* has been trained in the $Z \to ee$channel on both MC and Data.

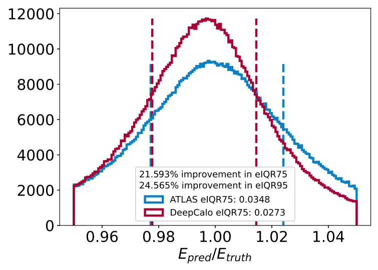


Figure 5.10: Predictions are on MC. The figure show the RE of $Ecalib^{(BDT)}$and $DeepCalo_{Zee,mc/data}$ with IQR75, *re*IQR75 and *re*IQR95 in the legend.

5.4.1 *Analysis for $DeepCalo_{Zee,mc/data}$*

Monte Carlo

We start by evaluating the performance on MC and compare it to $DeepCalo_{Zee,mc}$. The performance can be seen in equation 5.10. Both metrics are within the uncertainties of the $DeepCalo_{Zee,mc}$ model performance. This shows that the additional Data samples do not confuse the accuracy of the model. An example of the MC performance can be seen in figure 5.4.

$$\begin{aligned}\langle reIQR_{75}^{DeepCalo} \rangle &= 22.1 \pm 0.3\% \\ \langle reIQR_{95}^{DeepCalo} \rangle &= 25.3 \pm 0.8\%\end{aligned} \tag{5.10}$$

As seen from the BW⊗CB fit of the $DeepCalo_{Zee,mc}$ model on MC, the *re*IQR75 and σ_{CB} improvements are very similar.

Data

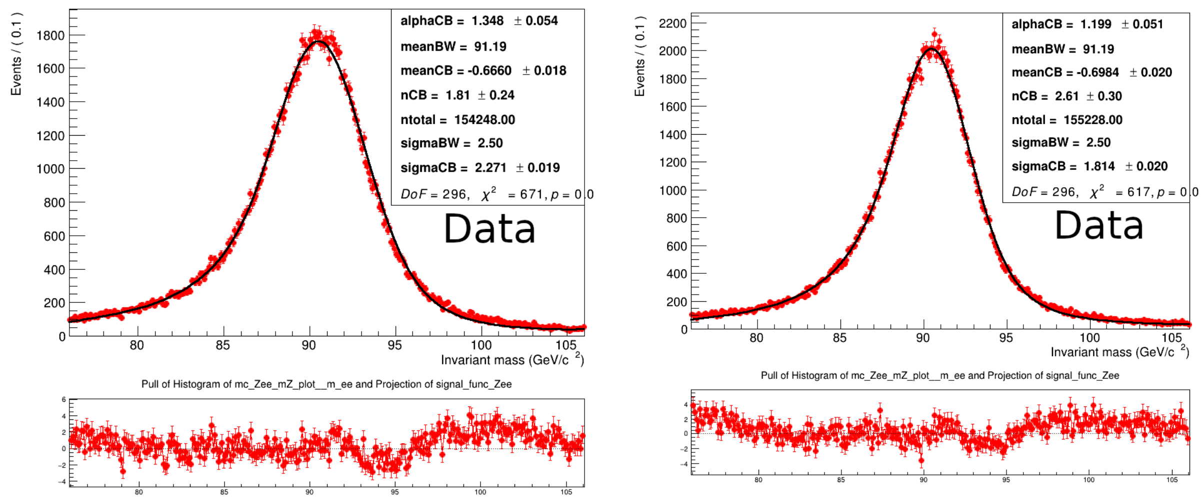
Nevertheless, the essential thing is the performance of *DeepCalo* on Data. An example of the $DeepCalo_{Zee,mc/data}$ performance on Data can be seen in figure 5.12, where we can see a large decrease of σ_{CB}. The combined performance of the four models can be seen in equation 5.11.

$$\langle 1 - \frac{\sigma_{CB}^{DeepCalo}}{\sigma_{CB}^{ATLAS}} \rangle = 1 - \frac{1.86 \pm 0.010}{2.271 \pm 0.019} = 18.3 \pm 0.8\%, \tag{5.11}$$

with an average shift in Data in equation 5.12.

$$\langle \mu_{data,CB} \rangle = -0.392 \pm 0.009 \text{ GeV} \tag{5.12}$$

As seen, the shift has shrunk between $DeepCalo_{Zee,mc/data}$ and $DeepCalo_{Zee,mc}$, with the former now being less than the Ecalib$^{(BDT)}$shift (see equation 5.12).



(a) Energy estimates from Ecalib$^{(BDT)}$ **(b)** Energy estimates from $DeepCalo_{Zee,mc/data}$

Figure 5.11: The figure shows the BW⊗CBfit to the energy estimations from $DeepCalo_{Zee,mc/data}$ and Ecalib$^{(BDT)}$on Data. The criteria from table 3.7 have been used to select events in Data.

The construction and performance of the $DeepCalo_{Zee,mc/data}$ model facilitates further ensemble models, as we see a large performance gain. Many different samples can easily be combined for further generalization. However, when adding multiple sample types, *DeepCalo* might require a variable called *type* for specifying whether the sample is MC or Data, as this will indicate if the network has learned the corrections of Data for that specific channel. This *type* variable should be an additional scalar variable.

We could continue with creating a unified model $DeepCalo_{MC/Data,Zee,H\gamma\gamma,Z\mu\mu\gamma}$ that would be able to reconstruct both electrons a photons. However, due to time constraints, we did not finish the creation of this model. However, as the model might learn MC discrepancies from $Z \rightarrow ee$, it would have been interesting to combine training on Data for $Z \rightarrow ee$ and MC for $H \rightarrow \gamma\gamma$ to see if there would be an improvement in $H \rightarrow \gamma\gamma$ Data.

Learning differences between MC and Data

Training an ensemble method raises some interesting questions when it comes to MC and Data. How prominent is the misalignment and how well is the network able to distinguish between MC and Data? Changing the final activation function to sigmoid, setting truth to binary 0 (Data) and 1 (MC) and changing the loss function to binary cross-entropy, the model can be transformed from a regression model to a classification model. The resulting accuracy can be seen in figure 9.5 in the form of a ROC curve and its AUC. It can be seen that the model is good at distinguishing MC from Data, meaning the misalignment is pretty prominent. If work went into the explainability of *DeepCalo*, the ensemble model might aid the understanding of discrepancies between MC and Data and account for them.

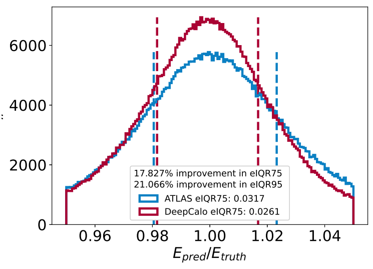


Figure 5.12: Predictions are on MC. General performance of *DeepCalo*$_\sigma$ and $Ecalib^{(BDT)}$using the *re*IQR of the RE.

5.5 *Additional properties from Neural Networks*

Uncertainties of predictions - DeepCalo$_{mc,\sigma}$

As explained in section 4.2.1, we are able to modify *DeepCalo* so the network outputs an energy with an additional uncertainty. We will only be showing the *DeepCalo*$_\sigma$ model for $Z \to ee$ trained on MC and refer to it as *DeepCalo*$_\sigma$, as this section is a proof of concept for the *DeepCalo*$_\sigma$. However, it could easily be extended to other channels. Performance will be measured by sorting S_{test} by σ_M, splitting S_{test} into five bins with 20% of the sample in each bin (similar to figure 5.5). As σ_M should indicate confidence or an uncertainty, it is expected that the lowest σ_M provides the highest precision and that when σ_M increases, performance degrades. Note that the *DeepCalo*$_\sigma$ model has only been run once. The performance of the *DeepCalo*$_\sigma$ will be assessed for both MC and Data, so σ_{CB} will be used as the metric. Therefore, the predicted uncertainties will be used to calculate the uncertainty of the Z mass, denoted σ_M.The expression used to calculate the uncertainty on the Z mass can be seen in chapter 9.

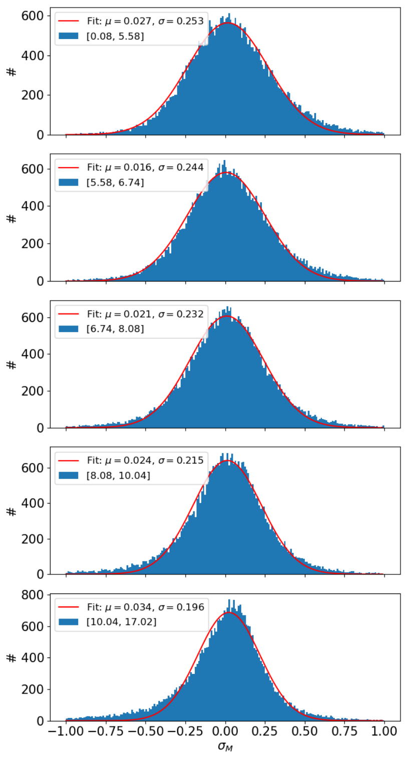


Figure 5.13: Predictions are on MC. The figure shows the pull residual of the five σ bins of *DeepCalo*$_\sigma$. A Gaussian χ^2 fit has been applied to measure the mean and width of the distributions. The parameter of the Gaussian can be seen in the legend of the figure.

Monte Carlo

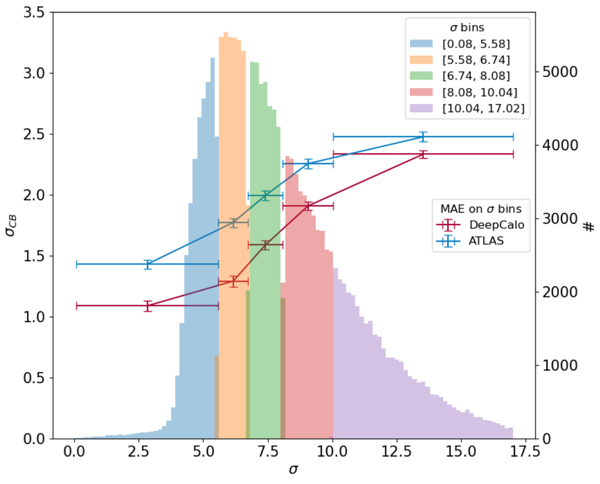
The general *DeepCalo*$_\sigma$ performance can be seen in figure 5.12, where we see a small performance decrease compared with *DeepCalo*$_{Zee,mc}$ (see figure 5.3). As the *DeepCalo*$_\sigma$ model is additionally tasked with predicting σ, this decrease is expected. The performance decrease is not as noticeable as expected. We will discuss the decrease and how it might be improved on in chapter 7. Note that σ will also be used to refer to σ_M

σ_M performance

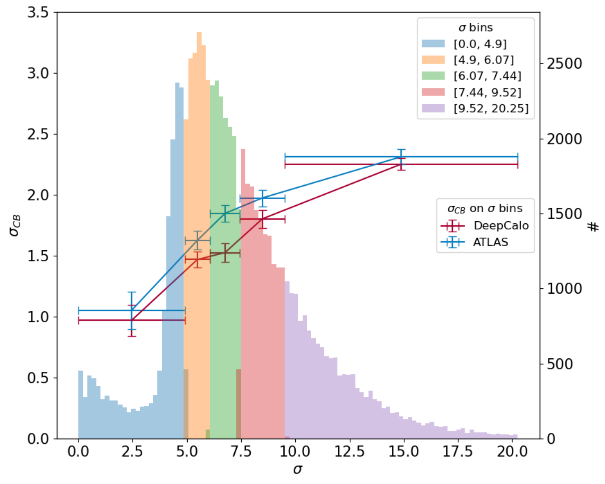
The important property is σ_M and how it can be used as a confidence or uncertainty measure. Figure 5.14 illustrates the performance of *DeepCalo*$_\sigma$ on MC and Data, where σ' has been transformed back to $\sigma = \sqrt{e^{\sigma'}}$ and used to calculate σ_M. The uncertainties have been sorted and divided into five percentiles, with the BW⊗CB fit applied to each bin, measuring the performance of each percentile or σ_M bin. The performance of each bin is shown in figure 5.14 for both MC and Data.

Figure 5.14a is the *DeepCalo*$_\sigma$ performance on MC, it can be seen that σ_M is indeed an uncertainty of the estimation, since σ_{CB} increases with σ_M. We also see *DeepCalo*$_\sigma$ consistently perform more accurately than Ecalib$^{(BDT)}$. Additionally, figure 5.13 shows the pull plot of the five σ_M bins with a Gaussian fit measuring the width and mean of the distributions. If we discard the last $\sigma_M = [10.04, 17.02]$ bin, as it has a tail, the widths of the Gaussians show that the σ_Ms are about 4.5 times too large[3]. While not perfect, the pull residuals have a reasonable distribution to re-scale the uncertainties with 4.5 to achieve an almost unit Gaussian. The re-scaled pull plot can be seen in figure 9.29. For figure 5.14b on Data, we see the same behavior

[3] Because the width of the pull residuals must be 1.



(a) *DeepCalo*$_\sigma$ on MC samples



(b) *DeepCalo*$_\sigma$ on Data samples

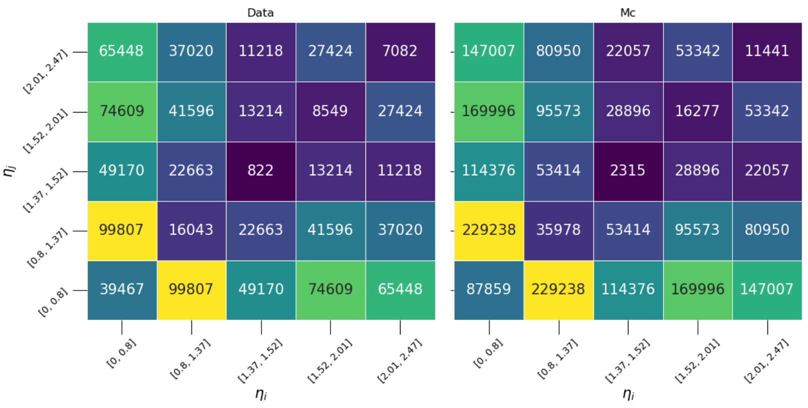
Figure 5.14: The figure shows the performance of the *DeepCalo*$_\sigma$ on MC and Data (left figure and right figure, respectively). The error bar plot in each figure belongs to the left y-axis, which is the σ_{CB} measure. The colored histogram belongs to the right y-axis, which counts the number and range of the σ_M in each of the five bins.

for σ_M, namely that with increasing σ_M the performance degrades. However, we see some irregularities in Data. The distribution of σ_M mostly follows that of MC, except for the first bin at $\sigma > 5$, where a large number of events are located. As can be seen from figure 9.28, there is a cluster of poorly reconstructed events at $E = [0, 10]$ in Data that is not present in MC. These events might be a cause for these irregularities. These irregularities must be investigated before using the confidence in a final model.

5.6 Corrections and resolutions

From the results above, we see a general performance increase by applying *DeepCalo*. As covered in chapter 1, the Ecalib$^{(BDT)}$consists of multiple corrections for the invariant mass between MC and Data to be aligned. In the following section, we will apply one of these corrections, namely an η shift, to each of the $|\eta|$ bins. Afterwards,

we will calculate the $|\eta|$ resolution for both MC and Data to measure the $|eta|$ performance on MC and Data. It will all be applied to the $DeepCalo_{Zee,mc}$.
We illustrate the number of events in each of the $|\eta|$ bins in figure 5.15, as bins with low statistic will have an effect on fitting (see figure 7.1). As seen in figure 5.15, we have multiple bins with low statistics,


Data

η_j \ η_i	[0, 0.8]	[0.8, 1.37]	[1.37, 1.52]	[1.52, 2.01]	[2.01, 2.47]
[2.01, 2.47]	65448	37020	11218	27424	7082
[1.52, 2.01]	74609	41596	13214	8549	27424
[1.37, 1.52]	49170	22663	822	13214	11218
[0.8, 1.37]	99807	16043	22663	41596	37020
[0, 0.8]	39467	99807	49170	74609	65448

Mc

η_j \ η_i	[0, 0.8]	[0.8, 1.37]	[1.37, 1.52]	[1.52, 2.01]	[2.01, 2.47]
[2.01, 2.47]	147007	80950	22057	53342	11441
[1.52, 2.01]	169996	95573	28896	16277	53342
[1.37, 1.52]	114376	53414	2315	28896	22057
[0.8, 1.37]	229238	35978	53414	95573	80950
[0, 0.8]	87859	229238	114376	169996	147007

Figure 5.15: Figure shows the amount of events in each $|\eta|$ bins for both MC (right) and Data (left). The figure is symmetric. The color-coding range is not shared between the two figures.

namely in the crack and at high $|\eta|$. This must be kept in mind when viewing the results, as the fit may not be as accurate as desired.

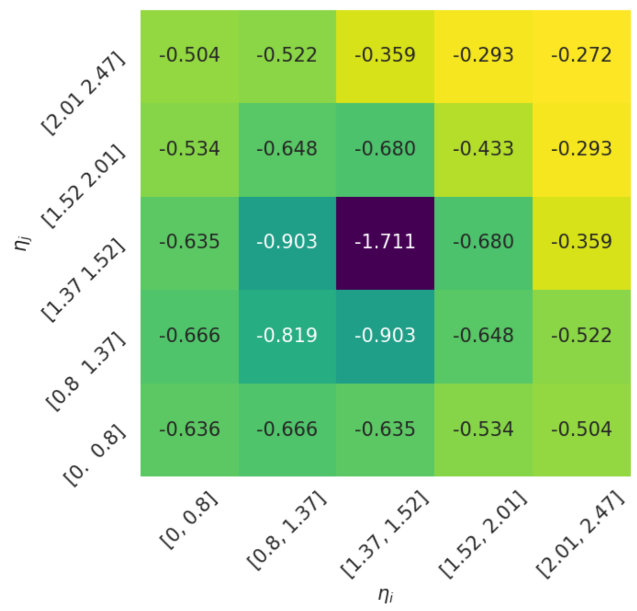
η shifting

As seen in the fit from figures 5.4, 5.6, 5.9 and 5.12, the μ_{CB} is non-zero, so the peak is not centered at the Z-boson mass. This may seem like a small shift, but it can lead to a flattening of the BW⊗CB distribution if the shifts are misaligned in the $|\eta|$ bins, (see figure 9.2). Looking at figure 5.16, we see that μ_{CB} varies significantly on MC and Data. Thus, aligning the distribution might improve the width of BW⊗CB fit.
This shift can be corrected for by applying a small fraction α_i to an energy in the ith $|\eta|$ bin[4]. The complete derivation for finding the α_i can been seen in equation 9.7 with the result in equation 5.13.

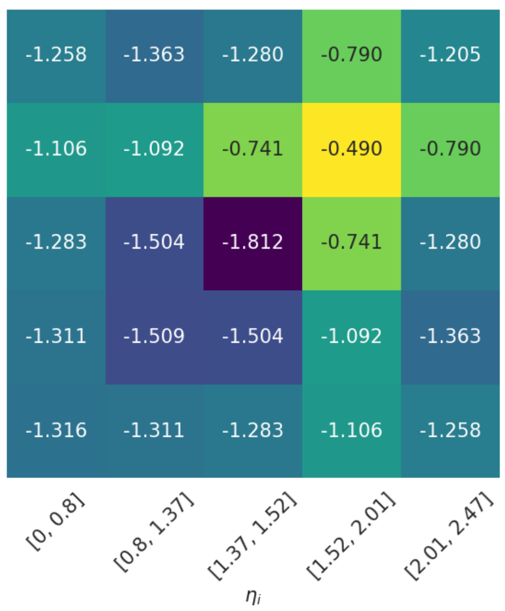
[4] The energy shift will be $E^{corr}_{pred} = E_{pred} \cdot (1 + \alpha_i)$.

$$\chi^2(\alpha) = \sum_{(i,j)\in accpected} \left(\frac{\mu_{ij} - m^{reco}_{Z,ij} \cdot (\frac{\alpha_i + \alpha_i^T}{2})}{\sigma_{\mu_{i,j}}} \right)^2 \tag{5.13}$$

A χ^2 fit is applied to solve for α_i. There may be some bins where the BW⊗CB fit is not successful. These should not contribute to α_i, and only *accpected* (i, j) bins, where the fit was successful, will be used. The resulting α_is for the $|\eta|$ shift can be seen in figure 5.17 for both MC and Data. As a sanity check, we expect a larger shift for Data compared to MC, because the μ_{CB}s in figure 5.16 are larger in Data then MC. From figure 5.17, we see the α_is are almost twice as large for Data compared to MC, so the assumption holds. The result from the $|\eta|$ shift can be seen in figure 5.18. It has to be compared with figures 5.4 and 5.6. The μ_{CB} parameter has shifted significantly closer to the invariant mass of the Z-boson for both MC and Data.



(a) η shift for MC.



(b) η shift for Data.

Figure 5.16: The figures shows the μ_{CB} for both Data (figure a) and MC (figure (b)) in each $|\eta|$ bin.

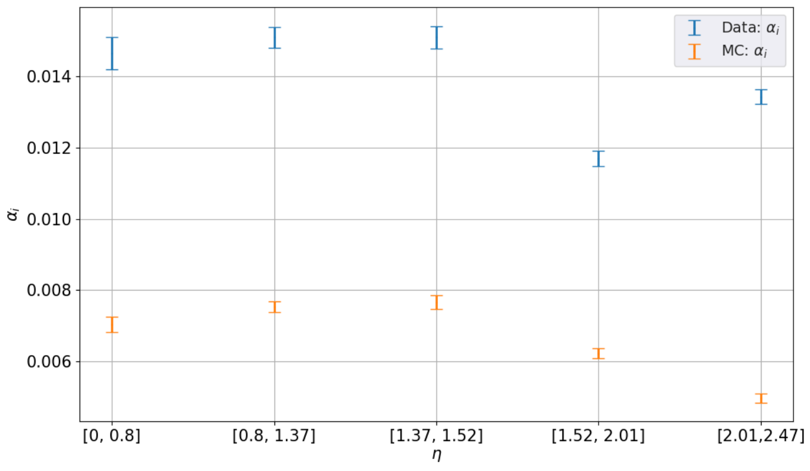
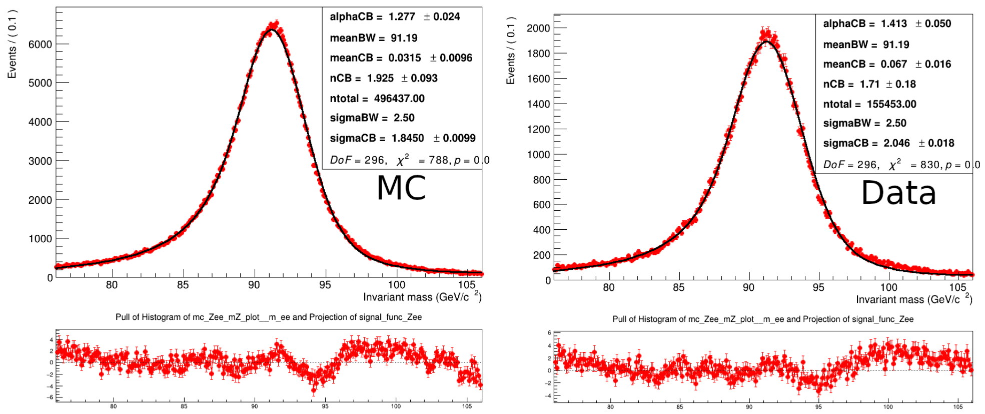


Figure 5.17: The figure shows the α_is from MC and Data for $Z \rightarrow ee$ with uncertainties. The values are computed by solving equation 5.13.

For MC, a small improvement in σ_{CB} of about $\approx 1\%$ can be seen. We sadly see no improvement for Data, as σ_{CB} increases compared with figure 5.6.



(a) $|\eta|$ corrections in MC. **(b)** $|\eta|$ corrections in Data.

Figure 5.18: The BW⊗CB fit of the same model as used in figure 5.6 and 5.4, but with the $|\eta|$ correction from 5.17 added.

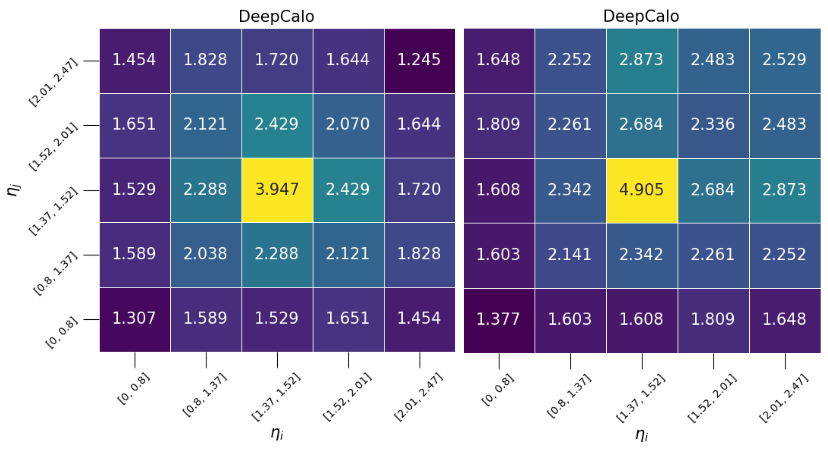
$|\eta|$ *resolution*

Using the results from section 5.6, we are able use the BW⊗CB fit in each $|\eta|$ bin to compute an $|\eta|$ resolution for both MC and Data. We define σ_{CB} of each bin as σ_{ij}, constructed as $\sigma_{ij}^2 = \sigma_i^2 + \sigma_j^2$. We thus have 5 unknowns and 15 knowns from figure 5.19, which we can solve using equation 5.14.

$$\begin{aligned} \sigma_{ij}^2 &= \sigma_i^2 + \sigma_j^2 \\ \chi^2(\sigma_i) &= \sum_{i,j \in accepted} \left(\frac{\sigma_{ij}^2 - (\sigma_i^2 + \sigma_j^2)}{\sigma_{\sigma_{CB}}^2} \right) \Updownarrow \\ \text{we then have to optimize } \chi^2(\alpha) &= \sum_{i,j \in accepted} \left(\frac{\sigma_{ij}^2 - (\alpha^2 + \alpha_T^2)}{\sigma_{\sigma_{CB}}^2} \right) \end{aligned} \tag{5.14}$$

As we use the BW⊗CB fit, we need to make sure not to use σ_{CB} from fits that do not follow its distribution. Thus, as previously, we only use *accepted* i, j in the sum. In figure 5.19, we can see the σ_{CB} measure from each $|\eta|$ bin. Solving equation 5.14 with the values from figure 5.19, we obtain the $|\eta|$ resolutions seen in figure 5.20. Here, we see a quite significant improvement between *DeepCalo* and Ecalib$^{(BDT)}$, except for the bin at large $|\eta|$, where we only see a small improvement. We see an interesting behavior when comparing the MC and Data performance between *DeepCalo* and Ecalib$^{(BDT)}$. The Ecalib$^{(BDT)}$ reconstruction is consistently more accurate for Data than MC, where *DeepCalo* has the opposite behavior. It would be expected that the MC performance for both models are the most accurate as ooth are trained in MC.

