Energy regression at ATLAS using graph neural networks

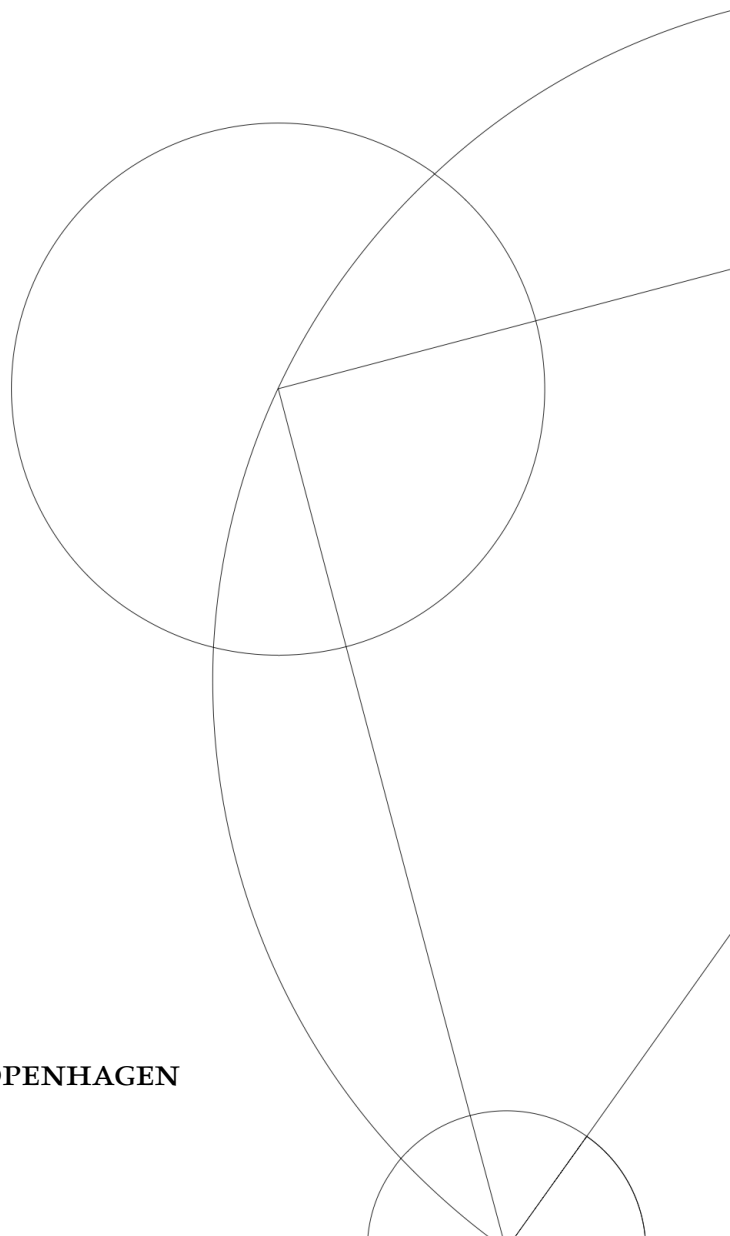
Exploring a more general approach to energy regression combining tracking and calorimetry.

Master Thesis in Computational Physics

Written by *Aske Rosted*

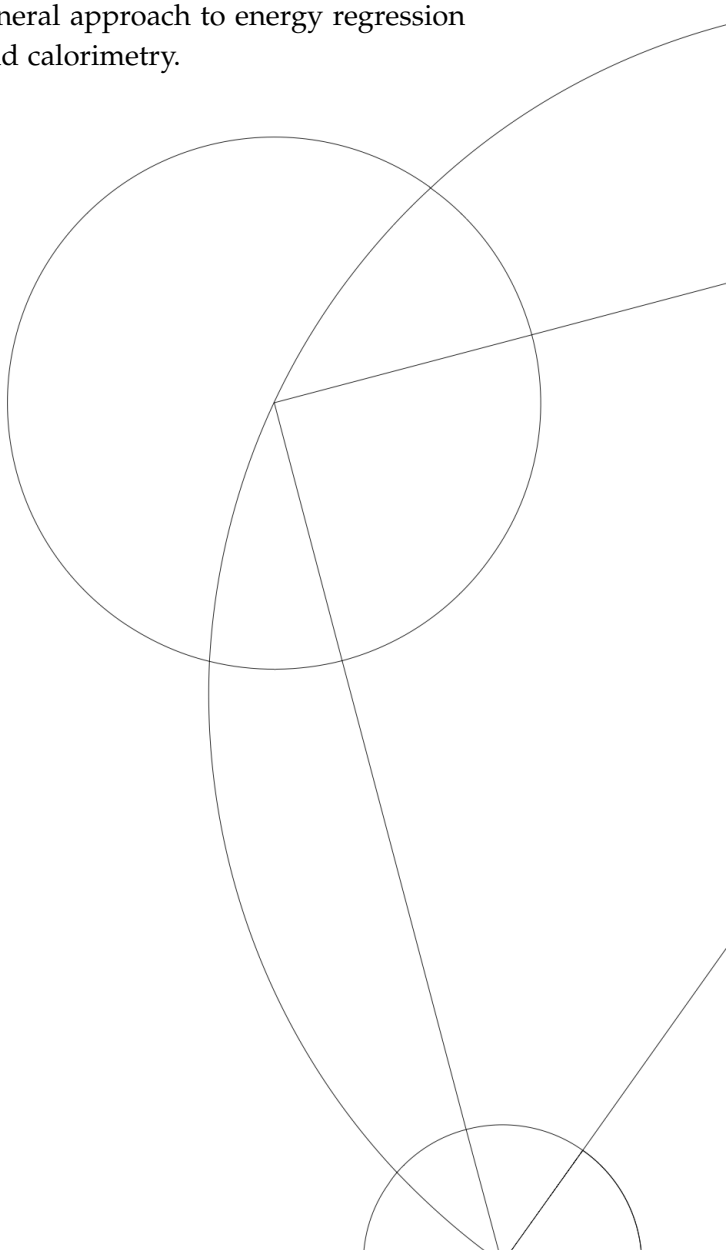
2021-2022

Supervised by
Troels C. Petersen



University of Copenhagen

Name of Institute:	Faculty of Science
Name of Department:	The Niels Bohr Institute
Author(s):	Aske Rosted
Email:	ksn733@alumni.ku.dk
Title and subtitle:	Energy regression at ATLAS using graph neural networks - Exploring a more general approach to energy regression combining tracking and calorimetry.
Supervisor(s):	Troels C. Petersen
Handed in:	2021-2022
Defended:	2021-2022



Name _____

Signature _____

Date _____

Abstract

The resolutions of the Z and higgs bosons is dependent on the electron and photon resolutions. These energy resolutions can only be improved on by either improving the algorithms behind the regression or by upgrading/building new detectors, the latter which is an extremely expensive endeavor.

The ATLAS collaboration currently uses boosted decision tree algorithms for their energy regression. These have proven quite effective but developments within the machine learning field have brought about newer algorithms capable of including spatial information about input data. This thesis implements a cutting edge machine learning method called graph neural network to improve upon energy regression of $Z \to ee$. A relative improvement over currently implemented methods of

$$ReIQR = 12.5 \pm 3 \times 10