However, additional corrections to the Ecalib$^{(BDT)}$ have been made in Data, and this may indicate that these correction have significantly improved Ecalib$^{(BDT)}$ Data performance.



(a) σ_{CB} of the BW$\otimes$CB fit in MC. **(b)** σ_{CB} of the BW$\otimes$CB fit in Data.

Figure 5.19: The figure shows the σ_{CB} from *DeepCalo*$_{Zee,mc}$ for both MC (figure a) and Data (figure (b)) in each $|\eta|$ bin. Its $Ecalib^{(BDT)}$ counterpart can be seen in figure 9.3. The number of events per bin can be seen in figure 5.15.

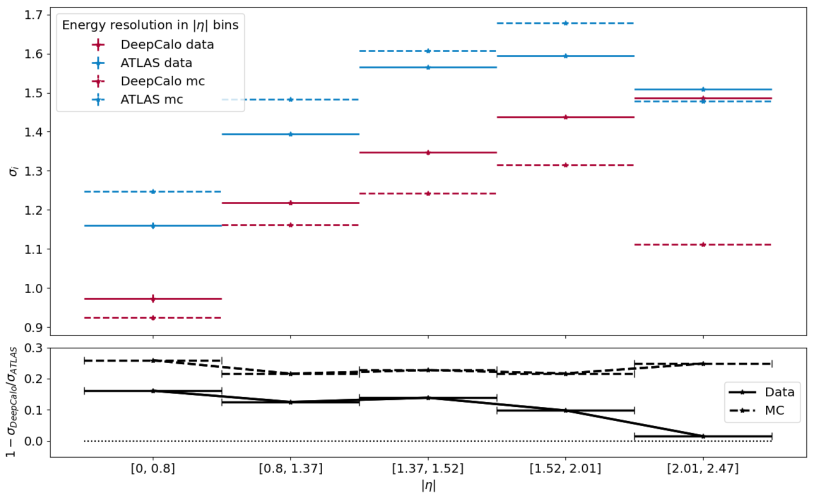


Figure 5.20: The figure shows the resolution of *DeepCalo* and $Ecalib^{(BDT)}$ in the $|\eta|$ bins. The resolution is found by solving equation 5.14 using values from figure 5.19. While each resolution has an error associated with it, it is very small and not visible on the figure. The dashed lines are the MC resolution, and the solid ones are Data.

6 Photon Reconstruction

In continuation with the previous chapter, we will in the following one evaluate the photon energy reconstruction capabilities of *DeepCalo*. Note that some of the *DeepCalo* properties shown in chapter 5 also exist for photon reconstruction, but that they will not be shown, as the capabilities have already been assessed in the previous chapter. From table 4.6, the decay channels in question ar $H \to \gamma\gamma$ (for MC) and $Z \to \mu\mu\gamma$ (for MC/Data). Note that chapter and 4 describe an architecture optimized for $Z \to ee$. However, onl small changes have been made in the architecture to process photons Most notably, the input to *ScalarNet* has been changed, as photons d not have charge, and are therefore not measured in the ID, resultin in no track associated with them.

$$H \to \gamma\gamma$$

6.1 Analysis of $DeepCalo_{Hyy,mc}$

6.1.1 Monte Carlo

Energy evaluation

For energy reconstruction in MC, we use the same metric as in the previous chapter. We compare $DeepCalo_{Hyy,mc}$ and $\text{Ecalib}^{(BDT)}$ to E_{truth} and use the *re*IQR to compare the two models. In equation 6.1, we see the performance of the $DeepCalo_{Hyy,mc}$ model with an example of the RE in figure 6.3.

$$\begin{aligned} \langle reIQR_{75}^{DeepCalo} \rangle &= 12.8 \pm 0.3\% \\ \langle reIQR_{95}^{DeepCalo} \rangle &= 12.4 \pm 0.4\% \end{aligned} \tag{6.1}$$

We see an improvement of $\approx 12\%$ for the photon reconstruction in $H \to \gamma\gamma$ for both *re*IQR$_{75}$ and *re*IQR$_{95}$, which is a significant performance gain. However, compared with the improvements in electron reconstruction, this is a diminution. Further investigations are needed to shed light on the shortcomings of the model.

The correlation plot in figure 6.1 may give a small hint to the decline in performance. As we saw before, for $Z \to ee$, *DeepCalo* improves the underestimated low energy outliers compared to $\text{Ecalib}^{(BDT)}$. However, the general performance of $\text{Ecalib}^{(BDT)}$on $H \to \gamma\gamma$ is far more

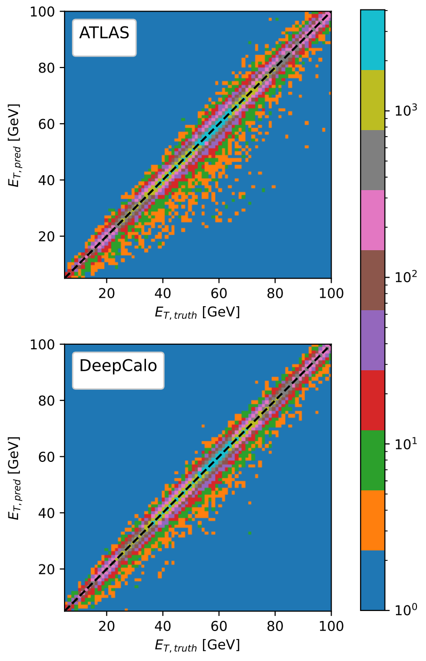


Figure 6.1: Predictions are on MC. The figure shows a correlation plot between $E_{T,truth}$ and the model prediction from $\text{Ecalib}^{(BDT)}$or $DeepCalo_{Hyy,mc}$. The black dashed line is a linear line. The color concentration is logarithmic, indicating the amount of events.

accurate to E_{truth} than its $Z \rightarrow ee$ counterpart (see figure 5), and, as a result, making improvements to Ecalib$^{(BDT)}$ becomes more difficult. The accuracy of Ecalib$^{(BDT)}$ in $H \rightarrow \gamma\gamma$ and $Z \rightarrow ee$ can be seen in the legend of figure 5.3 and 6.3[1].

[1] For $H \rightarrow \gamma\gamma$, the ATLAS eIQR$_{75}$ = 0.0197 but for $Z \rightarrow ee$ it is eIQR$_{75}$ = 0.0348, meaning Ecalib$^{(BDT)}$ for $H \rightarrow \gamma\gamma$ is much closer to E_{truth} than for Z $\rightarrow$ ee.

The performance in the $|\eta|$ vs E_T bins can be seen in figure 6.2 with the bottom plot comparing the two models. We see at low $|\eta|$ values that the performance gain is roughly 20%. However, at $|\eta| \geq 1.5$ (in and after the crack), we see a performance decline for the $|\eta|$ and E_T bins. Figure 6.2 shows the underlying reason for a performance

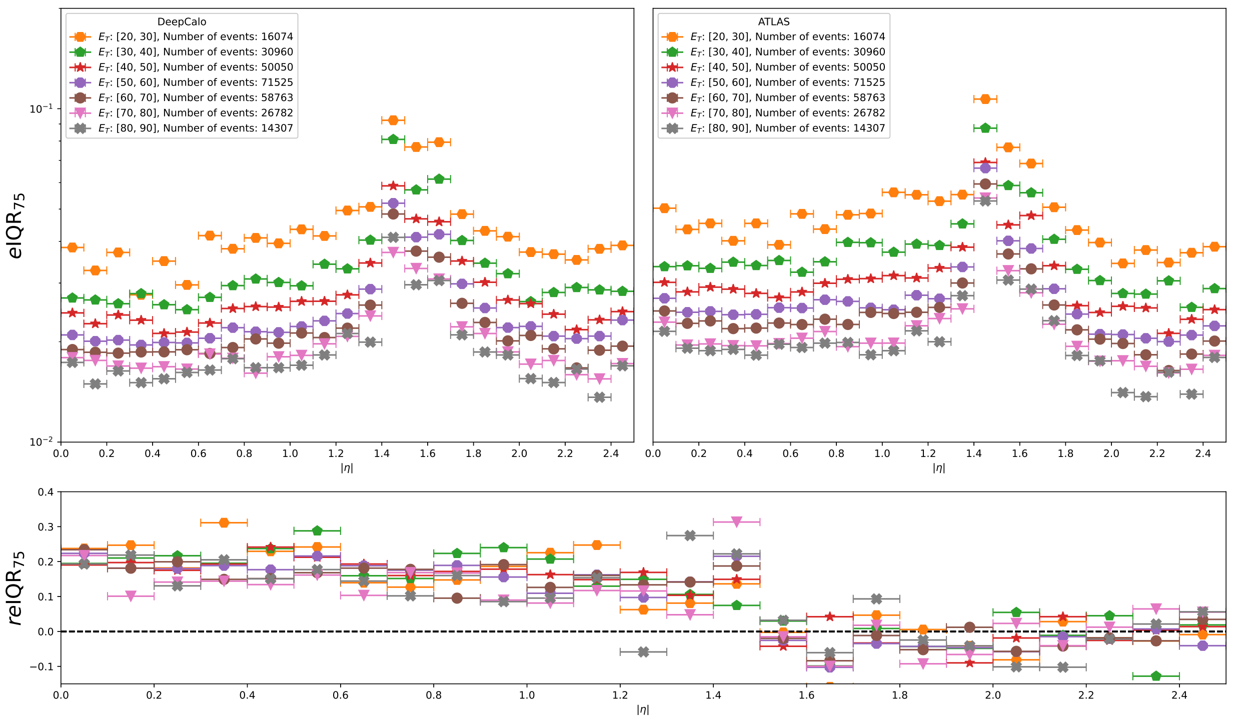


Figure 6.2: Predictions are on MC. $eIQR_{75}$ of $\sigma_{calib}/\sigma_{truth}$ as a function of $|eta|$ and E_T with a bin size of 0.1 in $|\eta|$. They share a *y axis* and the legend shows the number of events in each E_T bin.

gain of only $\approx$ 12%. The photons beyond the crack are poorly reconstructed. Increasing performance at $|\eta| \geq 1.5$ will help the general reconstruction performance of $DeepCalo_{Hyy,mc}$ (see figure 6.3). A discussion on improving the performance in the $|\eta| \geq 1.5$ region is included in chapter 7. For more detailed plots, see figure 9.15, where the amount of events in each bins can be seen as well.

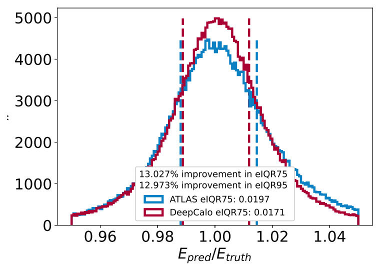


Figure 6.3: Predictions are on MC. E_{calib}/E_{truth} for Ecalib$^{(BDT)}$ and $DeepCalo_{MC}$ with reIQR as in the legend. The more narrow the distribution, the better.

The pileup behavior of $DeepCalo_{Hyy,mc}$ can be seen in figure 6.4, where the pileup range has been divided into five bins. The figure shows the Ecalib$^{(BDT)}$ model suffering more than *DeepCalo* from pileup. This follows the same behavior as the $DeepCalo_{Zee,mc}$ model, which is promising for the future upgrades of LHC and ATLAS. Unfortunately, $H \rightarrow \gamma\gamma$ samples with ECAL images are unavailable in Data. Thus, performance on Data can not be shown.

However, using the $Z \rightarrow ee$ reconstruction as a basis, we can give an approximation of performance on Data for $H \rightarrow \gamma\gamma$. For $Z \rightarrow ee$, we saw a deduction of 50% from $DeepCalo_{Zee,mc}$ to Data. Assuming this holds, we would expect an improvement of about $\approx$ 6% for

$H \to \gamma\gamma$. However, we make this assumption tentatively, as the relationship between MC and Data for electrons might not be the same as photons.

The invariant mass distribution and additional information to the $|\eta|$ vs E_T figure can be found in section 9.5.3. A double Gaussian has been used to fit the invariant mass of the $H \to \gamma\gamma$.

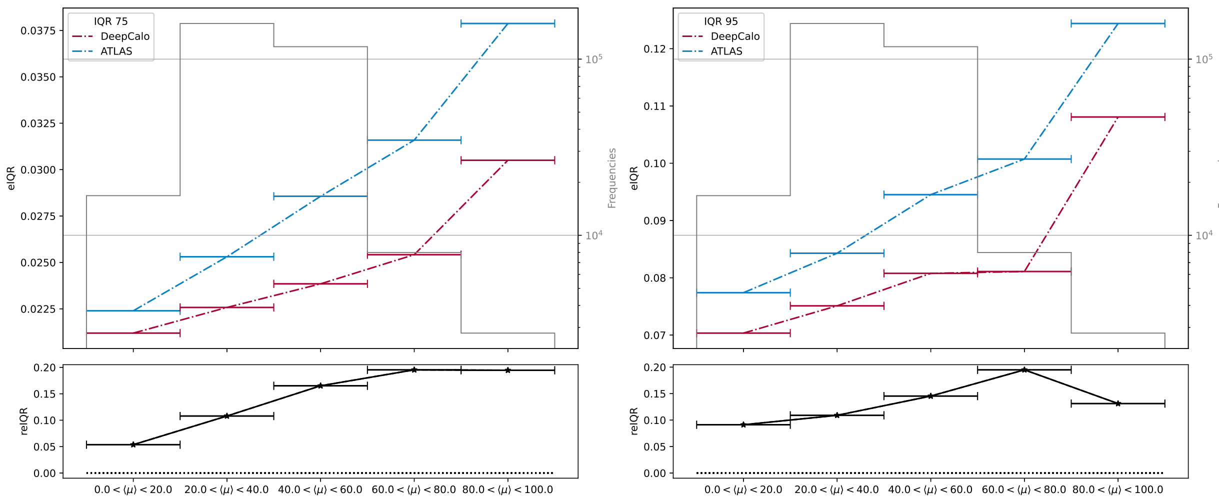


Figure 6.4: Predictions are on MC. The figure shows the pileup performance of the $DeepCalo_{Hyy,mc}$ model in both eIQR$_{75}$ and eIQR$_{95}$. The error bar plot indicates the eIQR with its x-errors showing the range of the bins. The histogram shows the number of events in each bin. The bottom plot shows the reIQR of the eIQR between *DeepCalo* and Ecalib$^{(BDT)}$.

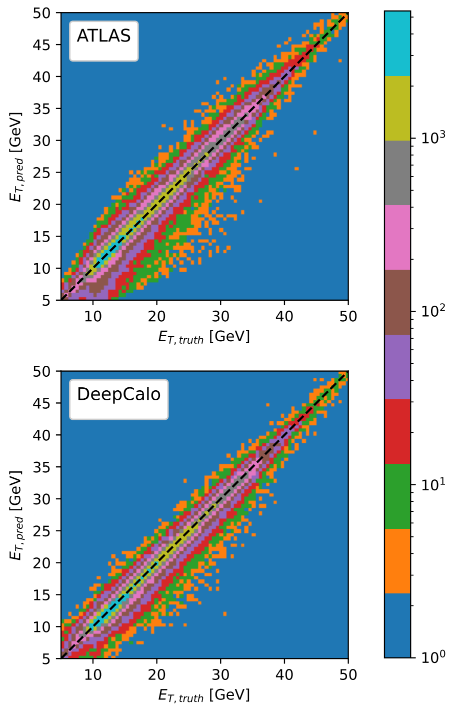
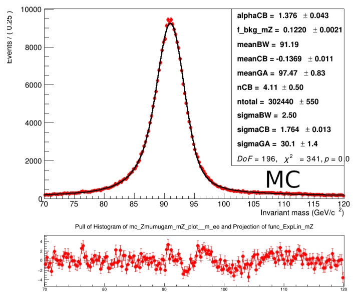
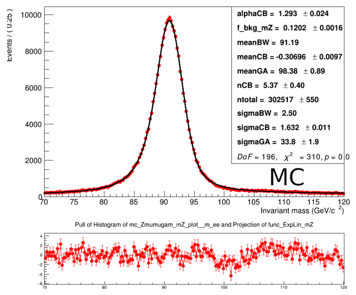


Figure 6.5: Predictions are on MC. The figure shows a correlation plot between $E_{T,truth}$ and the model prediction from Ecalib$^{(BDT)}$ or $DeepCalo_{Hyy,mc}$. The black dashed line is a linear line. The color concentration is logarithmic, indicating the amount of events.



(a) Energy estimates from Ecalib$^{(BDT)}$



(b) Energy estimates from $DeepCalo_{Z\mu\mu\gamma,data}$

Figure 6.6: The figure shows the BW⊗CB fit to the energy estimations from $DeepCalo_{Z\mu\mu\gamma,mc}$ and Ecalib$^{(BDT)}$ on MC.

$Z \to \mu\mu\gamma$

6.2 *Analysis of $DeepCalo_{Z\mu\mu\gamma,mc}$*

The photon reconstruction performance in Data cannot be tested for $H \to \gamma\gamma$ as the Data files are unavailable. However, for $Z \to \mu\mu\gamma$ Data images are available and can be used for photon reconstruction.

Note that the energy ranges between $Z \to \mu\mu\gamma$ and $H \to \gamma\gamma$ are only mildly overlapping (see figure 1.11), so $Z \to \mu\mu\gamma$ on Data can only give us a small indication of performance for low energy photons, and cannot be directly translated to $H \to \gamma\gamma$ performance.

Monte Carlo

Energy evaluation

To validate the MC performance for $Z \to \mu\mu\gamma$, the same metric as previously will be used. An example of the $DeepCalo_{Z\mu\mu\gamma,mc}$ model can be seen in figure 6.8. From equation 6.2, the $DeepCalo_{Z\mu\mu\gamma,mc}$ model achieves a average improvement of $\approx$ 18% compared to Ecalib$^{(BDT)}$.

$$\begin{aligned} \langle reIQR_{75}^{DeepCalo} \rangle &= 18.0 \pm 0.8\% \\ \langle reIQR_{95}^{DeepCalo} \rangle &= 18.3 \pm 1.4\% \end{aligned} \tag{6.2}$$

From the correlation plot in figure 6.1.1, the model has no visible biases, but, as seen from the previous correlation plots, a visible reduction in outliers compared to Ecalib$^{(BDT)}$.

The $|\eta|$ vs E_T performance is illustrated in figure 6.7. Before the crack region, the gain in performance using *DeepCalo* is between 15%-35%, with some of the low statistic bins fluctuating. However, the model is again struggling with reconstructing photons beyond the crack region. Possible improvements and causes will be discussed in section 7.4 and 7.5, respectively.
The pileup evaluation for MC and Data are illustrated in figure 6.10 and figure 9.19.

Mass evaluation

The invariant mass of $Z \to \mu\mu\gamma$ is dominated by the resolution of two muons. Hence, testing the BW⊗CB metric on MC samples is important in order for the BW⊗CB fit to be translated to a photon resolution. It is highly recommended to take a look at figure 9.17 before continuing, where $E_{truth,\gamma}$ has been applied. It shows the muon domination of the fit. Due to the background photons in the $Z \to \mu\mu\gamma$ selection, a background function is required. This is explained in section 4.2.2, but, in summary, it is a BW⊗CB fit with an additional Gaussian term for the background distribution with the

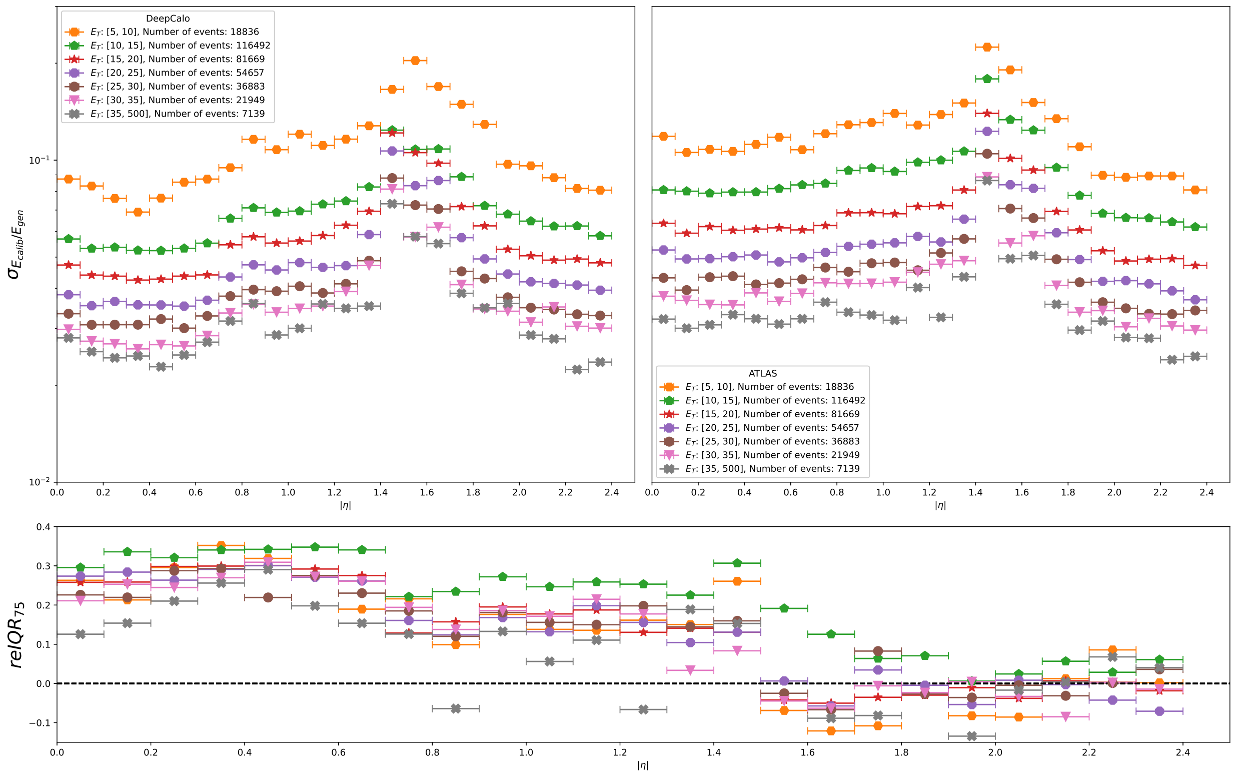


Figure 6.7: $eIQR_{75}$ of $\sigma_{calib}/\sigma_{truth}$ as a function of $|eta|$ and E_T with bin size of 0.1 in $|\eta|$. They share a *y axis* and the legend shows the number of events in each E_T bin.

f_bkg_mZ parameter as the ratio of background function used. An example of the $DeepCalo_{Z\mu\mu\gamma,mc}$ performance on the BW⊗CB fit on MC can be seen in figure 6.6, with the general $DeepCalo_{Z\mu\mu\gamma,mc}$ performance being shown in equation 6.3.

$$\langle 1 - \frac{\sigma_{CB}^{DeepCalo}}{\sigma_{CB}^{ATLAS}} \rangle = 1 - \frac{1.625 \pm 0.006}{1.764 \pm 0.013} = 7.9 \pm 0.8\%. \tag{6.3}$$

Despite the dominance of muons, the $DeepCalo_{Z\mu\mu\gamma,mc}$ achieves an $7.9 \pm 0.8\%$ improvement over Ecalib$^{(BDT)}$on MC. Using σ_{CB} from figure 9.17, the muon dominance can be removed so the improvement in photon reconstruction from BW⊗CB can be measured.

$$1 - \frac{1.625 \pm 0.006 - 1.269 \pm 0.012}{1.764 \pm 0.013 - 1.269 \pm 0.012} = 28.1 \pm 2.4\% \tag{6.4}$$

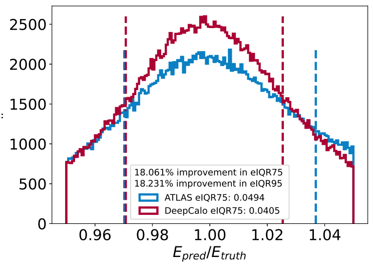


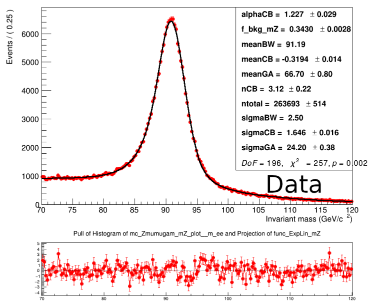
Figure 6.8: Predictions are on MC. E_{calib}/E_{truth} for Ecalib$^{(BDT)}$and $DeepCalo_{MC}$ with *rel*IQR as in the legend. The narrower the distribution, the better.

An increase in photon reconstruction will only improve a fraction of the CB width, as it is dominated by the muons. However, removing the muon dominance in the BW⊗CB fit reveals a 28.1% improvement in the photon reconstruction.

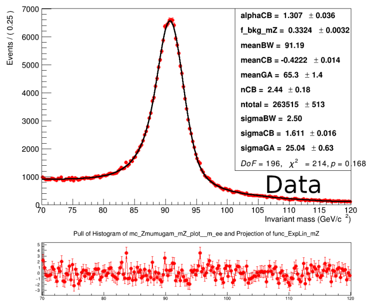
Data performace

The general performance of the $DeepCalo_{Z\mu\mu\gamma,mc}$ model can be seen in equation 6.6. An example of a fit can be seen in figure 6.9.

$$\langle 1 - \frac{\sigma_{CB}^{DeepCalo}}{\sigma_{CB}^{ATLAS}} \rangle = 1 - \frac{1.606 \pm 0.009}{1.646 \pm 0.016} = 2.4 \pm 1.1\% \tag{6.5}$$



(a) Energy estimates from $Ecalib^{(BDT)}$



(b) Energy estimates from $DeepCalo_{Z\mu\mu\gamma,data}$

Figure 6.9: Predictions are on MC. The figure shows the BW⊗CB fit to the energy estimations from $DeepCalo_{Z\mu\mu\gamma,mc}$ and $Ecalib^{(BDT)}$on Data. The selection of events follows table 3.7.

The $DeepCalo_{Z\mu\mu\gamma,mc}$ achieves a small improvement in the BW⊗CB fit. Assuming the muon dominance in Data is similar to MC, we can remove the dominance from the BW⊗CB fit.

$$1 - \frac{1.606 \pm 0.009 - 1.269 \pm 0.012}{1.646 \pm 0.016 - 1.269 \pm 0.012} = 10.6 \pm 4.5\% \qquad (6.6)$$

Thus, with the assumption that muon reconstruction performance is transferable between MC and Data, the $DeepCalo_{Z\mu\mu\gamma,mc}$ model achieves a photon reconstruction improvement of 10.6% compared to $Ecalib^{(BDT)}$. However, the muon reconstruction performance on MC and Data has not been tested.
The pileup performance in both MC and Data can be seen in figure 6.10 using σ_{CB} as a metric (so it will be affected by the muon dominance).

The improvements in pileup are not as prominent as for previous models (due to the muon dominance), but a general improvement is visible. Note that the muon dominance has not been removed in the σ_{CB}s. For $Z \to ee$ we were able to calculate the $|\eta|$ resolution for MC and Data (see figure 5.20). Computing the $|\eta|$ resolution for $Z \to \mu\mu\gamma$ is easier, as only a single particle in each event is reconstructed. Thus, sorting photons in the $|\eta|$ bins and measuring the σ_{CB} will reveal their resolution (with muon dominance). The $|\eta|$ resolutions for MC and Data can be seen in figure 6.11 for both $DeepCalo_{Z\mu\mu\gamma,mc}$ and $Ecalib^{(BDT)}$. We see an improvement over $Ecalib^{(BDT)}$ across the

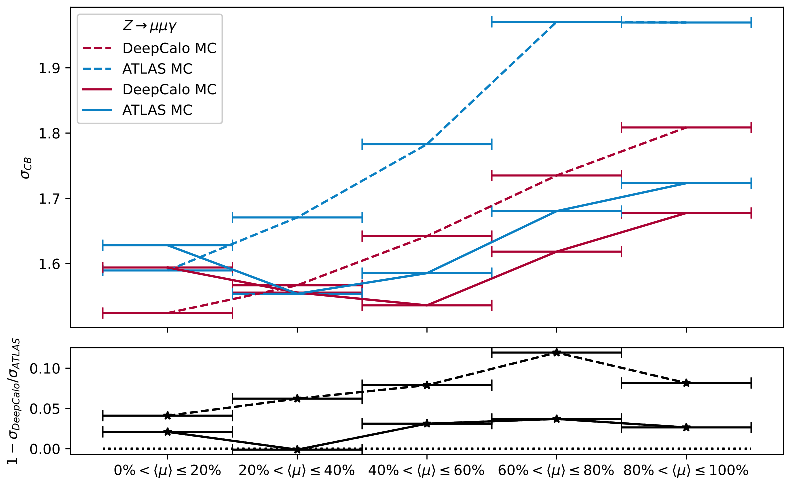


Figure 6.10: The figure shows the pileup performance of the $DeepCalo_{Zmumugam,mc}$ and $Ecalib^{(BDT)}$models on MC and Data. The metric on the y-axis is the σ_{CB} value and the bottom plot shows the ratio comparing the two models.

five $|\eta|$ bins for MC and Data. For MC, $DeepCalo_{Z\mu\mu\gamma,mc}$ had problems with reconstructing photons beyond the crack region, which is also seen in figure 6.11. However, this behavior is not present in the Data resolution. It is mostly constant in the $|\eta|$ bins with a $\approx 2.5\%$ improvement following equation 6.6.

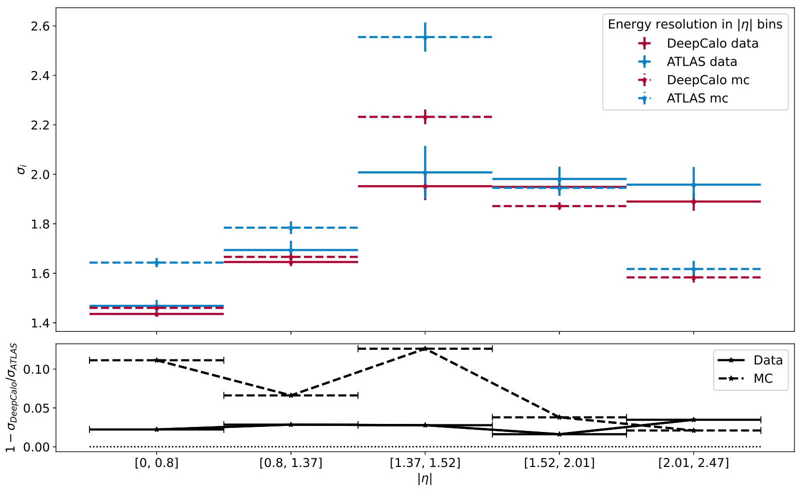


Figure 6.11: The figure shows the $|\eta|$ resolution of $DeepCalo_{Z\mu\mu\gamma,mc}$ and $E\text{calib}^{(BDT)}$on MC and Data. The MC performance is shown with dashed lines and the Data performance with solid lines. The metric used is σ_{CB} from the BW$\bigotimes$CB fit.

7 Discussion

7.1 Metric performance

Fitting is an art and the BW$\otimes$CB fit is very advanced and has many parameters that can be tweaked. In the following section, we will show the dependency of σ_{CB} on some of these parameters to ensure that they are not biased towards a specific regression algorithm. Predictions from the $DeepCalo_{Zee,mc}$ model will be used.

Effects of number of bins

In figure 7.1, the σ_{CB} measure at varying numbers of bins of the BW$\otimes$CB fit can be seen. Fortunately, it shows that the number of bins does not have an significant effect on the σ_{CB} and, in the bottom plot, the ratios are within the uncertainties of each other.

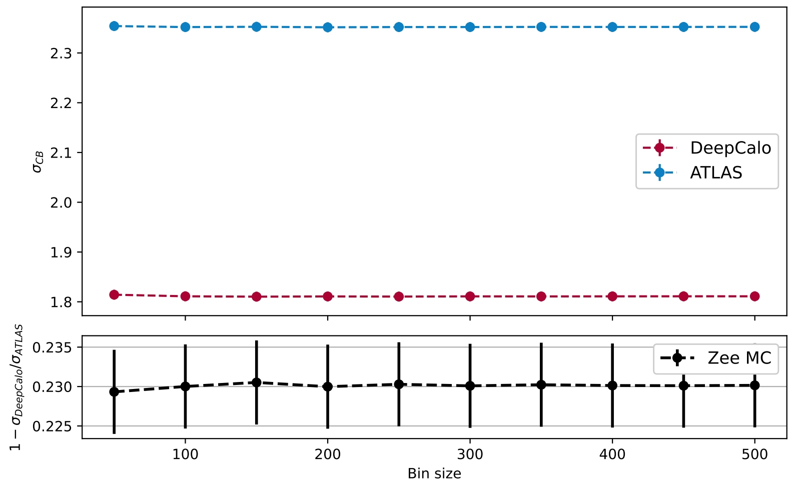


Figure 7.1: The figure shows the influence in the number of bins on the BW$\otimes$CB fit. In the *top figure*, the y-axis shows the σ_{CB} of the different BW$\otimes$CB fits at different numbers of bins. The *bottom figure* shows the relative metric used to compare *DeepCalo* and Ecalib$^{(BDT)}$. This is use to see if the metric is biased towards any of them. The decay is $Z \rightarrow ee$ and $DeepCalo_{Zee,mc}$ is used.

Effects of number of events

There may be fluctuations in σ_{CB} dependent on the number of events in the fit. To see the influence of the fluctuations, we vary the number of events in $Z \rightarrow ee$ and apply BW$\otimes$CB fit to measure the σ_{CB}. The results are shown in figure 7.2, where we see a variation in σ_{CB} at a low number of events, but no significant effects on the BW$\otimes$CB fit above 250.000 events. The variation at $<$ 100.000 events may be a problem, as performance in $|\eta|$, E_T and $\langle\mu\rangle$ are often split in bins

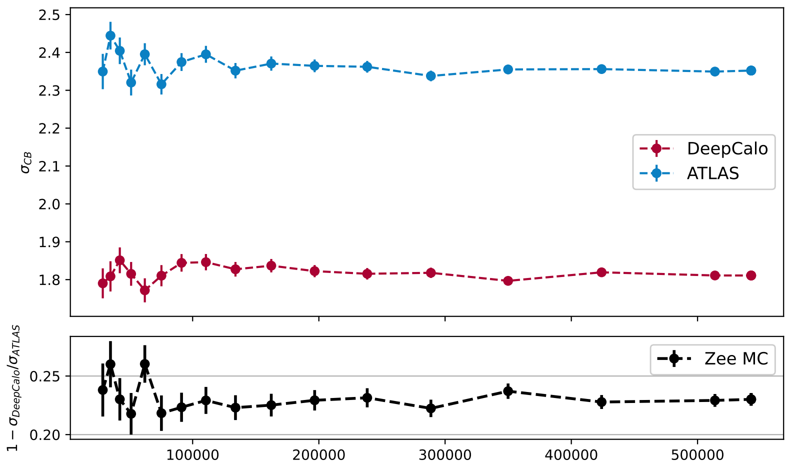


Figure 7.2: The figure shows the influence of the number of events on the BW⊗CB. The *top figure*, the y-axis shows the σ_{CB} of the different BW⊗CB fits at different numbers of events. The *bottom figure* shows the relative metric used to compare *DeepCalo* and Ecalib$^{(BDT)}$. This is use to see if the metric is biased towards any of them. The decay is $Z \to ee$ and *DeepCalo*$_{Zee,mc}$ is used.

with a few number of events (see figure 5.15). This must be kept in mind when reviewing the performance for $|\eta|$, E_T and $\langle\mu\rangle$ of the models.

Effects of nCB and α_{CB}

It is important to note the other fitting parameters of the *CB*, as they will have small effects on the width measure of the distribution. These parameters are *nCB* and α_{CB}, and their effects can be seen in figure 7.3.

The expression for the *CB* function can be seen in section 9.1.1, where the parameters are explained as well. We will not go into a detailed description of the *CB* parameters. However, it is important to note that when using σ_{CB} as a performance measure between two model, *nCB* and α_{CB} have to be similar between the models. The fits used in this thesis, *nCB* and α_{CB}, are within the reasonable intervals.

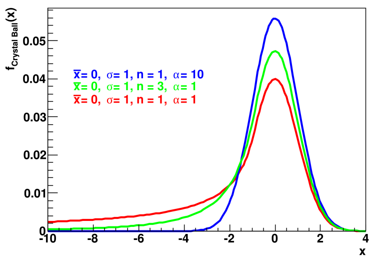


Figure 7.3: The figure shows the *CB* distribution with different *nCB* and α_{CB} values. Here, it is possible to see how these parameters affect the width of the *CB*. Figure from [49].

7.1.1 Correlation plots

In the MC analysis in chapter 5 and 6, many correlation plots between $E_{T,truth}$ and $E_{T,pred}$ were shown. As the reader might have noticed, the correlation plots from Ecalib$^{(BDT)}$ has a *cone-like* shape of outliers. This is due to equation 3.4 in section 3.4, where $k = 0.6$ was applied to remove outliers above k. If equation 3.4 had not been applied, the correlation plot for figure 6.1.1 would look like figure 7.5 (for $Z \to \mu\mu\gamma$). As seen in the correlation plot of Ecalib$^{(BDT)}$, a large amount of outliers are present, especially at low energies.

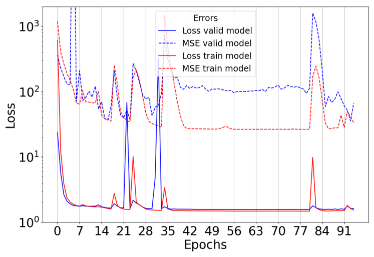


Figure 7.4: The figure displays the convergence of *DeepCalo*$_\sigma$. Both dashed lines are MSE, and the solid line is the σ loss function given in equation 4.3. The color indicates training or validation samples.

By applying equation 3.4, the performance of Ecalib$^{(BDT)}$ will be improved, as outliers are removed. However, for some events it is difficult to know whether an outlier is due to an error in the clustering or poor reconstruction by Ecalib$^{(BDT)}$. A restricting $k = 0.6$ was chosen, ensuring that only "correctly" reconstructed Ecalib$^{(BDT)}$ events were selected. Applying k will result in

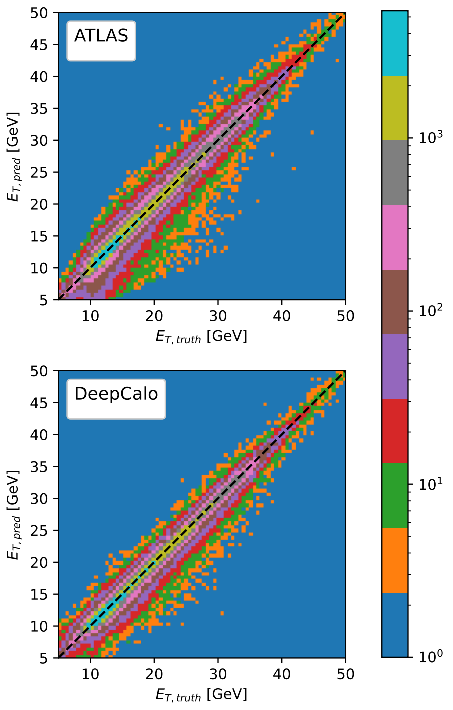


Figure 7.5: Correlation plot between $E_{T,truth}$ and $E_{T,pred}$, where equation 3.4 has not been used. The color-scale is logarithmic.

- improving the Ecalib$^{(BDT)}$ accuracy, as poorly reconstructed event will also be removed,
- ensuring an Ecalib$^{(BDT)}$-biased performance comparison between Ecalib$^{(BDT)}$and *DeepCalo*,
- making the *DeepCalo* less robust, as the events Ecalib$^{(BDT)}$failed to reconstruct will be removed. This might be unique events that are present in Data.

Having a Ecalib$^{(BDT)}$-biased performance comparison is acceptable, as conclusions can easily be drawn when we see a large performance increase using *DeepCalo*. However, we would like to train on poorly reconstructed events. A solution could be a wider k or to only remove events where the pair share E_{truth}.

7.2 Loss function of DeepCalo$_\sigma$

In section 5.5, the performance of *DeepCalo*$_\sigma$ was evaluated. It revealed a small performance decrease compared to *DeepCalo*$_{Zee,mc}$. This was expected and can be explained from figure 7.6. For the solid line, at a large σ value, the difference between the lines are non-existent, meaning the error term $\left((y-\hat{y})^2\right)$ is meaningless. When decreasing σ, the error term increases its influence, and at $\sigma = 10$ we begin to see a noticeable difference between the error$=$ 10 and the lower errors. However, for the dashed line (σ'), the error term has no influence at $\sigma = 10$ and σ has to be significantly reduced before a noticeable difference between errors can be seen. This late

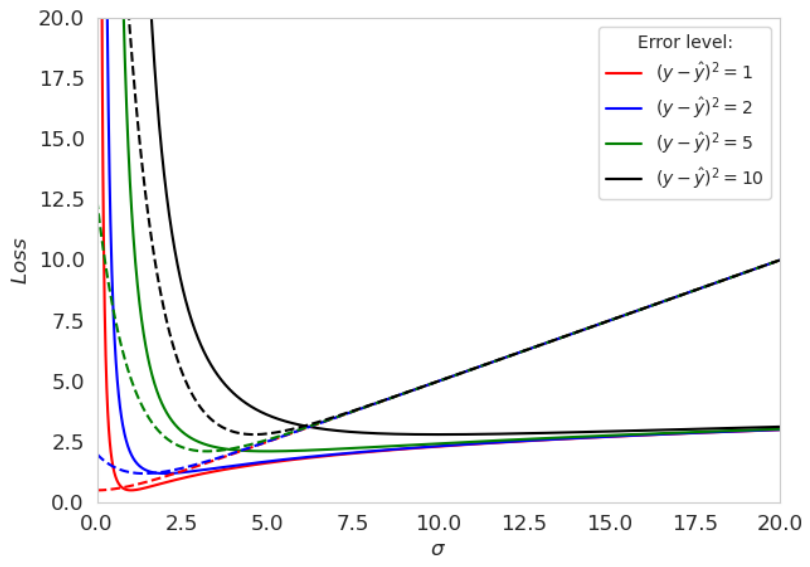


Figure 7.6: The figure displays the distributions of equation 4.3 (solid line) and 4.4 (dashed line) at different $(y-\hat{y})^2$ levels, also called *error levels*.

error optimization for the dashed line will have an effect on the precision of the predictions. As an example, figure 7.4 shows the convergence of *DeepCalo*$_\sigma$. Here, the misaligned spikes indicate the focus of minimizing σ rather than `MSE`. A solution could be to add an additional error term. We propose a solution by adding an error term

$(y-\hat{y})/(1+\sigma')$ to the loss function.

$$\mathcal{L}(y,\hat{y},\sigma) = \frac{1}{2}\left(\frac{(y-\hat{y})^2}{e^{\sigma'}} + \sigma' + \frac{1}{1+\sigma'}(y-\hat{y})\right) \tag{7.1}$$

The resulting distribution can be seen in figure 7.7, where the error now has influence at higher σs.

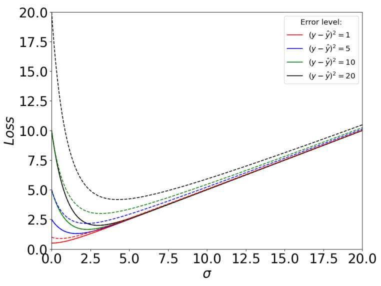


Figure 7.7: The figure displays the same as figure 7.6, except for the addition of equation 7.1.

7.3 *Reconstruction of E_{truth} in Data*

In chapter 5, *DeepCalo* was trained on Data on $Z \to ee$ samples using methods described in section 4.2. $E_{label,data}$ was constructed knowing it contained an error. This error can be measured by applying the same method on MC (creating $E_{label,mc}$) and comparing it with E_{truth}. The correlation plots of E_{acc} and $E_{label,mc}$ to E_{truth} can be seen in figure 7.8, with a logarithmic color bar indicating the number of events per bin. There is a high concentration of events following the dashed black line, which means $E_{label,mc}$ is a good approximation to E_{truth}. However, there are also outliers with significantly large errors with regards to the E_{truth}, some with more than double the E_{truth}. For the E_{acc}, we do not see the size and amount of outliers as before, however, it has the bias of underestimating the energy as well as being an affine linear line[1], meaning the shift is not uniform.

[1] What is meant by affine is that the distribution is not parallel to the black dashed line ($f(x) = x$), but rather affine $f(x) = ax + b$.

Figure 7.9 illustrates the RE of the two energies. The figure shows that E_{acc} has a narrow RE compared to $E_{label,mc}$ but is not centered at 1, whereas $E_{label,mc}$ is.
Overall, $E_{label,mc}$ performs fairly well in reconstructing E_{truth}, as no shifts or biases are present. Although, if the distribution and behavior of E_{acc} is similar between MC and Data, it may also be interesting to use it for reconstruct $E_{truth,acc}$ in Data and investigate if that will yield more accurate results. However, using E_{acc} may be a problem as the E_{acc} is a direct result of the ECAL images, hence a regression algorithm would merely sum the cells.

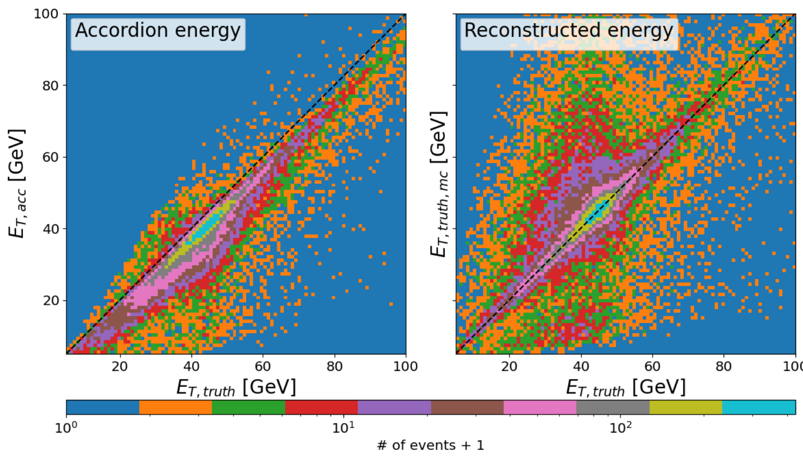


Figure 7.8: *Left figure* shows the correlation between E_{acc} and E_{truth}. *Right figure* shows the correlation between $E_{truht,mc}$ and E_{truth}. The figure compares the $E_{label,mc}$ and E_{acc} to E_{truth} to evaluate the performance of the reconstruction of E_{truth}. Note that the color scale is logarithmic.

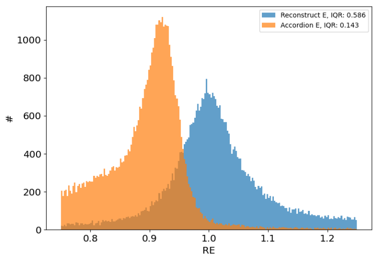


Figure 7.9: The figure shows the RE of the $E_{label,mc}$ and E_{acc}, with the IQR_{75} in the legend.

7.3.1 Variable distributions of $DeepCalo_{Zee,mc}$ outliers

From the correlation plot in figure 5, most events follow the dashed black line. However, the events far from the line are interesting, as they must have some behavior that make them difficult to reconstruct. In figure 9.26 and 9.27, the variable distribution of the outliers and the non-outliers[2] are illustrated. They can indicate if there are any systematic outliers.

[2] The events labeled as outliers can be seen in figure 7.10.

From η and η_{index} variables, we can see that many of the poorly reconstructed event are within the crack region, as expected. $DeepCalo_{Zee,mc}$ also has problems with high n_{track} events, which may indicate that additional track information is needed.

Looking at the long tails of $f0_{cluster}$ and $R12$, the reconstruction is also highly affected if the showers begin before or in the pre-sampler or if the energy is deposit in the tile-gap (see E_{TG3} and $E_{tile-gap}$).

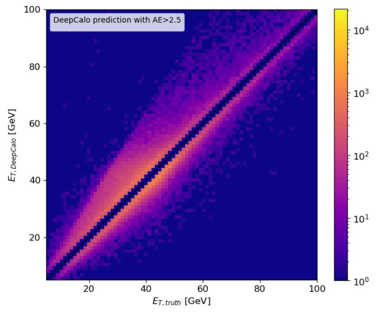


Figure 7.10: Correlation plot of $DeepCalo_{Zee,mc}$, where all events with an absolute error (AE) below 2.5 have been removed. The color map was changed so it was easier to see which events were deemed outliers.

7.4 Improving DeepCalo performance

When operating many different input types, the high customization of a NN comes in handy. However, finding the optimal architecture is very time consuming due to the possible permutations of the network. In the following section, we will go through some of the additional changes that may improve the performance but were unfortunately not tested.

Variables

Out of the 16 scalars variables, none of them are specialized for the photon. Most notably, the conversion properties of converged photons that are used in $Ecalib^{(BDT)}$ were not a part of the scalar variables. These will most definitely improve photon reconstruction. Scalar variables that must be tested are listed below.

- For charged particles, *dPOverP* ($\Delta p/p$): Is the momentum lost by tracks between the last measurement point and the perigee, all divided by the momentum at perigee. This will give an estimate of the lost momentum of the track.
- For photons, *ConvType* (*single/double*): Indicates if the tracks from the electron pair of a converted photon can be seen as a single or double track in the ECAL[3].
- For photons, *isConv*/γ*Conv*: Indicates if the photon was converted or not.
- For photons, r^{Conv}: This is the estimated radius of the conversion of a photon in the transverse plane and should indicate when in the detector the photons are converted.
- For all, $\langle\mu\rangle$: Pileup present in the event must also be added.

[3] If the electron pair has equal amounts of energy, the magnetic field in the ID is not able to separate the tracks before entering the ECAL. If the energy is split 80%/20%, both tracks can be seen in the ECAL, but if the ratio is 99%/1%, the small energy electron will curl and will not enter the ECAL while the other one will.

The three photon variables should help *DeepCalo* reconstruct converted photons, which likely will improve the photon reconstruction at $|\eta|$ beyond the crack.

In addition, the geometric scalar variables that are dependent on η could be transformed to use $|\eta|$ instead, as the detector is symmetric. This will double the amount of statistics in $|\eta|$ bins when training the model. For electrons, all the geometric track variables could also be transformed by subtracting the reconstructed track from the electron. However, this cannot be applied when training an ensemble model for both electron and photons, as this cannot be done for photons.
It is difficult to know if a variable will improve the performance or not. When training on MC and testing on Data, we need to ensure that additional variables do not add to the discrepancy between MC and Data.

Architecture

The additional module connections is missing in table 4.4 and could be tested, as some of them might yield improved performance.

The implementation of the ECAL time images has not been thoroughly investigated and should be implemented differently in order to take full advantage of the information. Instead of concatenating them to the energy image, we would have liked to implement *sigmoidal gating* on the energy image using the time image. Using a *gating* algorithm, the network would be able to re-weight individual pixels in the energy images and be a part of the back propagation so that the network also learns where to turn them off.

7.4.1 *Minimizing discrepancies between MC and Data*

Minimizing discrepancies between MC and Data will improve the performance gap between the MC and Data performance measure we saw in chapter 5 for the MC trained model. There are visible discrepancies in $1D$ seen in figures 3.3 and 3.4. They could be corrected for using histogram equalization on the MC variables to align them with Data.

Histogram equalization of MC

Histogram equalization is widely used in gray-scale image processing to equalize the light in a gray-scale image. Given two CDFs, one CDF can be mapped to another, which makes the two distribution share a PDF. Using histogram equalization, we can map MC histograms to its Data counterparts, which might help to improve the accuracy of MC trained *DeepCalo* on Data. In figure 9.4, histogram equalization has been applied to the MC distribution of $Z \rightarrow ee$ scalar variables (see table 3.3) and it has been mapped to $Z \rightarrow ee$ data. While most of the distributions between MC and data are not aligned, the transformed MC ($\mathrm{MC}_{\mathrm{trans}}$) is much more aligned with the Data in 1D, but testing the transformed variables in *DeepCalo* is required.

Auto-encoder

Notice that histogram equalization only transforms variables in 1D, yet, most discrepancies are in higher dimensions. An auto-encoder could be used to account for higher order discrepancies. Auto-encoders will not be explained in detail, but they essentially map variables unto themselves using a NN or a CNN. However, the output can be changed so the auto-encoder maps MC to Data and this may help account for the inconsistencies between MC and Data.

In future projects it may be beneficial to look at the discrepancies between MC and Data and attempt to correct them using an auto-encoder.

7.4.2 *Extrapolate between $Z \to \mu\mu\gamma$ energy ranges*

As seen in table 3.1, *Rucio* only contained $Z \to \mu\mu\gamma$ files with $E_T = [10, 35]$ and $[70, 140]$, not spanning the entire energy range. In figure 7.11, we can see the distribution of $E\text{calib}^{(BDT)}$ in MC and Data. The MC samples are clearly missing energy ranges compared to Data and the reconstruction of $Z \to \mu\mu\gamma$ will most certainly benefit from extending the ranges in MC, as we have Data events with energies $E_T > 40$ GeV but not in MC. However, additional files of $Z \to \mu\mu\gamma$ were not available on *Rucio*.

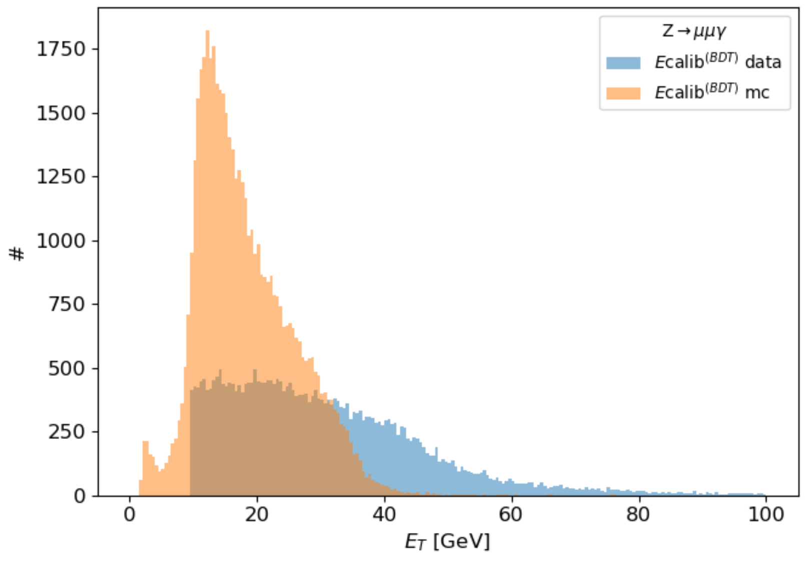


Figure 7.11: The figure shows the distribution of $E\text{calib}^{(BDT)}$in the Data used in this Thesis. The vertical lines shows the intervals where we have MC samples.

7.5 *Material budget*

In the reconstruction of photons (see chapter 6.), a general increase in performance was measured when using $DeepCalo_{Z\mu\mu\gamma,mc}$. However, this increase in performance vanishes beyond the crack region ($|\eta| > 1.4$), which poses a problem from the general performance of the photon reconstruction. From figure 7.12, we see the amount of

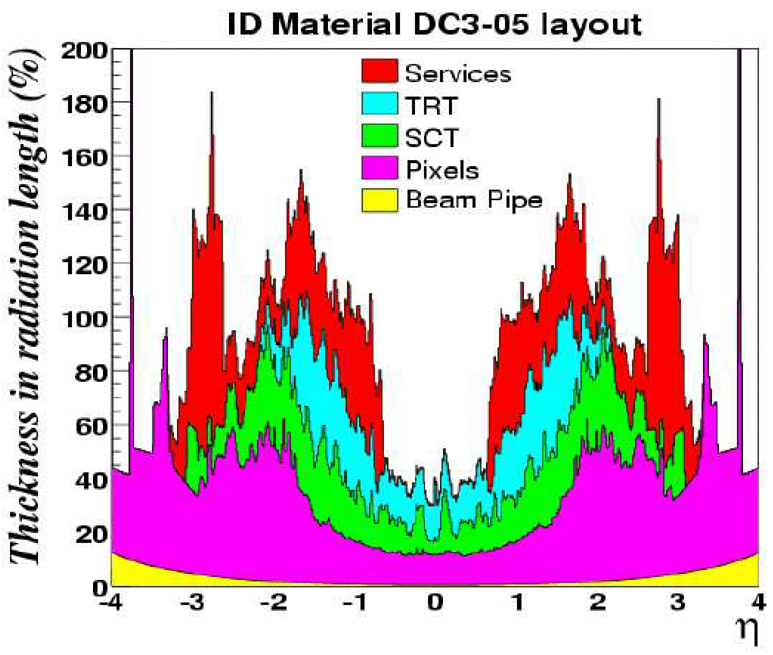


Figure 7.12: Figure from [4]. It shows the material budget of the ID with the color-code indicating the detector type of the material. While it extends to the forward region $\eta = [-4, 4]$, we are only looking at $\eta = [-2.5, 2.5]$.

material in the ID that the particles pass through. At $|\eta| > 1.4$, we see a general increase of material, especially in the Pixel and SCT layers. This might lead the photons to deposit more of their energy, and at high $|\eta|$, the conversion variables for photons will have a greater effect due to increased travel length.

Additional testing of the photon reconstruction is also needed. The optimizing and modeling of *DeepCalo* has been focused on electron reconstruction for $Z \to ee$, so a thorough analysis of the *DeepCalo*s behavior for photon reconstruction could increase performance too.

Performance measure at extended $\langle \mu \rangle$

Due to the upgrades to HL-LHC, the pileup of pp collisions will rise. We shown the pileup performance of *DeepCalo* in all the decay channels tested in this thesis. Nevertheless, the pileup range in the MC sample only extends to 80, so performance at very high pileup is still unknown. However, if it follows the trend shown in this thesis, *DeepCalo* will be much more accurate than $E\text{calib}^{(BDT)}$ at high pileup.

8 Conclusion

The aim of this thesis was to improve the energy reconstruction of electrons and photon in the channels seen in table 1.3. This improvement should be achieved using a convolutional neural network architecture developed in the thesis called *DeepCalo* which took advantage of the electromagnetic calorimeter images. The accuracy of *DeepCalo* was tested on Monte Carlo simulated data (MC) and real world Data, where E_{truth} was used to evaluate performance in MC, and the σ_{CB} of a Breit-Wigner convoluted with a Crystal Ball function (BW$\otimes$CB) was used to evaluate in Data.

Electron reconstruction

For the electron reconstruction, $Z \to ee$ was tested on MC and Data. The $DeepCalo_{Zee,mc}$ model had a performance gain of $\langle reIQR_{75}^{DeepCalo} \rangle = 22.4 \pm 0.7\%$ and $\langle reIQR_{95}^{DeepCalo} \rangle = 25.9 \pm 0.9\%$ over the $E\text{calib}^{(BDT)}$ on MC. Using the BW$\otimes$CB fit on MC, $DeepCalo_{Zee,mc}$ had a $23.5 \pm 0.4\%$ performance increase over $E\text{calib}^{(BDT)}$following the previous results. However, when used on Data, it achieved a performance increase of $9.4 \pm 0.9\%$. The model achieved a uniform performance increase over E_T, $|\eta|$ and $\langle \mu \rangle$ on MC and Data.

The $DeepCalo_{Zee,data}$ model used a method developed in this thesis that reconstructed an approximation of $E_{truth,data}$ for Data. The $E_{truth,data}$ was used to train *DeepCalo* directly on Data. It obtained a performance increase of $5.9 \pm 0.9\%$ compared with $E\text{calib}^{(BDT)}$. Further, it was able to center the invariant mass at the Z boson mass, which $DeepCalo_{Zee,mc}$ on Data was not.

This led to the construction of the $DeepCalo_{Zee,mc/data}$ model (the ensemble model), where training of the two previous models was combined so that the model was trained on both MC and Data. For MC, the model achieved an improvement of $\langle reIQR_{75}^{DeepCalo} \rangle = 22.1 \pm 0.3\%$ and $\langle reIQR_{95}^{DeepCalo} \rangle = 25.3 \pm 0.8\%$, similar to $DeepCalo_{Zee,mc}$. For Data, the model obtained a high performance increase of $18.3 \pm 0.8\%$ compared with $E\text{calib}^{(BDT)}$.

DeepCalo was also tested on an Electron-Gun sample without any decay channel. Here, *DeepCalo* achieved a performance gain over $E\text{calib}^{(BDT)}$of $\langle reIQR_{75}^{DeepCalo} \rangle = 6.811 \pm 0.077\%$ and $\langle reIQR_{95}^{DeepCalo} \rangle = 13.81 \pm 0.12\%$. The performance over $|\eta|$ and E_T was also tested, although it was difficult to draw a conclusion as there were fluctuations in the performance. Nevertheless, a small gain was visible.

Photon reconstruction

For the photon reconstruction, $H \to \gamma\gamma$ was tested on MC and $Z \to \mu\mu\gamma$ on MC and Data. The same extensive review of the different model types was not done for the photon reconstruction, but can easily be extended to the photon reconstruction. Only the performance of the MC models were shown.

For the $DeepCalo_{Hyy,mc}$ model, it achieved $\langle reIQR_{75}^{DeepCalo} \rangle = 12.8 \pm 0.3\%$ and $\langle reIQR_{95}^{DeepCalo} \rangle = 12.4 \pm 0.4\%$ with a uniform performance increase over $\langle \mu \rangle$ and E_T for MC. However, the $DeepCalo_{Hyy,mc}$ model was not able to accurately reconstruct events beyond the crack region $|\eta| < 1.4$. This is liekly due to the photon conversion at higher $|\eta|$, which is not a part of the variables for *DeepCalo*.

The $DeepCalo_{Z\mu\mu\gamma,mc}$ model was able to reconstruction $\langle reIQR_{75}^{DeepCalo} \rangle = 18.0 \pm 0.8\%$ and $\langle reIQR_{95}^{DeepCalo} \rangle = 18.3 \pm 1.4\%$ better than $E\text{calib}^{(BDT)}$for MC. It suffers from the same behavior as the $DeepCalo_{Hyy,mc}$ with poorly reconstruction of photons beyond the crack region. The model was also tested using the BW$\otimes$CB fit on MC. Here, it achieved a performance gain of $7.9 \pm 0.8\%$. The small increase in σ_{CB} is due to the $Z \to \mu\mu\gamma$ being dominated by the energy resolution of the muons. Using the BW$\otimes$CB fit on Data, the model had a performance gain of 2.4 ± 1.1. Further, this performance gain was uniform over $|\eta|$ and did not suffer beyond the crack region in Data.

Additional developments and summary

Using *DeepCalo*, it is possible to predict an additional uncertainty to the energy prediction by changing the loss function of the architecture to equation 4.3. The results from $DeepCalo_{\sigma}$ can be seen in figure 5.14, where σ can be used to select events with a high precision in MC and Data. The pull residuals in figure 5.13 show that the predicted uncertainties are about 4.5 times too large, as the width of the Gaussains are $\sigma \approx 0.2$.
A framework was also developed that creates *Python* ready files from the DxAOD format containing the ECAL imagery.

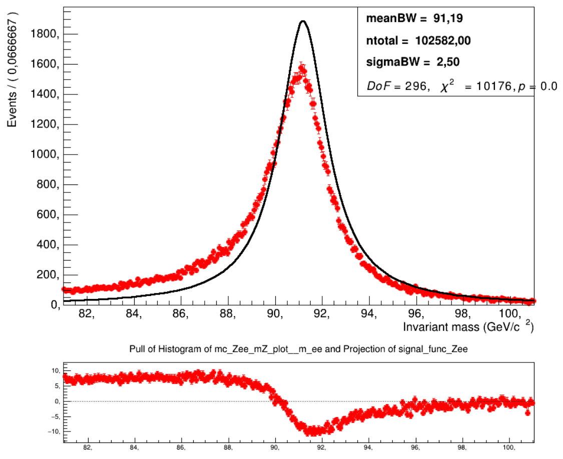
Table 8.1 shows a summary of the improvements achieved by *DeepCalo*. There are significant performance improvements throughout the tested samples, when comparing to the $E\text{calib}^{(BDT)}$ by ATLAS, that has been developed over many years. Although performance of *DeepCalo* on Data is not as impressive, still better. It can surely be improved, if the four corrections applied to $E\text{calib}^{(BDT)}$ were applied as well to *DeepCalo*. With the assistance from within ATLAS (in particular Debottam B. Gupta), the *DeepCalo* architectures is being implemented in the ATLAS software, and it will be interesting to see, if the algorithm or further developments of it at some point becomes used in public ATLAS results. In particular the good performance despite higher pileup might be the key to convincing others of the merits of the *DeepCalo* approach along with some of the other innovations of this master thesis.

		Monte Carlo		Data
Channel	Model	IQR$_{75}$	σ_{CB}	σ_{CB}
$Z \to ee$	$DeepCalo_{Zee,mc}$	$22.4 \pm 0.7\%$	$23.5 \pm 0.4\%$	$9.4 \pm 0.9\%$
	$DeepCalo_{Zee,data}$			$5.9 \pm 0.9\%$
	$DeepCalo_{Zee,mc/data}$	$22.1 \pm 0.3\%$	$23.3 \pm 0.4\% *$	$18.3 \pm 0.8\%$
	$DeepCalo_{\sigma}$	$17.4 \pm 1.5\%$	$9.0 \pm 0.9\% *$	$6.4 \pm 1.0\% *$
EG	$DeepCalo_{EG}$	$6.81 \pm 0.08\%$		
$H \to \gamma\gamma$	$DeepCalo_{Hyy,mc}$	$12.8 \pm 0.3\%$	$5.95 \pm 1.19\% *$	
$Z \to \mu\mu\gamma$	$DeepCalo_{Z\mu\mu\gamma,mc}$	$18.0 \pm 0.8\%$	$7.9 \pm 0.8\%$	$2.4 \pm 1.1\%$

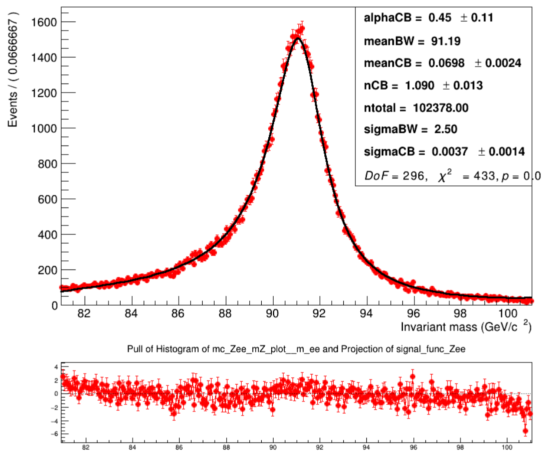
Table 8.1: Table showing the performance of all the *DeepCalo* models. $*$ indicates performance measures that was not shown in the analysis chapter, but figures can be found in the appendix.

9 Appendix

9.1 Proof of concept



(a) Here, a Breit-Wigner distribtuion has been fitted to the resonance peak of the Z. Both σ_{BW} and μ_{BW} have been set to constants at $\sigma_{BW} = 2.4952$ and $\mu_{BW} = 91.1876$, which are their theoretical values.



(b) Here, a Breit-Wigner convoluted with a Crystal Ball has been fitted to the same as in (a). Again, σ_{BW} and μ_{BW} are constant, but the CB parameters are floating.

Figure 9.1: In this figure, the effects of the convolution between BW and CB can be seen. Both are fits on the same Monte Carlo data, using its truth parameter with decay channel $Z \to ee$.

9.1.1 Crystal Ball function

Crystal ball (CB) is a Gaussian function with a power-law as a low-end tail at some threshold α. It is often used in high-energy physics with a BW to explain the resonance peaks. CB is given by a fork function, where it changes to a power law at some α

$$f(x;\alpha,n,\bar{x},\sigma) = N \cdot \begin{cases} exp\left(-\frac{(x-\bar{x})^2}{2\sigma^2}\right), & \text{for} \frac{(x-\bar{x})}{\sigma} > -\alpha \\ A \cdot (B \cdot \frac{x-\bar{x}}{\sigma}^{-n}), & \text{for} \frac{(x-\bar{x})}{\sigma} \leq -\alpha \end{cases}, \tag{9.1}$$

Where the parameters A, B, C, D and N depend on the fitting parameters

$$
\begin{aligned}
A &= \left(\frac{n}{|\alpha|}\right)^{n} \cdot \exp\left(-\frac{|\alpha|^2}{2}\right), \\
B &= \frac{n}{|\alpha|} - |\alpha|, \\
C &= \frac{n}{|\alpha|} \cdot \frac{1}{n-1} \cdot \exp\left(-\frac{|\alpha|^2}{2}\right), \\
D &= \sqrt{\frac{\pi}{2}} \left(1 + \text{erf}\left(\frac{|\alpha|}{\sqrt{2}}\right)\right), \\
N &= \frac{1}{\sigma \cdot (C + D)}
\end{aligned}
$$

Figure 9.1 displays the effects of the convolution with CB and illustrates the shortcoming of a single BW, as it is not able to describe Z-boson resonance. Note that in figure 9.1 (b) both σ_{CB} and μ_{CB} are small, meaning most of the peak and width is described by the BW. Thus, CB is mostly used to describe the tails of the distribution, and σ_{CB} and μ_{CB} could be set to zero without changing the fit.

9.1.2 *Calculating the uncertainties*

To use the uncertainties from the BW⊗CB fit, we will use weighted mean and its uncertainty to compute the average of the σ_{CB} and μ_{CB}. The expression used can be seen in equation 9.2, where δ indicates the error of the parameter.

$$
\begin{aligned}
\overline{x} &= \frac{\sum_{i=1}^{n} \frac{x_i}{\delta x_i}}{\sum_{i=1}^{n} \frac{1}{\delta x_i}} \\
\delta \overline{x} &= \sqrt{\frac{1}{\sum_{i=1}^{n} \frac{1}{\delta x_i}}}
\end{aligned}
\tag{9.2}
$$

If an expression uses a parameter with uncertainties, *propagation of errors* will be used to calculate the complete uncertainty.

Uncertainty on the invariant mass

Using equation 1.1 and propagation of errors, the uncertainty on the invariant mass of a two particle decay can be calculated. Starting at

$$
M^2 = 2\frac{E_1 \cdot E_2}{\cosh(\eta_1) \cdot \cosh(\eta_2)} (\cosh(\eta_1 - \eta_2) - \cos(\phi_1 - \phi_2)) \tag{9.3}
$$

redefining the invariant mass to

$$
M = \sqrt{E_1 \cdot E_2 \cdot k}, \qquad k = 2\frac{(\cosh(\eta_1 - \eta_2) - \cos(\phi_1 - \phi_2))}{\cosh(\eta_1) \cdot \cosh(\eta_2)} \tag{9.4}
$$

using propagation of errors

$$\begin{aligned}\sigma_M^2 &= \left(\frac{\partial M}{\partial E_1}\sigma_{E_1}\right)^2 + \left(\frac{\partial M}{\partial E_2}\sigma_{E_2}\right)^2 \\ &= \left(\frac{E_2 \cdot k}{\sqrt{E_1 \cdot E_2 \cdot k}} \cdot \sigma_{E_1}\right)^2 + \left(\frac{E_1 \cdot k}{\sqrt{E_1 \cdot E_2 \cdot k}} \cdot \sigma_{E_2}\right)^2 \\ &= \left(\frac{k}{\sqrt{E_1 \cdot E_2 \cdot k}}\right)^2 \cdot (E_2^2\sigma_{E_1}^2 + E_1^2\sigma_{E_2}^2).\end{aligned} \tag{9.5}$$

Thus, the uncertainty on the invariant mass of a two particle decay is

$$\sigma_M = \sqrt{\left(\frac{k}{\sqrt{E_1 \cdot E_2 \cdot k}}\right)^2 \cdot (E_2^2\sigma_{E_1}^2 + E_1^2\sigma_{E_2}^2)}. \tag{9.6}$$

9.2 *Eta-shift*

In section 5.6, *eta-shift* is applied to the resonance peak with the hope of a more narrow peak. It was achieved for MC but unfortunately not for Data. The idea behind shifting eta can be seen in figure 9.2. The resonance peak for different η bins might be shifted and this can result in a wider peak, as seen in figure 9.2.

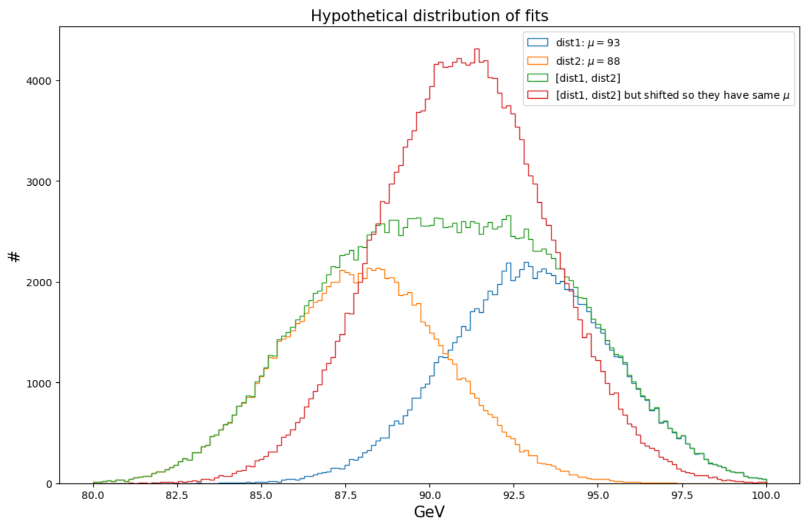


Figure 9.2: The figure shows three Gaussians that should indicate the $Z \rightarrow ee$ resonance. Dist1 and Dist2 should be in two different η bins, where their μ is shifted from the one of the Z.

Derivation of η shift

In the following equation, the deviation of the η correction can be seen.

$$\text{Invariant mass of massless particles: } m_{ij}^2 = 2E_iE_j \cdot (1 + cos(\theta))$$
$$\text{Mass correction: } E^{corr} = E_{pred} \cdot (1 + \alpha_i)$$

where i, j are the different η bins,

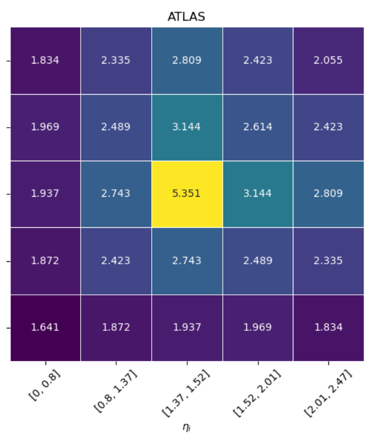
$$\begin{aligned}
m_Z^2 &= 2E_i^{corr}E_j^{corr} \cdot (1 + cos(\theta)) \Updownarrow \\
m_Z^2 &= 2E_i^{pred}E_j^{pred} \cdot (1 + cos(\theta)) \cdot (1+\alpha_i) \cdot (1+\alpha_j) \Updownarrow \\
m_Z^2 &= (m_{Z,ij}^{pred})^2 \cdot (1+\alpha_i) \cdot (1+\alpha_j) \Updownarrow \\
m_Z &= m_{Z,ij}^{pred} \cdot \sqrt{(1+\alpha_i+\alpha_j+\alpha_i\alpha_j)} \quad \alpha_i\alpha_j \approx 0 \Updownarrow \\
m_Z &= m_{Z,ij}^{pred} \cdot \sqrt{(1+\alpha_i+\alpha_j)}, \quad (1+x)^k = 1 + k \cdot x + ...(binomialseries) \Updownarrow \\
m_Z &= m_{Z,ij}^{pred} \cdot (1 + \frac{\alpha_i+\alpha_j}{2}) \Updownarrow \\
\mu_{ij} &= m_{Z,ij}^{pred} \cdot \frac{(\alpha_i+\alpha_j)}{2}, \quad \mu_{ij} = m_Z - m_{Z,ij}^{pred} \Updownarrow
\end{aligned} \tag{9.7}$$

μ_{ij} is the meanCB for each η fit We would then like to minimize:

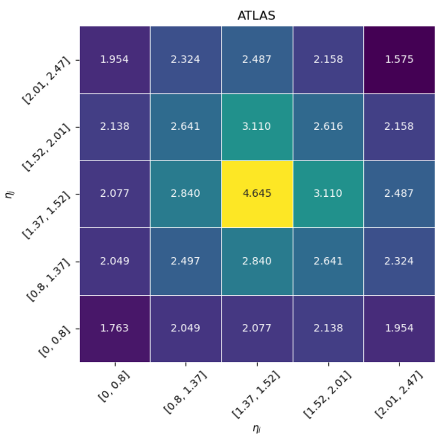
$$\chi^2(\alpha) = \sum_{(i,j)\in accpected} \left(\frac{\mu_{ij} - m_{Z,ij}^{pred} \cdot (\frac{\alpha_i+\alpha_j}{2})}{\sigma_{\mu_{i,j}}} \right)^2 \tag{9.8}$$

Equation 9.8 is the equation that is solved in chapter 5. The peak shift of Ecalib$^{(BDT)}$ can be seen in figure 9.3.

Figure 9.3: The figure shows the σ_{CB} from Ecalib$^{(BDT)}$ for both MC (figure a) and Data (figure (b)) in each $|\eta|$ bin. Its *DeepCalo* counterpart can be seen in figure 5.19. The number of events per bin can be seen in figure 5.15.



(a) σ_{CB} of the BW⊗CB fit in MC.



(b) σ_{CB} of the BW⊗CB fit in Data.

9.3 *Transformation of MC*

In figure 9.4, histogram equalization has been applied to the scalar variables of MC and mapped to the same variables of Data. This could help improve the misalignment between MC and Data and, as can be seen from the figure, the variables between MC and Data are better aligned.

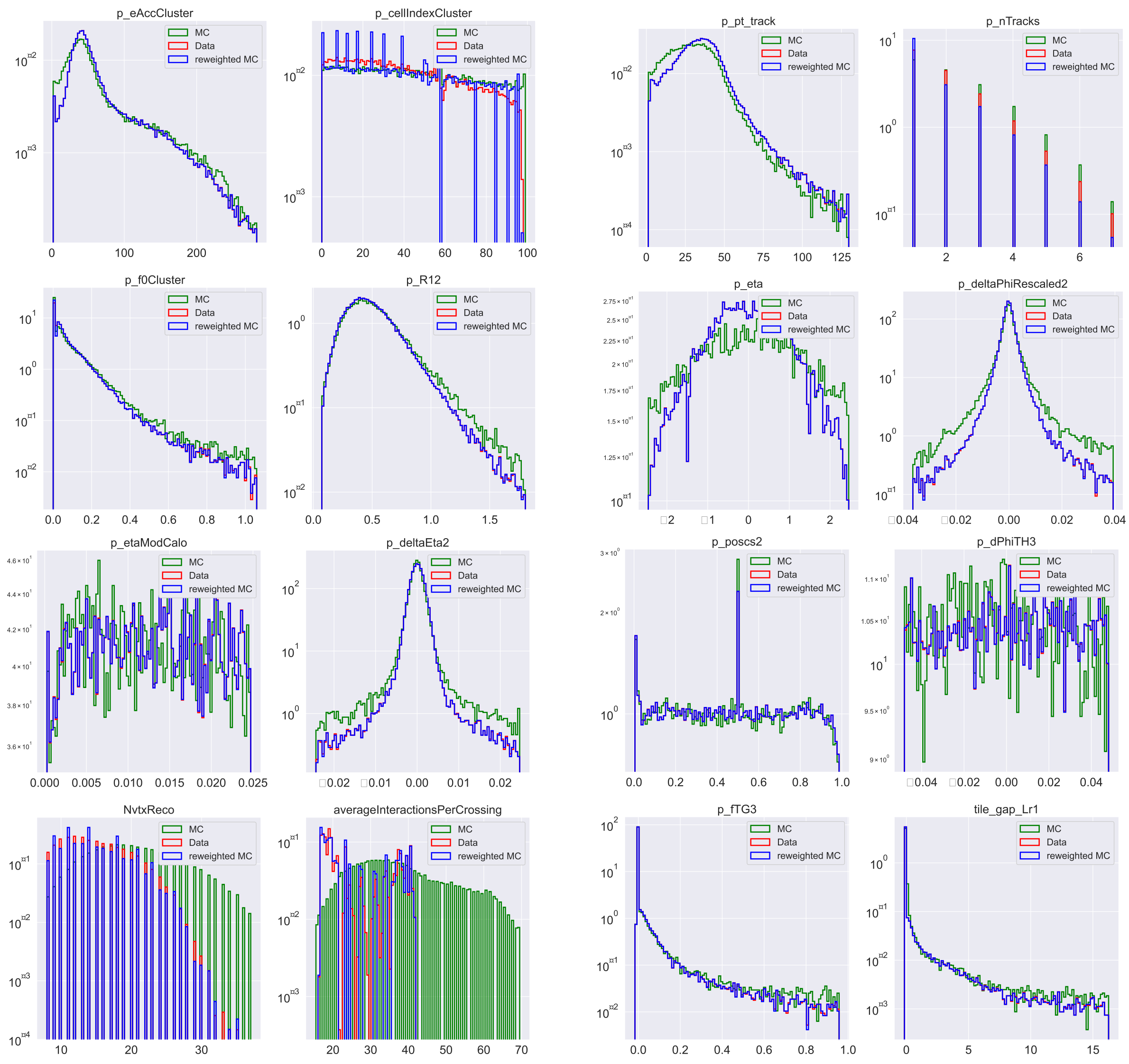


Figure 9.4: The scalar variables of MC have been transformed using histogram equalization such that their PDF is the same as for Data.

9.4 *Classifying the difference between MC and data*

In the ensemble method $DeepCalo_{Zee,mc/data}$, where we combine training of both MC and Data, for better generaliztion on Data, although without a indicator of the event were a MC event or a Data event. However, we can measure the distinction and this will give us an indicator of how well MC is modeled after Data. In figure 9.5, we have changed the loss function and the last activation layer of the network so it would be able to classify whether an event is MC or Data. To measure the performance, we used a receiver operating characteristic curve (ROC), which is a measure of true positive classification (TPR) compared to false positive classification (FPR). The area-under-curve (AUC) is then a direct measure of how well the algorithm performed. Looking at figure 9.5, we see that the $AUC = 0.82$, so the algorithm clearly distinguishes between MC and Data. However, as we saw from chapter 5 and 7, the increase of reconstruction performance and the correlation plot of MC agrees that some event are correctly reconstructed by creating $E_{label,mc}$.

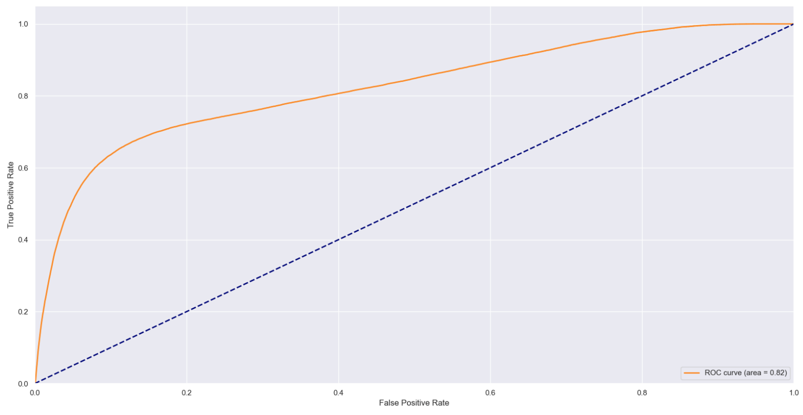


Figure 9.5: The ROC curve has the TPR on the y-axis and FPR on x-axis, with the AUC in the legend ($AUC = 0.82$).

9.5 *Analysis plots*

In the following section, some additional analysis figures have been added.

9.5.1 $Z \rightarrow ee$

In figure 9.6, we see the invariant mass from the energies reconstructed by $\text{Ecalib}^{(BDT)}$ and *DeepCal* as well as the E_{truth}. Thus, the resonance peaks can be compared visually. We see a clear performance increase from $\text{Ecalib}^{(BDT)}$ to *DeepCalo*, however, we also see that we still have a ways to go for estimating E_{truth}. In figure 9.7, we see the BW$\otimes$CB fit performance of the $DeepCalo_{\sigma}$ model.

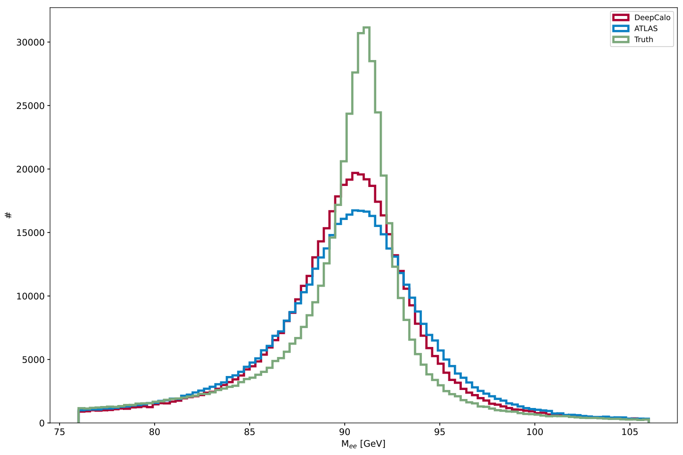
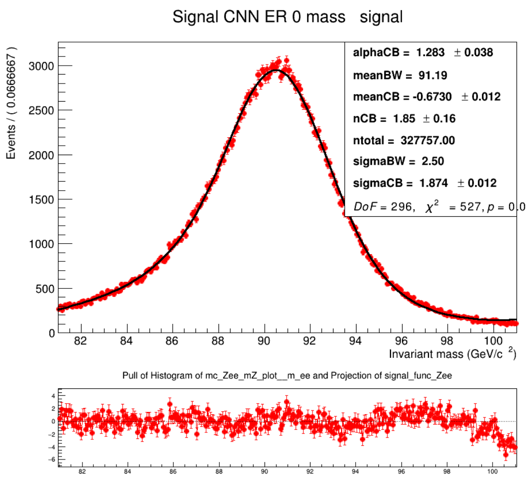
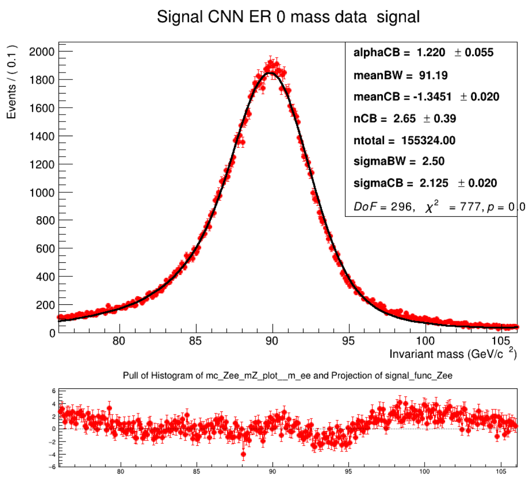


Figure 9.6: The $Z \rightarrow ee$ resonance peak for MC using E_{truth}, $E_{DeepCalo}$ and $E\text{calib}^{(BDT)}$ energies.



(a) $Z \rightarrow ee$ performance on MC for $DeepCalo_{\sigma}$.

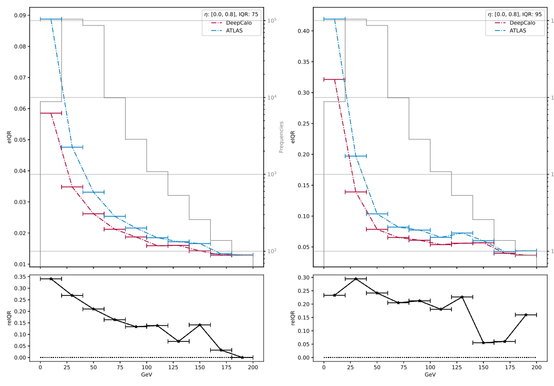


(b) $Z \rightarrow ee$ performance on Data for $DeepCalo_{\sigma}$.

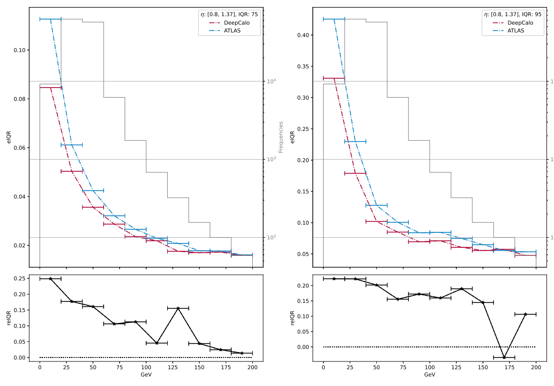
Figure 9.7: The figures show the BW$\otimes$CB fit performance of $DeepCalo_{\sigma}$ for both MC and Data

Zee $|\eta|$ *vs* E_T

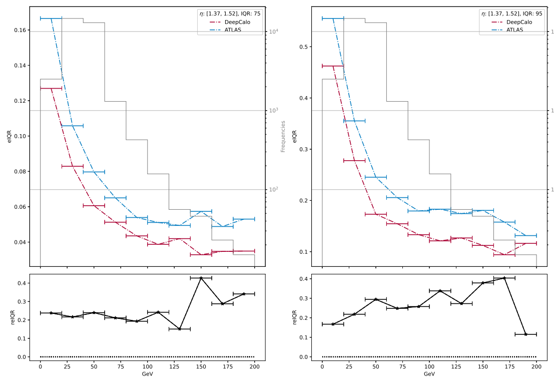
In figure 9.8, we see the *e*IQR and *re*IQR of different E_T and $|\eta|$ bins. It is meant to give more context to figure 5.2, as we see both the 75 and 95 *re*IQR and the number of events per bin shown by the gray histogram.



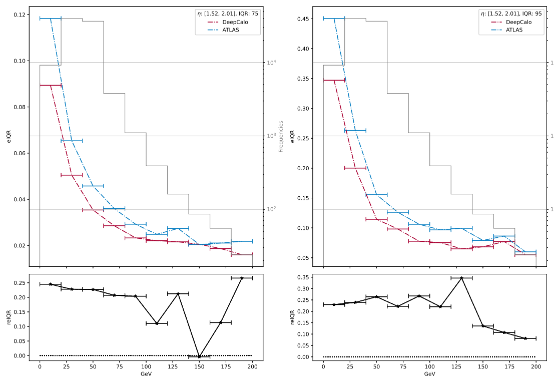
(a) $Z \to ee$ events with $|\eta| = [0.0, 0.8[$



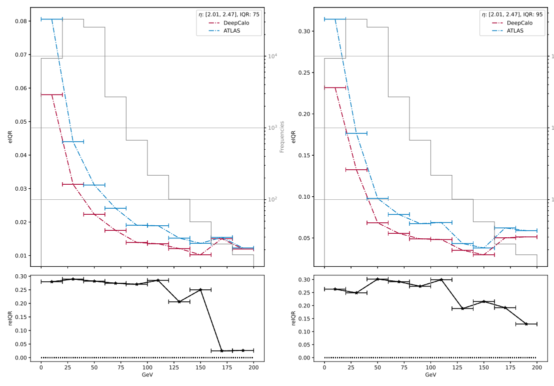
(b) $Z \to ee$ events with $|\eta| = [0.8, 1.37[$



(c) $Z \to ee$ events with $|\eta| = [1.37, 1.52[$



(d) $Z \to ee$ events with $|\eta| = [1.52, 2.01[$



(e) $Z \to ee$ events with $|\eta| = [2.01, 2.5[$

Figure 9.8: The figure shows the *e*IQR at 75 and 95 of *DeepCal* and Ecalib$^{(BDT)}$, as well as the compared performance *re*IQR between the two. It also shows the number of events in each of the $|\eta|$ vs E_T bins.

9.5.2 $Z \to ee$ *ensemble model*

A correlation plot without a logarithmic color scale of number of events can be seen in figure 9.9. Here the shift in E_{acc} is visible. The figure shows the correlation between $E_{label,mc}$ and E_{truth} The BW⊗CB fit of the $DeepCalo_{Zee,mc/data}$ model on MC can be seen in figure 9.10.

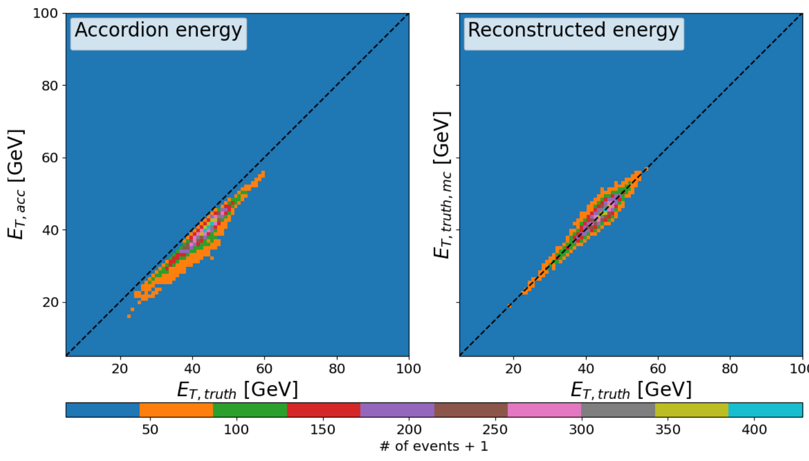


Figure 9.9: *Left figure* shows the correlation between E_{acc} and E_{truth}. *Right figure* shows the correlation between $E_{truht,mc}$ and E_{truth}. The figure compares the $E_{truth,mc}$ and E_{acc} to E_{truth} to evaluate the performance of the reconstruction of E_{truth}. Note that the color scale is logarithmic.

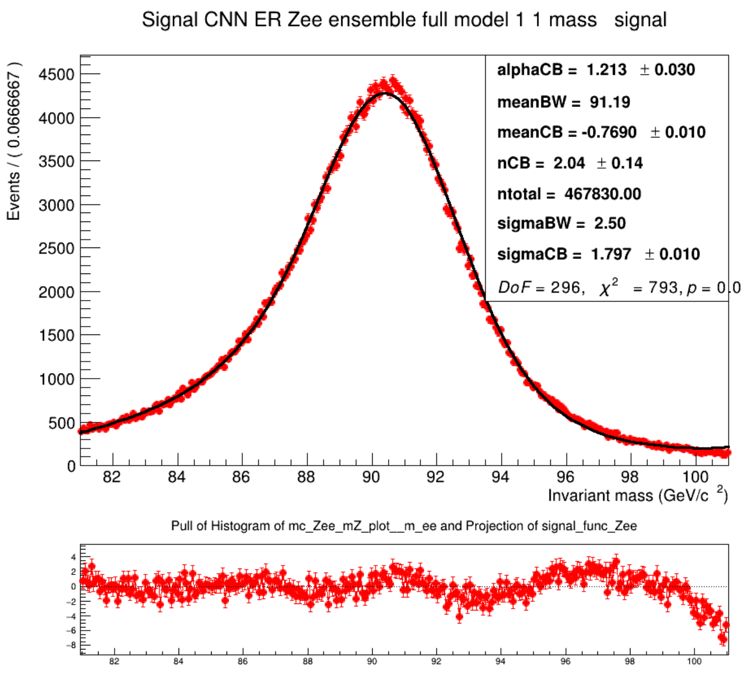


Figure 9.10: The fit shows the BW⊗CB fit on the $DeepCalo_{Zee,mc/data}$.

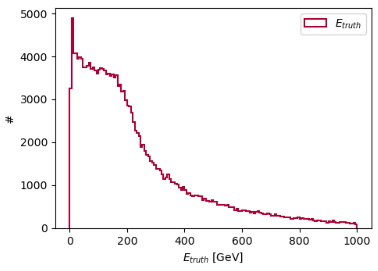
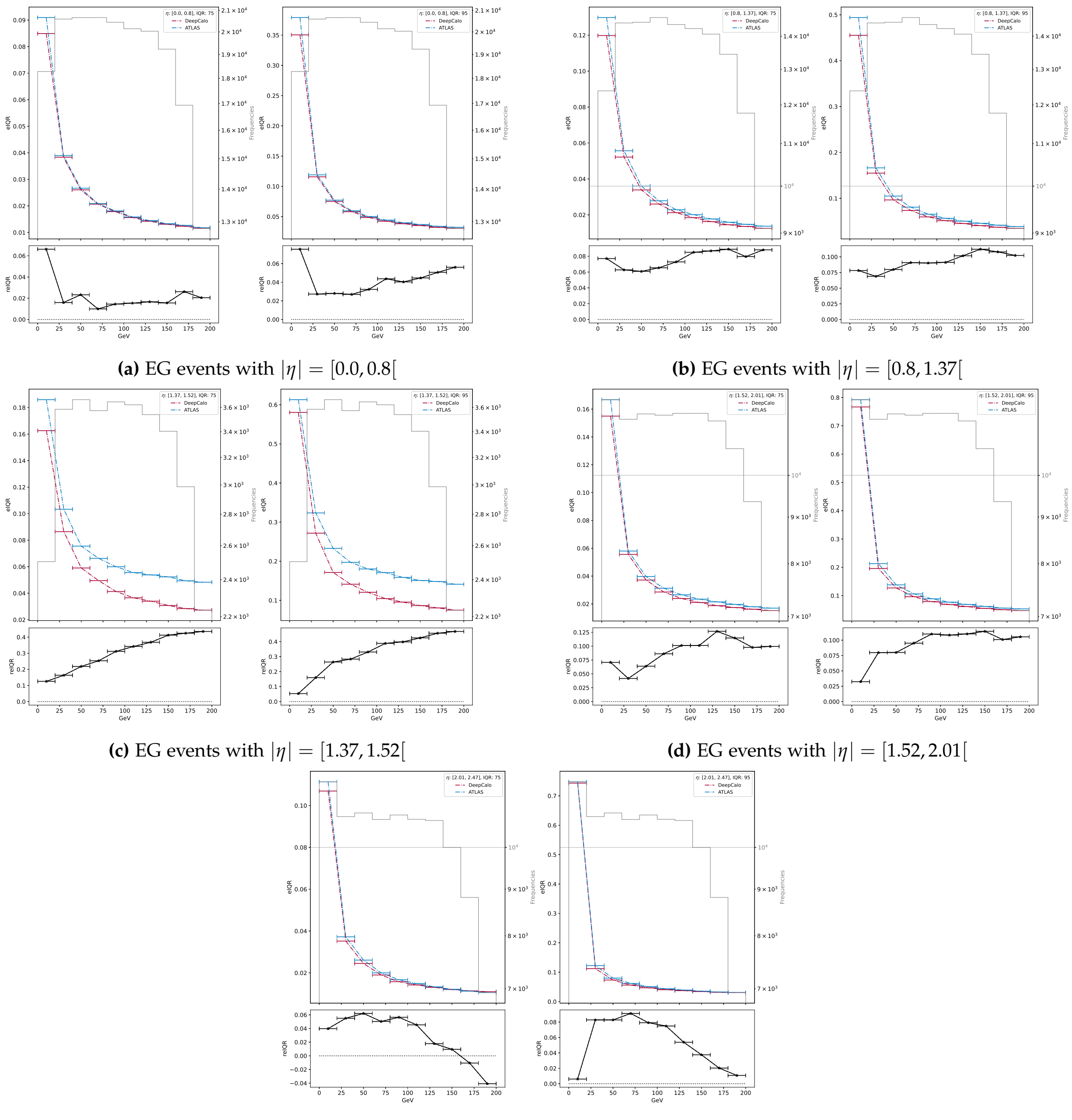


Figure 9.11: The figure shows the η and E_{truth} distribution for the EG.

Electron cannon

In figure 9.11, we see the distribution of E_{truth} for the EG samples. We see that the energy range is far larger compared to $Z \to ee$, which helps us test the generalization in energies for *DeepCalo*.

Figure 9.12 is the same as figure 9.8, but for the Electron Gun sample. As it is the EG sample, the E_T could have been extended, however, we still see some drops in the number of events at 200 GeV, so the upper E_T was kept the same as previously.



(a) EG events with $|\eta| = [0.0, 0.8[$

(b) EG events with $|\eta| = [0.8, 1.37[$

(c) EG events with $|\eta| = [1.37, 1.52[$

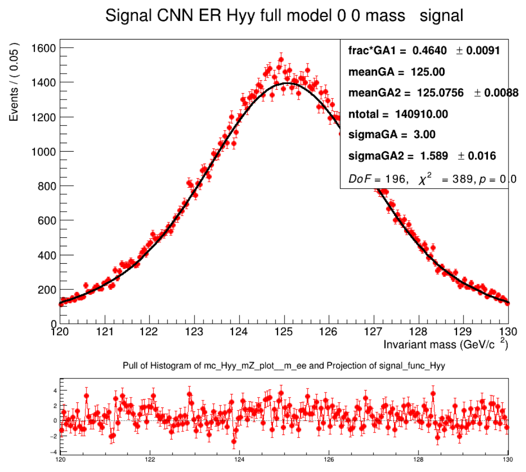
(d) EG events with $|\eta| = [1.52, 2.01[$

(e) EG events with $|\eta| = [2.01, 2.5[$

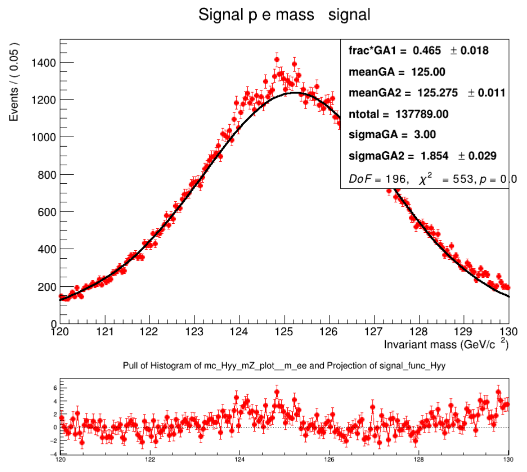
Figure 9.12: The figure shows the *e*IQR at 75 and 95 of *DeepCalo* and Ecalib$^{(BDT)}$ as well as the compared performance *re*IQR between the two for the EG sample. It also shows the number of events in each of the $|\eta|$ vs E_T bins.

9.5.3 $H \to \gamma\gamma$

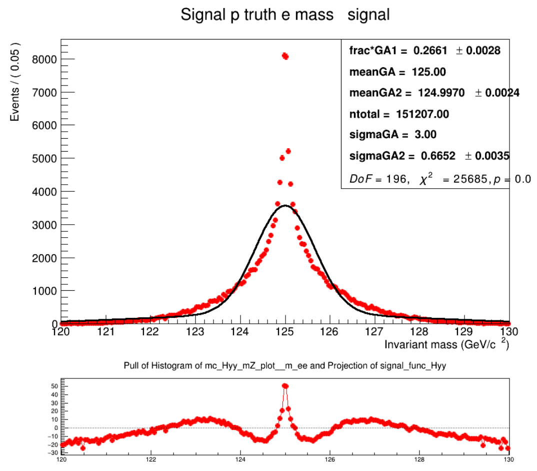
We have not shown any fits to the invariant mass of the Higgs boson due to no Data being available. Thus, measuring performance from the resonance fit was not important, however, the Higgs resonance does not follow a BW⊗CB fit either. It follows a Gaussian distribution. We have tested different fitting functions and found that two Gaussians will do the job for the reconstructed energy. The fitting attempts can be seen in figure 9.13. The invariant masses of



(a)



(b)



(c)

Figure 9.13: The figures shows a double Gaussian fit on the invariant mass for the $H \to \gamma\gamma$. The title of the figure indicate the reconstruction method used. p e is the Ecalib$^{(BDT)}$ (figure b), CNN is *DeepCalo* (figure a) and p truth e is the E_{truth} (figure c).

$H \to \gamma\gamma$ follows fairly well the double Gaussian. It is not able to fit to the invariant mass of E_{truth} as it only has a width of a few MeV (in figure 9.13, the reconstructed η and ϕ are used, which is why the invariant mass of E_{truth} has a width.).

In figure 9.14, we see the invariant mass from the energies reconstructed by Ecalib$^{(BDT)}$ and *DeepCal* for $H \to \gamma\gamma$. Thus, the resonance peak can be compared visually. We see a small performance increase from $\text{Ecalib}^{(BDT)}$ to *DeepCalo*, however, as seen from figure 9.13, there is still much to be gain to achieve reconstruction close to E_{truth}.

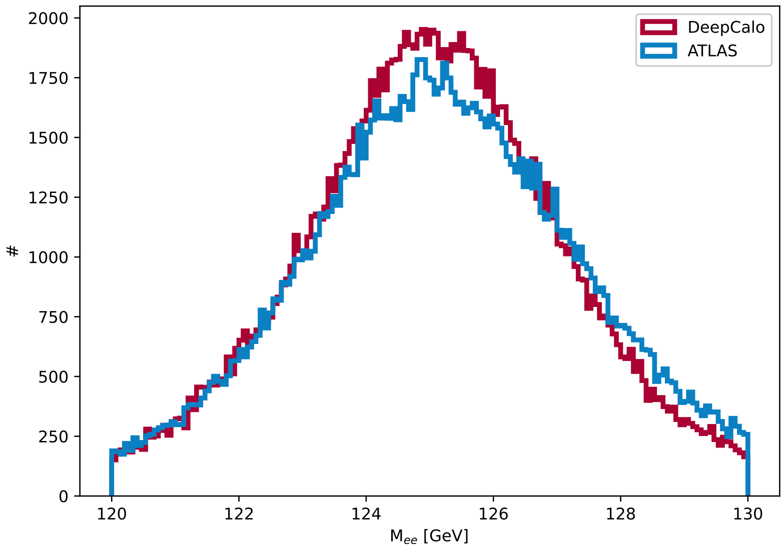
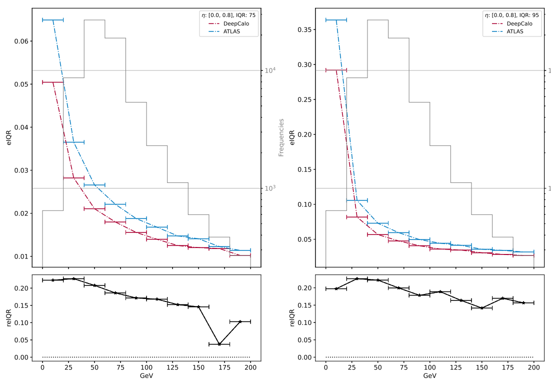
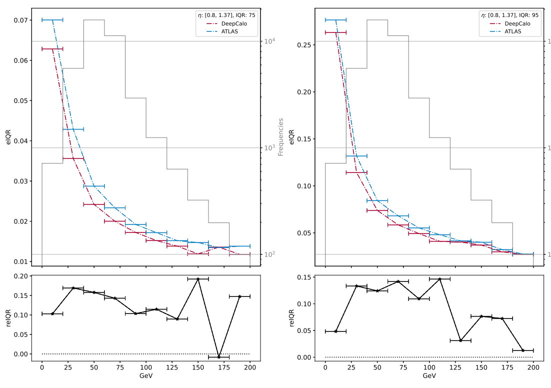


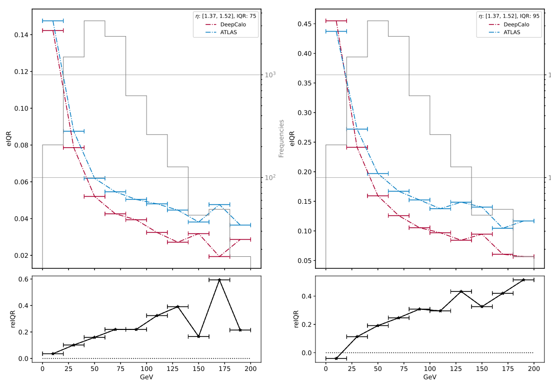
Figure 9.14: The $H \rightarrow \gamma\gamma$ resonance peak in MC using E_{truth}, $E_{DeepCalo}$ and $\text{Ecalib}^{(BDT)}$ energies.

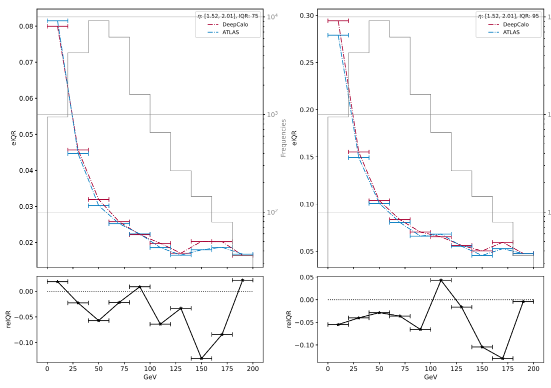
$|\eta|$ *vs* E_T

Figure 9.15 shows the $|\eta|$ vs E_T bins of the $H \rightarrow \gamma\gamma$ *DeepCalo* model. It is a more detailed depiction of figure 6.2, as the *e*IQR, *re*IQR for 75 and 95 and number of events can be seen in the figure.


(a) $H \rightarrow \gamma\gamma$ events with $|\eta| = [0.0, 0.8[$


(b) $H \rightarrow \gamma\gamma$ events with $|\eta| = [0.8, 1.37[$


(c) $H \rightarrow \gamma\gamma$ events with $|\eta| = [1.37, 1.52[$


(d) $H \rightarrow \gamma\gamma$ events with $|\eta| = [1.52, 2.01[$

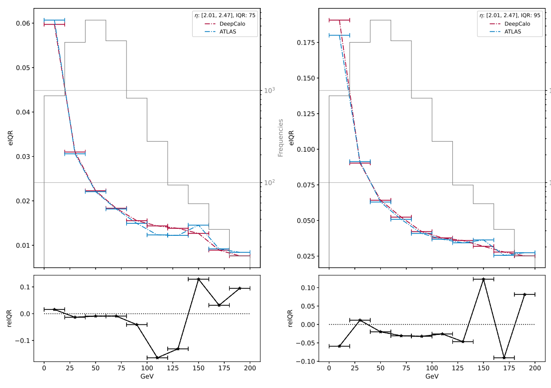

(e) $H \rightarrow \gamma\gamma$ events with $|\eta| = [2.01, 2.47[$

Figure 9.15: The figures show the performance in *DeepCalo* and $E\text{calib}^{(BDT)}$ in $|\eta|$ and E_T bins for $H \rightarrow \gamma\gamma$.

9.5.4 $Z \rightarrow \mu\mu\gamma$

In figure 9.16, we can see the comparison between the invariant mass distribution between *DeepCalo* and $E\text{calib}^{(BDT)}$. As seen from the figure, we only see a small increase in performance using *DeepCalo* compared to $E\text{calib}^{(BDT)}$. However, we also see that both reconstruction

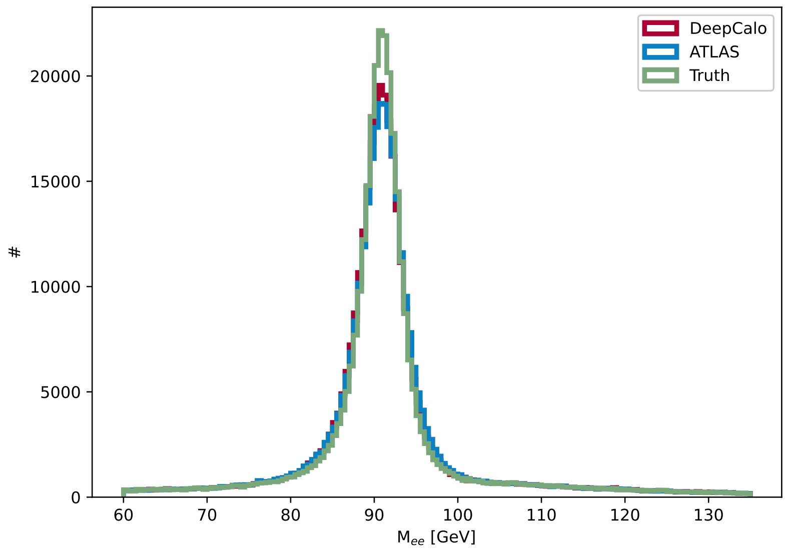


Figure 9.16: The $Z \rightarrow \mu\mu\gamma$ resonance peak in MC using E_{truth}, $E_{DeepCalo}$ and $E\text{calib}^{(BDT)}$ energies with reconstructed η and ϕ.

methods are not far from E_{truth} compared to $H \rightarrow \gamma\gamma$ and $Z \rightarrow ee$. This is due to the $Z \rightarrow \mu\mu\gamma$ channel being dominated by the accuracy in the μ reconstruction, as seen in section 9.5.4.

Muon domination in $Z \rightarrow \mu\mu\gamma$

In $Z \rightarrow \mu\mu\gamma$, the invariant mass is highly dominated by the reconstructed energy of the two muons. How much it is dominated cannot be measured for Data, however, we can measure it for MC by using $E_{truth,\gamma}$ and $E\text{calib}^{(BDT)}$ for the two muons. This has been done in figure 9.17, where we can see, by using $E_{truth,\gamma}$, that the σ_{CB} of the fit is still at $\sigma_{CB} = 1.269 \pm 0.012$. Thus, the energy resolution of the muons still has a significant impact on the width.

$|\eta|$ *vs* E_T

In figure 9.18, we again show the $|\eta|$ vs E_T bins, but this time for $Z \rightarrow \mu\mu\gamma$ of the *DeepCalo* model. It is a more detailed depiction of figure 6.7, as the *e*IQR, *re*IQR for 75 and 95 and number of events can be seen in the figure.

MC $|\eta|$ *performance*

The evaluation of pileup performance can be seen in figure 9.19, where the bottom plots display the *re*IQR comparing *DeepCalo* and $E\text{calib}^{(BDT)}$. The *re*IQR is steadily increasing with pileup. Thus, fol-

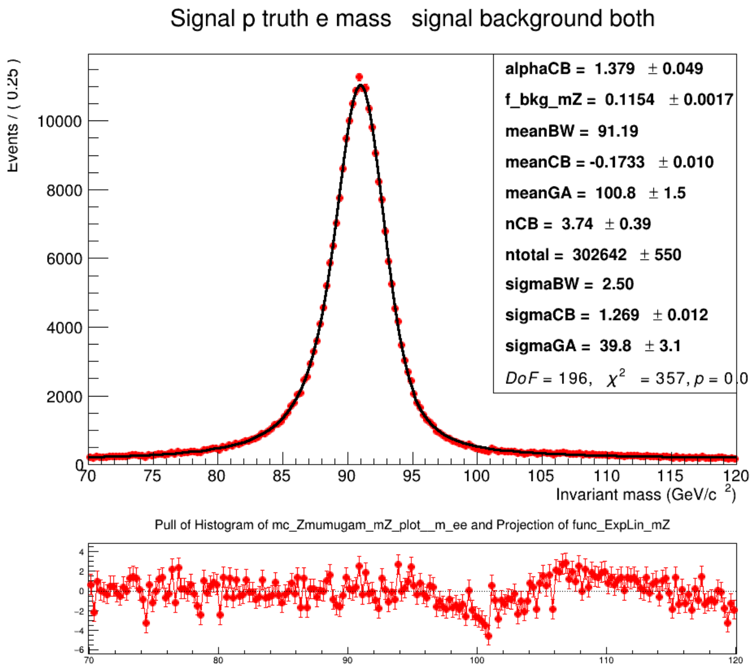
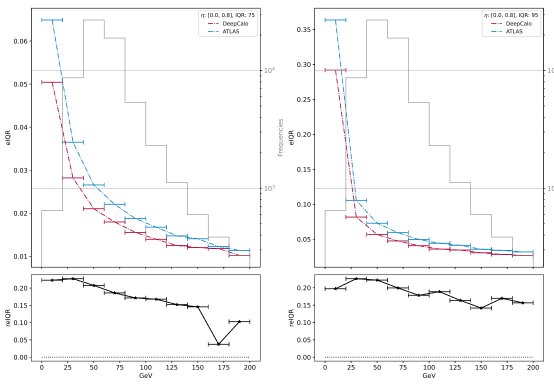
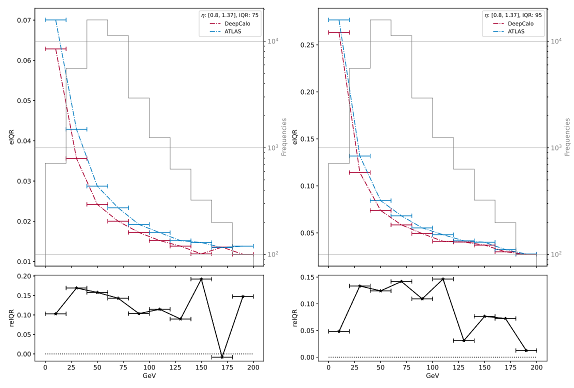


Figure 9.17: The figure shows the BW$\otimes$CB fit to $Z \rightarrow \mu\mu\gamma$, where $E_{truth,\gamma}$ has been used for the photon and $E\text{calib}^{(BDT)}$ for the two muons.

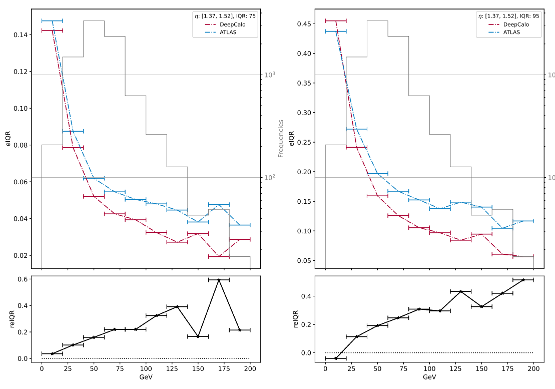
lowing the trend from the previous models, *DeepCalo* is not as affected by increasing pileup, as $E\text{calib}^{(BDT)}$is.



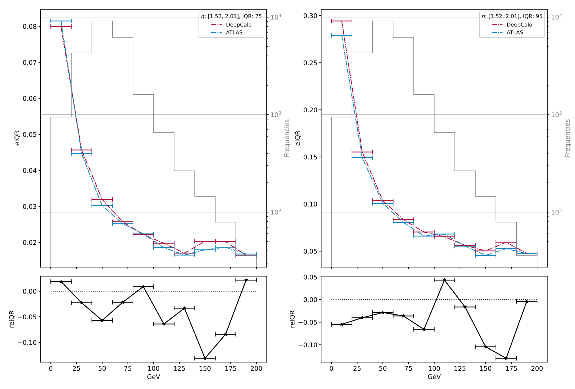
(a) $Z \to \mu\mu\gamma$ events with $|\eta| = [0.0, 0.8[$



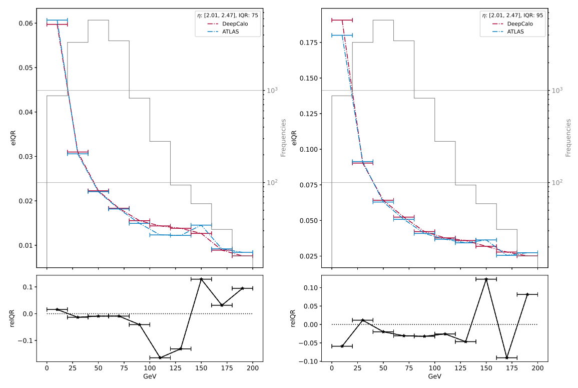
(b) $Z \to \mu\mu\gamma$ events with $|\eta| = [0.8, 1.37[$



(c) $H \to \gamma\gamma$ events with $|\eta| = [1.37, 1.52[$



(d) $Z \to \mu\mu\gamma$ events with $|\eta| = [1.52, 2.01[$



(e) $Z \to \mu\mu\gamma$ events with $|\eta| = [2.01, 2.47[$

Figure 9.18: The figures show the performance in *DeepCalo* and $E\text{calib}^{(BDT)}$ in $|\eta|$ and E_T bins for $Z \to \mu\mu\gamma$.

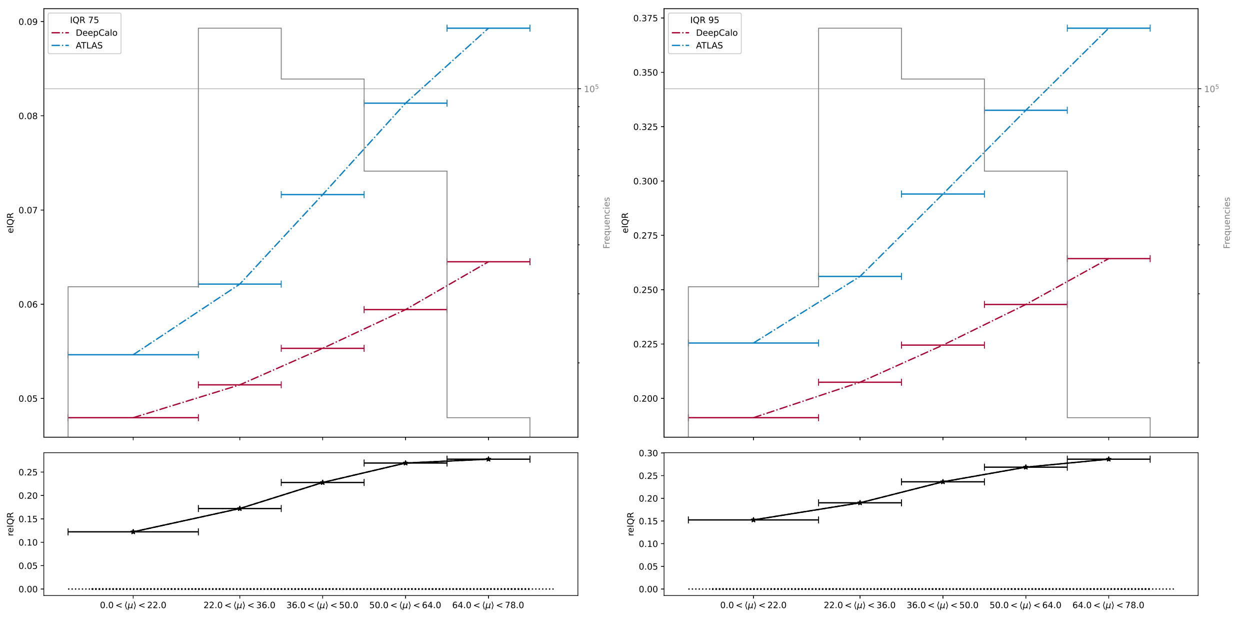


Figure 9.19: The figure shows the pileup performance of the *DeepCalo*$_{Zmmg,mc}$ model for both eIQR$_{75}$ and eIQR$_{95}$. The errorbar-plot indicates the eIQR$_{75}$ and its x-errors show the range of the bin. The histogram associated with the right axis shows the number of events in each bin. The bottom plot is the reIQR of the eIQR between *DeepCalo* and Ecalib$^{(BDT)}$.

9.6 Variables

9.6.1 Track variables

Figure 9.20 is related to the track variables from table 3.4 that can be seen in 9.20. It shows the geometrics of the track variables in the xy and zR plane.

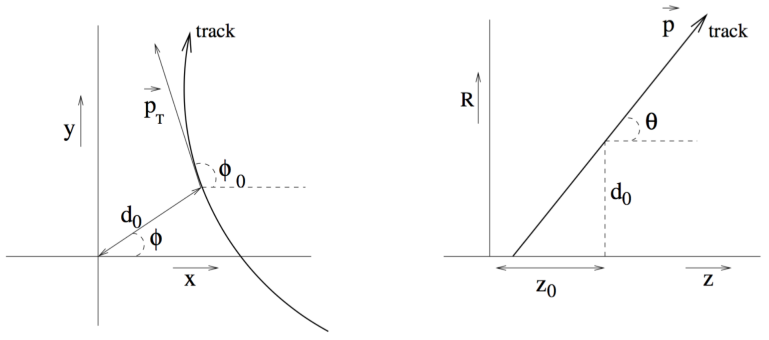


Figure 9.20: From [37]. The figure illustrates the perigee parameters of the tracks in the x, y plane in *left* figure and in the R, z plane in the *right* figure.

9.6.2 ECAL images

Figure 9.21 illustrates that if upscaling is not applied before filling the pixel values, the barycentre is shifted. It should be compared to figure 3.6. This is why the image is up-sampled before filling the pixel values.

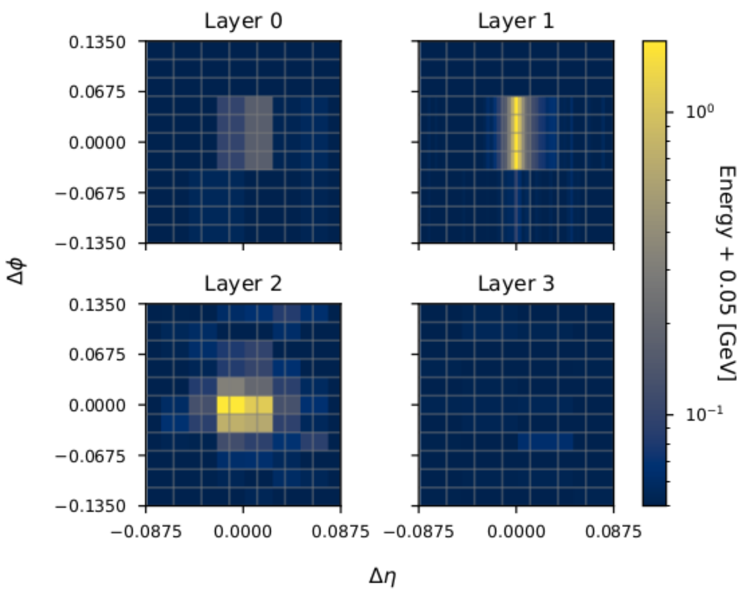


Figure 9.21: From [18], where the energies have been filled in before the up-sampled.

9.6.3 Standardization techniques

Two types of standardization were applied in this thesis, both algorithms from *scikit-learn* [33]. QuantileTransformer was used on distributions with a heavy-tail and *RobustScaler* was applied on the rest.

The *RobustScaler* is robust towards outliers, but instead of using the mean and standard deviation of a distribution[1], it uses the median for centering and the interquantile range for scaling. Each variable is standardized using

[1] Where both the mean and standard deviation are outlier dependent.

$$\hat{\mathbf{x}} = \frac{\mathbf{x} - Q_2(\mathbf{x})}{Q_3(\mathbf{x}) - Q_1(\mathbf{x})} \tag{9.9}$$

$\mathbf{x}$ being the variable and Q_n denoting the nth quartile.

For heavy-tailed distributions, QuantileTransformer is applied. It can transform variables to follow a uniform or a normal distribution to reduce the impact of outliers. The algorithm estimates the cumulative distribution function G^{-1} of a feature and applies it to construct a uniform distribution. Afterwards, a quantile function of a desired output distribution is mapped to the previous obtained distribution. The transformation is non-linear, so it may distort the linear correlation between features at same scale, but makes features measured at different scales more comparable.

Using RobustScaler and QuantileTransformer for scaling variables

In figure 9.22 and 9.23, the *RobustScaler* from [33] has been used to standardize the variables. Many of the variables still have an unintended long-tail, which can be problematic for a NN.

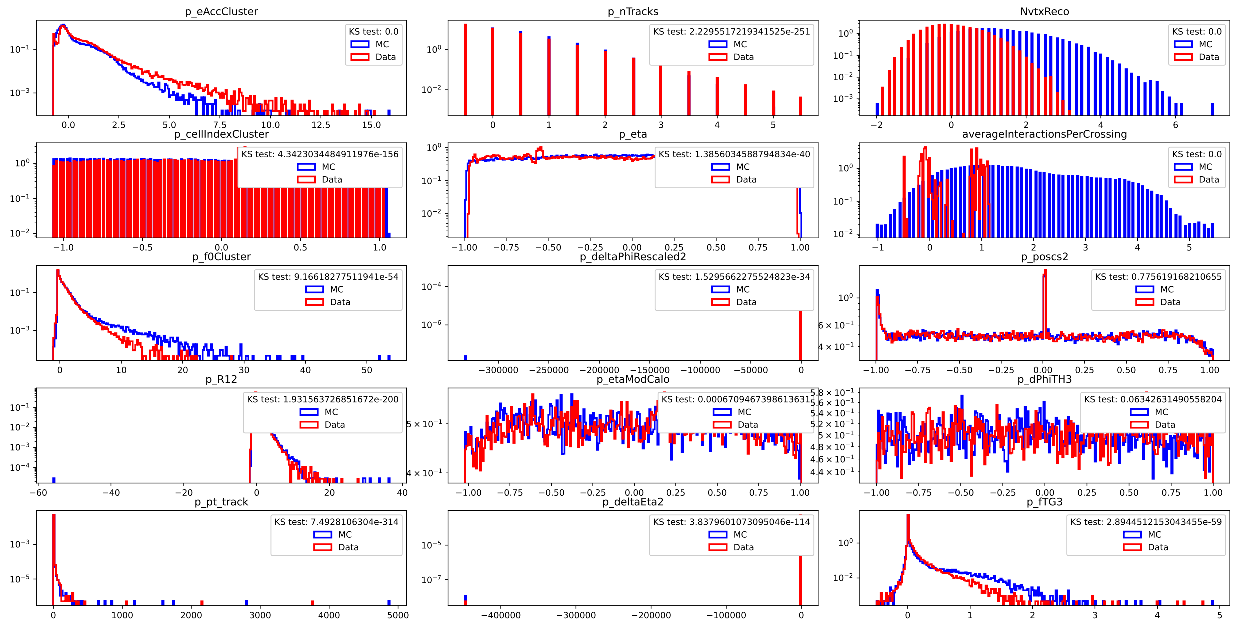


Figure 9.22: The figure shows the distribution of the scalar variables in MC standardized by the *RobustScaler*. The scalar distributions belong to the $Z \to ee$ MC sample.

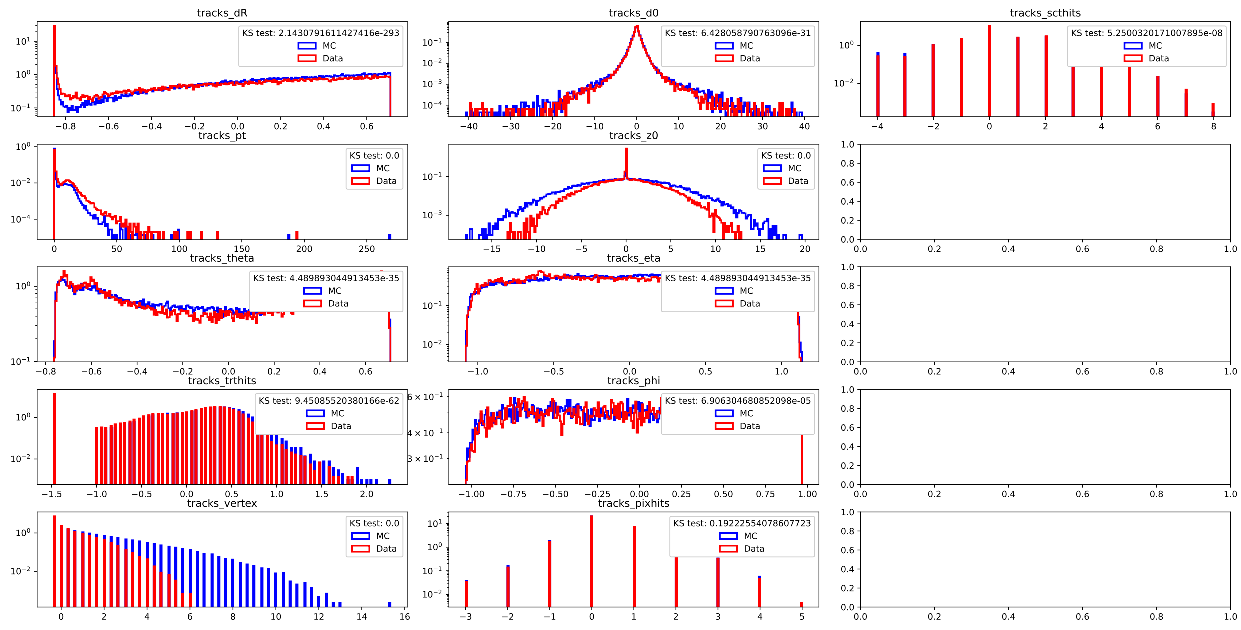


Figure 9.23: The figure shows the distribution of the track variables in MC standardized by the *RobustScaler*. The track distributions belongs to the $Z \to ee$ MC sample.

To fix the long-tailed distributions we apply the QuantileTransformer instead *RobustScaler* to the distribution with the tail. The result can be seen in figure 9.24 and 9.25, where the long-tailed distributions have been removed. Note that when using QuantileTransformer, the linear correlation between variables might change.

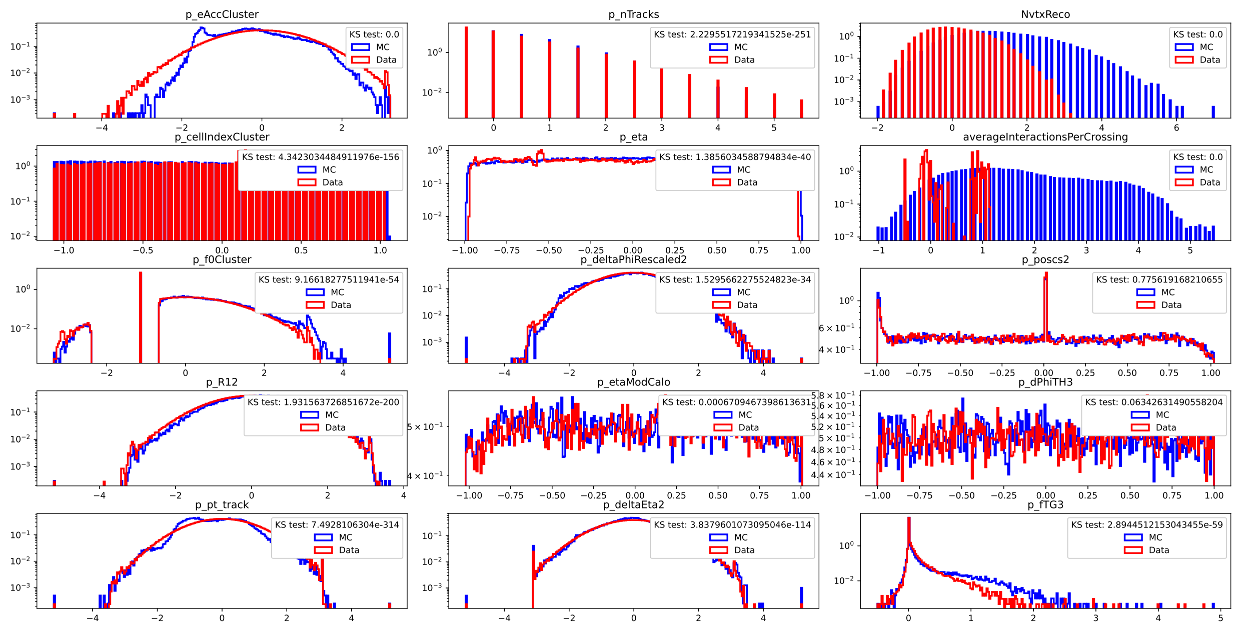


Figure 9.24: The figure shows the distribution of the scalar variables in MC standardized by the *RobustScaler* and QuantileTransformer. QuantileTransformer has been used for distributions with a long tail. The scalar distributions belong to the $Z \to ee$ MC sample.

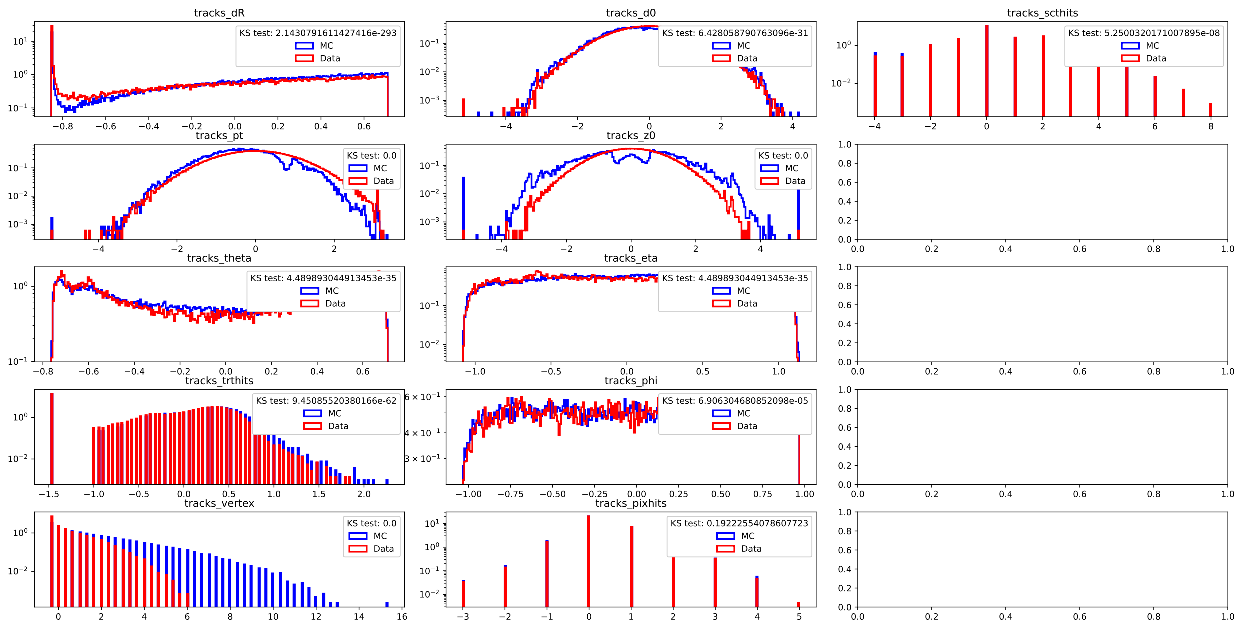


Figure 9.25: The figure shows the distribution of the track variables in MC standardized by the *RobustScaler* and QuantileTransformer. QuantileTransformer has been used for distributions with a long tail. The track distributions belong to the $Z \to ee$ MC sample.

9.6.4 *Variable distributions of Outliers*

In figure 9.26 and 9.27, the distribution of scalar and track variables of MC for $Z \to ee$ used in testing the $DeepCalo_{Zee,mc}$ model are shown. The variables have been divided into outliers and non-outliers depending on whether their absolute error (AB) is above 2.5. The correlation plot of the outliers selected can be seen in figure 7.10. Figures 9.26 and 9.27 are used in the discussion to present any possible systemics in the outliers.

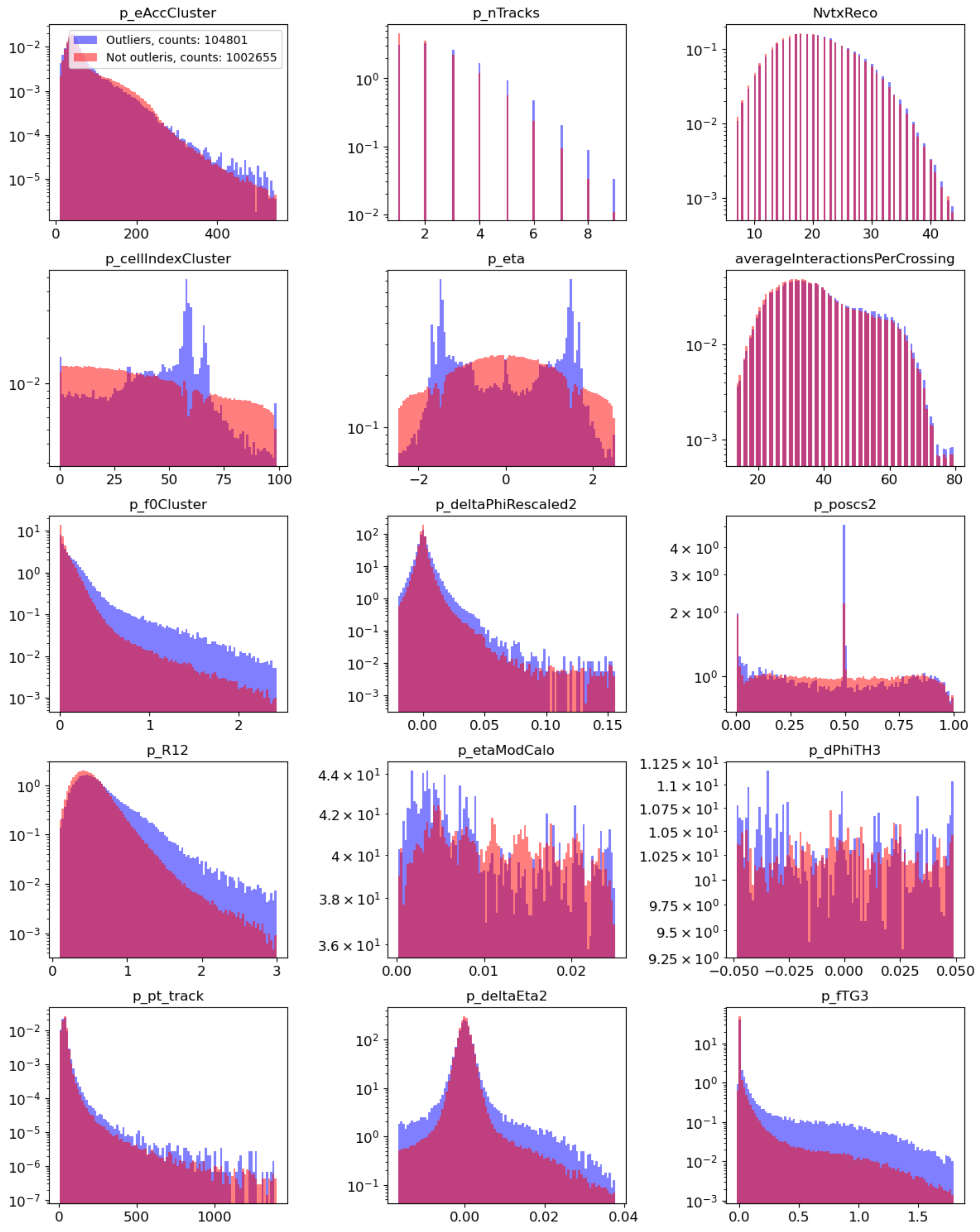


Figure 9.26: The figure shows the distribution of the scalar variables of MC for the outliers (Blue) and the non-outliers (Red) for the $DeepCalo_{Zee,mc}$ model from chapter 5. The outliers have been identified as the events with a $AE > 2.5$ and the number of outliers can be seen in the legend.

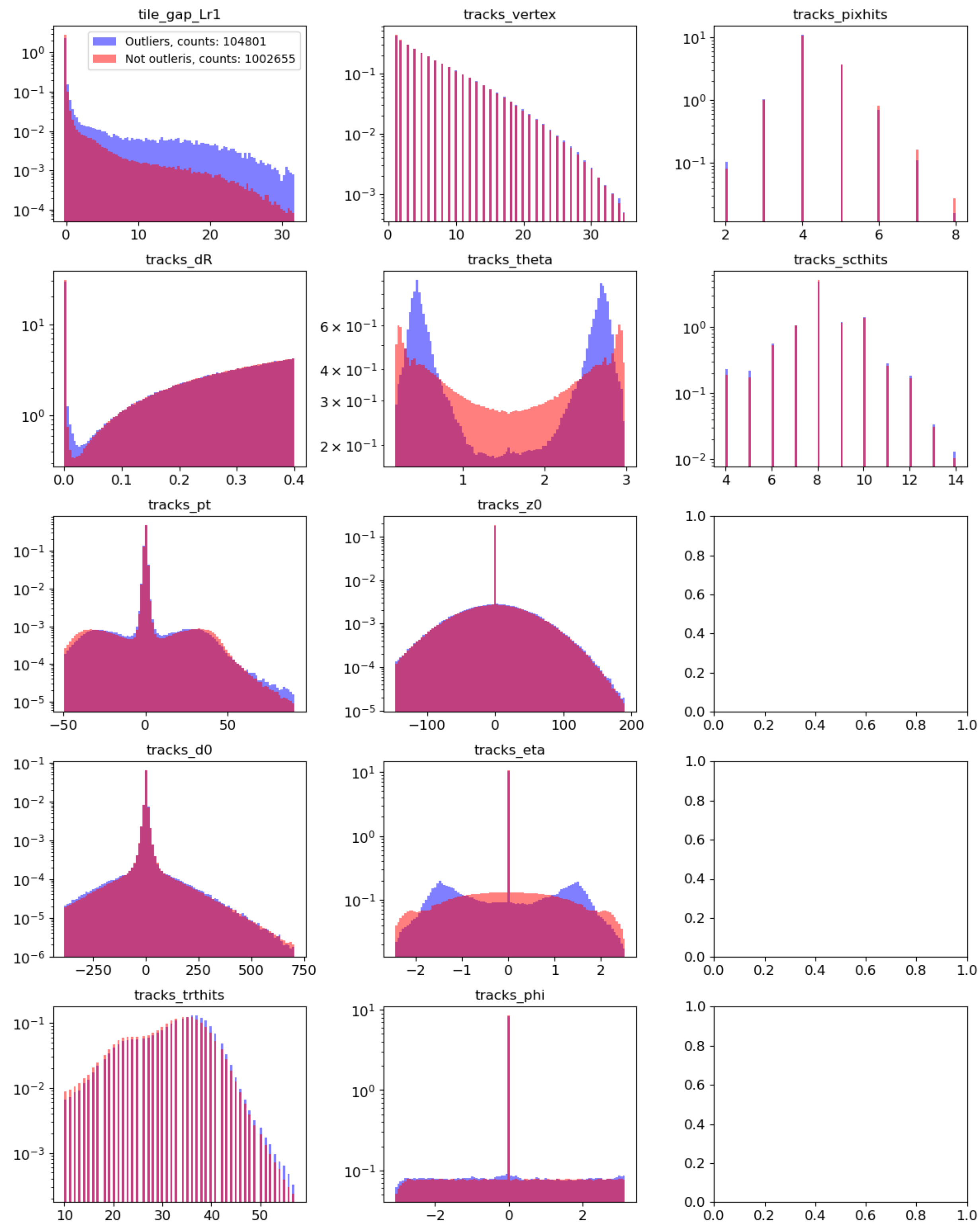
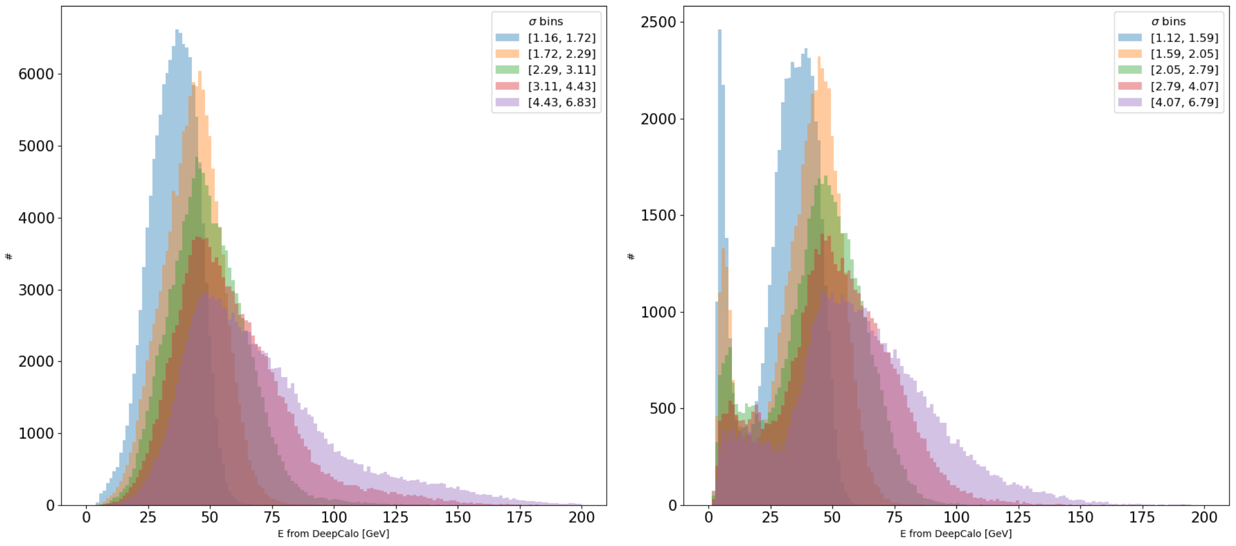


Figure 9.27: The figure shows the distribution of the track variables of MC for the outliers (Blue) and the non-outliers (Red) for the $DeepCalo_{Zee,mc}$ model from chapter 5. The outliers have been identified as the events with a $AE > 2.5$ and the number of outliers can be seen in the legend.

9.7 *Performance for the uncertainties*

Figure 9.28 shows the energy prediction from *DeepCalo*$_\sigma$ for MC and Data for the five bins. In Data, there is a cluster of poorly reconstructed events at $E = [0, 10]$ that is not visible in MC. In figure 9.29,



(a) Prediction on MC **(b)** Prediction on Data

Figure 9.28: The figure shows the distribution of energies predicted by *DeepCalo*$_\sigma$ for the five σ bins.

we see the pull plot using the re-scale uncertainties from *DeepCalo*$_\sigma$. The uncertainties are reduced by 4.5.

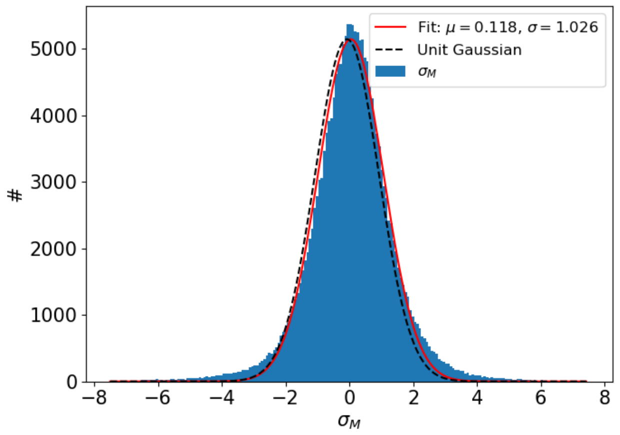


Figure 9.29: The figure shows the distribution of energies predicted by *DeepCalo*$_\sigma$ for the five σ bins, where the σ_M has been reduced by 4.5.

9.8 *DeepCalo architecture*

´ In the following section, we will be showing the sub-module architectures of *DeepCalo* for $Z \to ee$, illustrated used *Tensorflow*. Some of the figures are small, as they must fit on a page, so viewing them in PDF form is recommend.

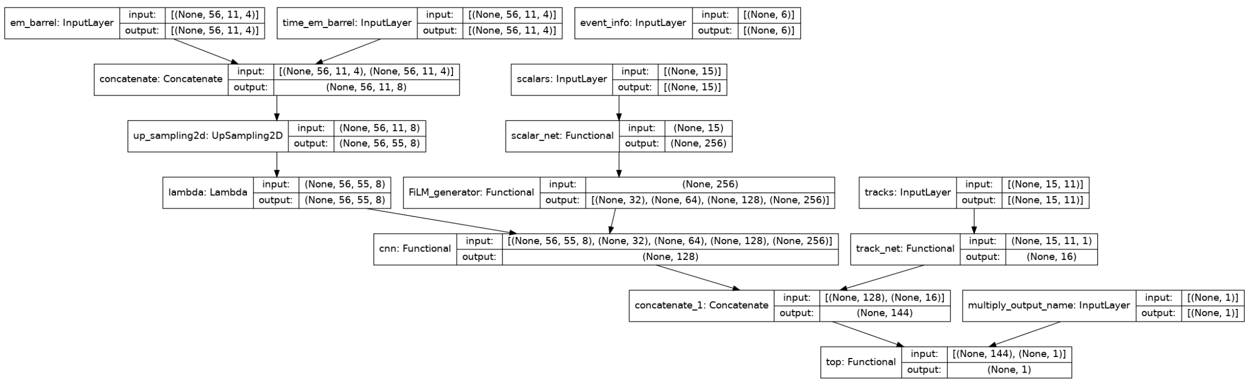

em_barrel: InputLayer
input: [(None, 56, 11, 4)]
output: [(None, 56, 11, 4)]
time_em_barrel: InputLayer
input: [(None, 56, 11, 4)]
output: [(None, 56, 11, 4)]
event_info: InputLayer
input: [(None, 6)]
output: [(None, 6)]
concatenate: Concatenate
input: [(None, 56, 11, 4), (None, 56, 11, 4)]
output: (None, 56, 11, 8)
scalars: InputLayer
input: [(None, 15)]
output: [(None, 15)]
up_sampling2d: UpSampling2D
input: (None, 56, 11, 8)
output: (None, 56, 55, 8)
scalar_net: Functional
input: (None, 15)
output: (None, 256)
lambda: Lambda
input: (None, 56, 55, 8)
output: (None, 56, 55, 8)
FiLM_generator: Functional
input: (None, 256)
output: [(None, 32), (None, 64), (None, 128), (None, 256)]
tracks: InputLayer
input: [(None, 15, 11)]
output: [(None, 15, 11)]
cnn: Functional
input: [(None, 56, 55, 8), (None, 32), (None, 64), (None, 128), (None, 256)]
output: (None, 128)
track_net: Functional
input: (None, 15, 11, 1)
output: (None, 16)
concatenate_1: Concatenate
input: [(None, 128), (None, 16)]
output: (None, 144)
multiply_output_name: InputLayer
input: [(None, 1)]
output: [(None, 1)]
top: Functional
input: [(None, 144), (None, 1)]
output: (None, 1)

Figure 9.30: The figure shows the *DeepCalo* architectures with its submodules.

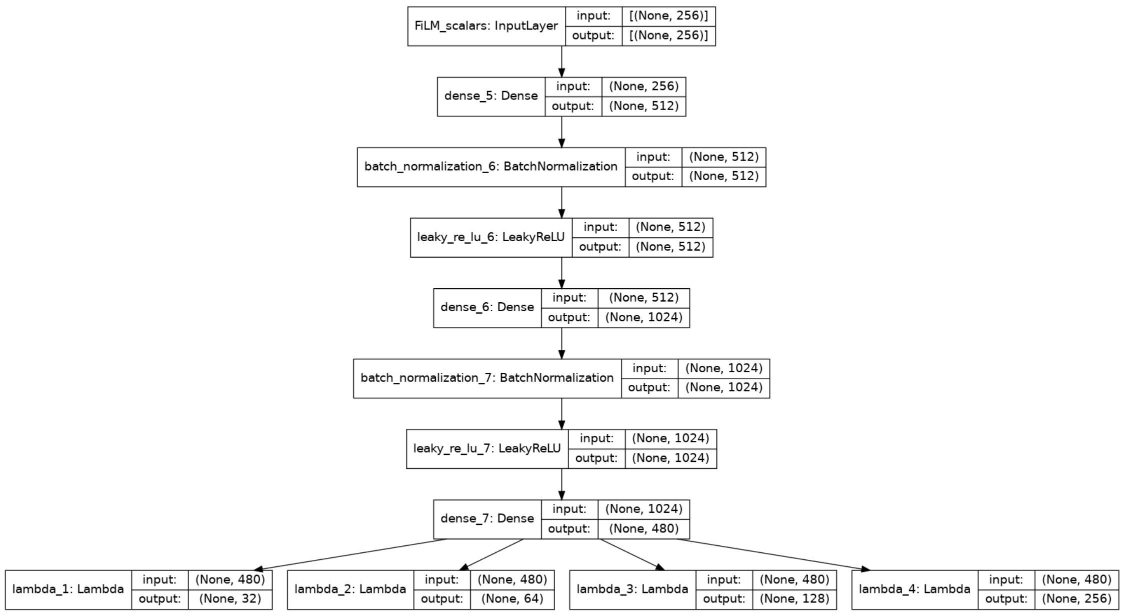


Figure 9.31: The figure shows the *FiLM gen.* of *DeepCalo*.

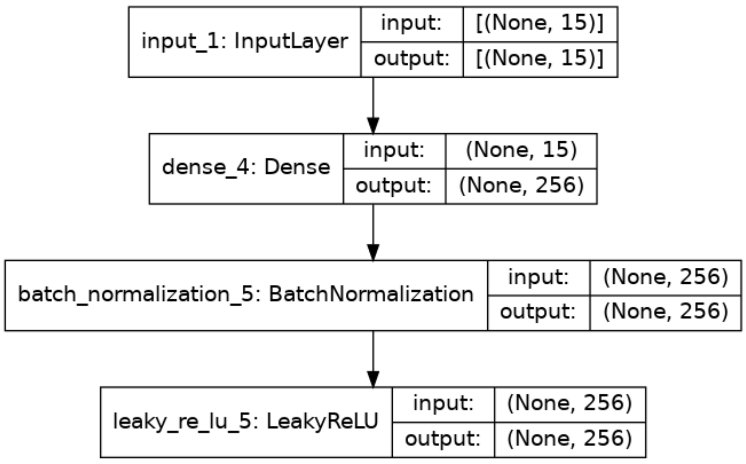


Figure 9.32: The figure shows the *ScalarNet* of *DeepCalo*.

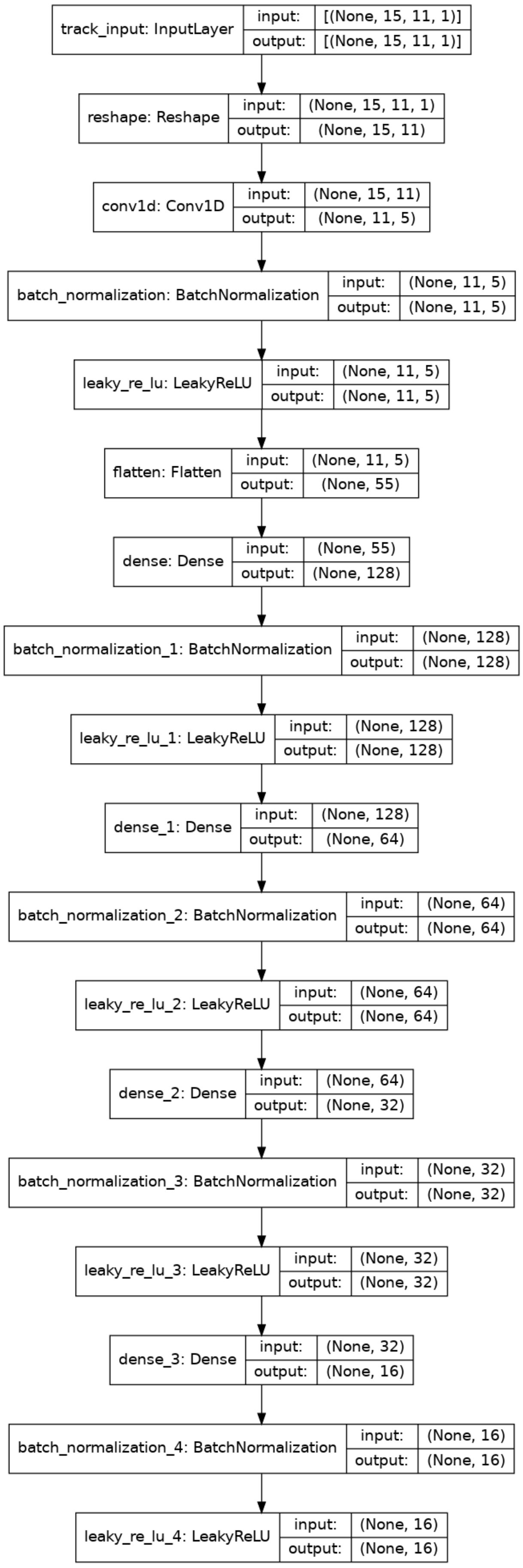


Figure 9.33: The figure shows the *TrackNet* of *DeepCalo*.

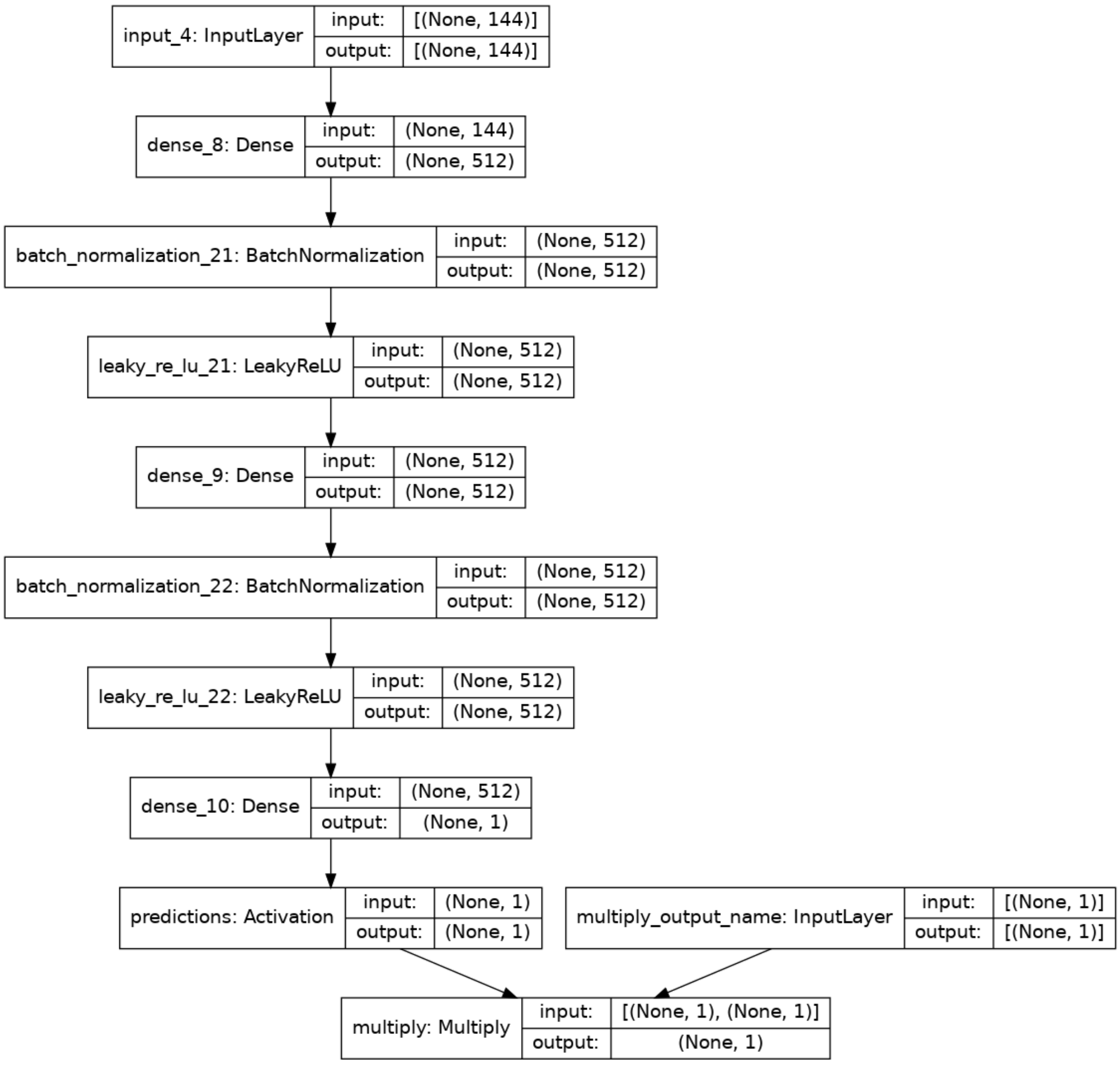

input_4: InputLayer | input: | [(None, 144)]
output: | [(None, 144)]
dense_8: Dense | input: | (None, 144)
output: | (None, 512)
batch_normalization_21: BatchNormalization | input: | (None, 512)
output: | (None, 512)
leaky_re_lu_21: LeakyReLU | input: | (None, 512)
output: | (None, 512)
dense_9: Dense | input: | (None, 512)
output: | (None, 512)
batch_normalization_22: BatchNormalization | input: | (None, 512)
output: | (None, 512)
leaky_re_lu_22: LeakyReLU | input: | (None, 512)
output: | (None, 512)
dense_10: Dense | input: | (None, 512)
output: | (None, 1)
predictions: Activation | input: | (None, 1)
output: | (None, 1)
multiply_output_name: InputLayer | input: | [(None, 1)]
output: | [(None, 1)]
multiply: Multiply | input: | [(None, 1), (None, 1)]
output: | (None, 1)

Figure 9.34: The figure shows the *Top* of *DeepCalo*.

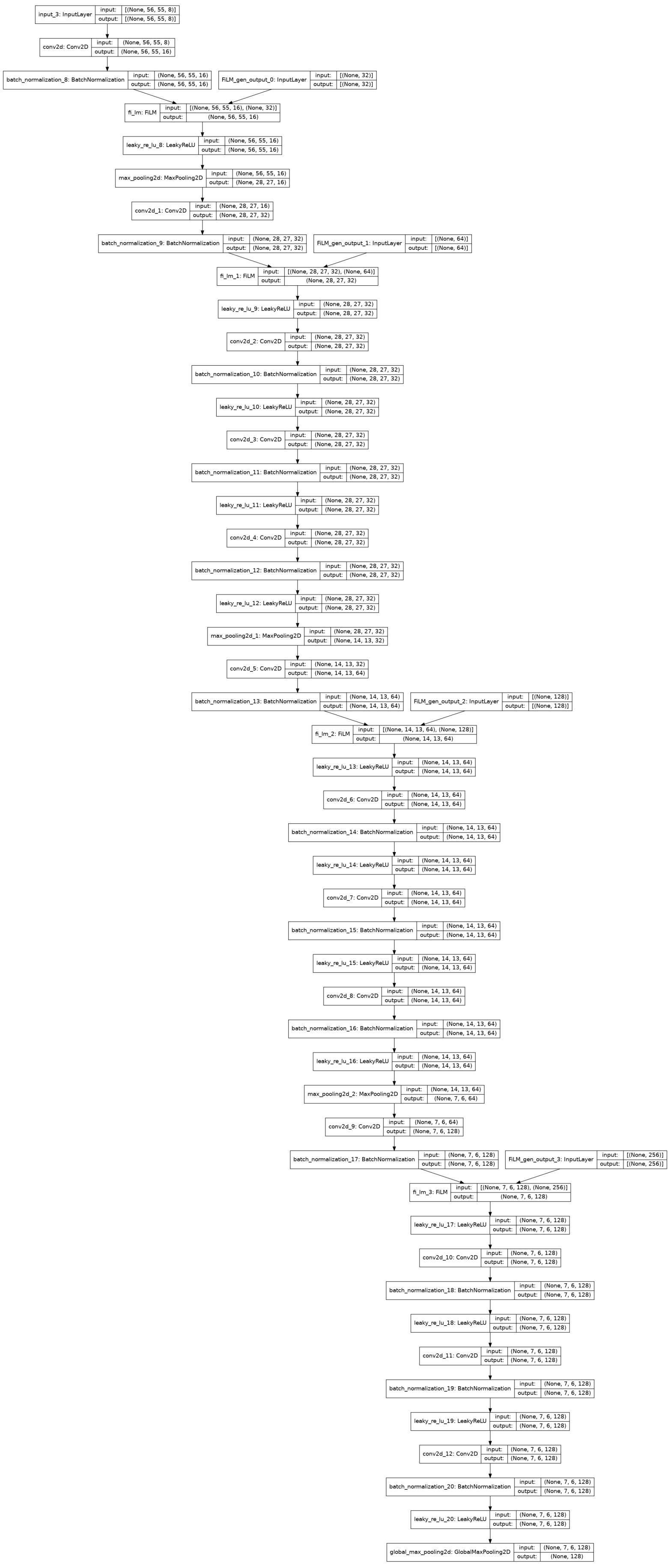


Figure 9.35: The figure shows the *CNNnet* of *DeepCalo*.

Bibliography

[1] G. Aad et al. "A search for the $Z\gamma$ decay mode of the Higgs boson in pp collisions at $\sqrt{s} = 13$ TeV with the ATLAS detector." In: *Physics Letters B* (2020), pp. 1–3. DOI: 10.1016/j.physletb.2020.135754. URL: http://dx.doi.org/10.1016/j.physletb.2020.135754.

[2] M (CERN) Aleksa et al. *ATLAS Liquid Argon Calorimeter Phase-I Upgrade Technical Design Report*. Tech. rep. 2013. URL: https://cds.cern.ch/record/1602230.

[3] Malte Algren. *NTupleProduction*. URL: https://gitlab.cern.ch/disciples-of-troels/ntupleproduction/-/tree/CellEnerg.

[4] Evelina Bouhova-Thacker and Vakhtang Kartvelishvili. "*Electron bremsstrahlung recovery in ATLAS tracking using Dynamic Noise Adjustment*." In: (Sept. 2021), Figure 2.

[5] James Catmore. *DerivationFramework*. URL: https://twiki.cern.ch/twiki/bin/viewauth/AtlasProtected/DerivationFramework. (accessed: 25.11.2020).

[6] CERN. *Facts about CMS*. 2013. URL: https://cms-docdb.cern.ch/cgi-bin/PublicDocDB/RetrieveFile?docid=4030&version=4&filename=CMSFactsheet_EN_Sept2013.pdf. (accessed: 20.09.2021).

[7] ATLAS Experiment CERN. *The Inner Detector*. 2020. URL: https://atlas.cern/discover/detector/inner-detector.

[8] S. Chatrchyan et al. "Observation of a new boson at a mass of 125 GeV with the CMS experiment at the LHC." In: (2012), p. 1. DOI: 10.1016/j.physletb.2012.08.021. URL: http://dx.doi.org/10.1016/j.physletb.2012.08.021.

[9] ATLAS Collaboration. ""Observation of a new particle in the search for the Standard Model Higgs boson with the ATLAS detector at the LHC"." In: (2012).

[10] CERN collaboration. *Rucio documentation*. URL: https://rucio.cern.ch/. (accessed: 01.08.2021).

[11] The ATLAS Collaboration. "Electron and photon energy calibration with the ATLAS detector using 2015–2016 LHC proton–proton collision data." In: (2019), pp. 6–9. URL: http://dx.doi.org/10.1088/1748-0221/14/03/P03017.

[12] The ATLAS Collaboration. "Electron and photon performance measurements with the ATLAS detector using the 2015-2017 LHC proton-proton collision data." In: (2019), pp. 11–13. DOI: 10.1088/1748-0221/14/12/P12006. URL: https://cds.cern.ch/record/2684552.

[13] The ATLAS Collaboration. "The ATLAS Experiment at the CERN Large Hadron Collider." In: (2008).

[14] Atlas Collection. *Athena*. URL: https://gitlab.cern.ch/atlas/athena. (accessed: 27.08.2021).

[15] Chris Cunningham. *Data Transfer Rates Compared (RAM vs PCIe vs SATA vs USB vs More!)* URL: https://blog.logicalincrements.com/2018/08/data-transfer-rates-bandwidth-cpu-ram-pcie-m-2-sata-usb-hdmi/.

[16] Timothy Dozat. "Incorporating Nesterov Momentum into Adam." In: (2016).

[17] Harm de Vries Vincent Dumoulin Ethan Perez Florian Strub and Aaron C. Courville. "FiLM: Visual Reasoning with a General Conditioning Layer." In: (2017), pp. 1–4. URL: http://arxiv.org/abs/1709.07871.

[18] Frederik G. Faye. *Energy reconstruction of electrons and photons using Convolutional Neural Networks*. 2019.

[19] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016, pp. 326–365.

[20] Peter Hansen. *Particle detectors and accelerators Lecture notes*. 3rd. 2016, pp. 9–12, 77–80, 93–95, 97–99, 101–106.

[21] Sergey Ioffe and Christian Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift." In: (2015), pp. 1–5. URL: http://arxiv.org/abs/1502.03167.

[22] Shruti Jadon. *Introduction to Different Activation Functions for Deep Learning*. URL: https://medium.com/@shrutijadon10104776/survey-on-activation-functions-for-deep-learning-9689331ba092. (accessed: 01.07.2021).

[23] Jeremy Jordan. *Setting the learning rate of your neural network.* URL: https://www.jeremyjordan.me/nn-learning-rate/. (accessed: 01.08.2021).

[24] Alex Kendall and Yarin Gal. "What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision?" In: (2017), pp. 1–4. URL: http://arxiv.org/abs/1703.04977.

[25] Susanne Kuehn. "Impact of the HL-LHC detector upgrades on the physics program of the ATLAS and CMS experiments." In: ().

[26] Sandrine Laplace. *HGamma DxAOD samples*. URL: https://twiki.cern.ch/twiki/bin/view/AtlasProtected/HggDerivationSamples#HIGG1D1_and_HIGG1D2_skimming. (accessed: 25.11.2020).

[27] Y. LeCun et al. “Backpropagation Applied to Handwritten Zip Code Recognition.” In: *Neural Computation* 1 (1989), pp. 541–551.

[28] Scott Lundberg and Su-In Lee. *SHapley Additive exPlanations*. URL: https://github.com/slundberg/shap. (accessed: 24.08.2021).

[29] Giovanni Marchiori. *EGamma AOD derivations*. URL: https://twiki.cern.ch/twiki/bin/view/AtlasProtected/EGammaxAODDerivations#Derivations_defined_for_egamma. (accessed: 25.11.2020).

[30] Peter McCready. *Atlas and the search for dark matter*. 2018. URL: http://news.bbc.co.uk/2/hi/science/nature/7534850.stm. (accessed: 20.09.2021).

[31] Gordon E. Moore. “Cramming more components onto integrated circuits, Reprinted from Electronics, volume 38, number 8, April 19, 1965, pp.114 ff.” In: *IEEE Solid-State Circuits Society Newsletter* 11.3 (2006), pp. 33–35. DOI: 10.1109/N-SSC.2006.4785860.

[32] Nvidia. *CUDA C++ Programming Guide - The Benefits of Using GPUs*. URL: https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html.

[33] F. Pedregosa et al. “Scikit-learn: Machine Learning in Python.” In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[34] Course by Peter Hansen and Jens Jørgen Gaardhøje. *Introduction to Nuclear and Particle Physics*. 2018.

[35] Karolos Potamianos. “The upgraded Pixel detector and the commissioning of the Inner Detector tracking of the ATLAS experiment for Run-2 at the Large Hadron Collider.” In: (2016), pp. 2–3.

[36] Andrew Purcell. *https://cds.cern.ch/record/1473657*. 2nd. 2017.

[37] K. Ran. “Study on Simulation of ATLAS ITK Strips.” In: (2016), [figure 8]. URL: https://www.desy.de/f/students/2016/reports/KunlinRan.pdf.gz.

[38] Sebastian Ruder. “An overview of gradient descent optimization algorithms.” In: (2016). arXiv: 1609.04747. URL: http://arxiv.org/abs/1609.04747.

[39] Yevgeny Seldin. *Machine Learning Lecture Notes*. Department of Computer Science, University of Copenhagen, 2019, pp. 20–26.

[40] Eduardo Simas, Jose M. Seixas, and Luiz Caloba. “Self-organized mapping of calorimetry information for high efficient online electron / jet identification in ATLAS.” In: (2007), [Figure 2 on page 3]. DOI: 10.22323/1.050.0055.

[41] Leslie N. Smith. “No More Pesky Learning Rate Guessing Games.” In: (2015), pp. 1–5. URL: http://arxiv.org/abs/1506.01186.

[42] S. Stärz. “Upgraded readout electronics for the ATLAS LAr calorimeter at the phase I of LHC.” In: (2013).

[43] M. Tanabashi et al. "Review of Particle Physics." In: *Phys. Rev. D* 98 (3 2018), pp. 33–35. DOI: 10.1103/PhysRevD.98.030001. URL: https://link.aps.org/doi/10.1103/PhysRevD.98.030001.

[44] Tensorflow. *TensorBoard: TensorFlow's visualization toolkit*. URL: https://www.tensorflow.org/tensorboard.

[45] Tensorflow. *TFRecord and tf.train.Example*. URL: https://www.tensorflow.org/tutorials/load_data/tfrecord.

[46] Tensorflow and Google. *XLA Architecture*. URL: https://www.tensorflow.org/xla/architecture.

[47] Tensorflow and Nvidia. *Mixed precision*. URL: https://www.tensorflow.org/guide/mixed_precision.

[48] Scott Thornton. *NVMe vs SATA: What's the difference and which is faster?* URL: https://www.microcontrollertips.com/why-nvme-ssds-are-faster-than-sata-ssds/.

[49] Wikipedia. *Crystal Ball Function*. URL: https://en.wikipedia.org/wiki/Crystal_Ball_function#/media/File:CrystalBallFunction.svg. (accessed: 27.08.2021).

[50] Nishio M. Do R.K.G. et al. Yamashita R. "Convolutional neural networks: an overview and application in radiology." In: (2018), pp. 611–629. DOI: \url{https://doi.org/10.1007/s13244-018-0639-9}.

List of Figures

List of Tables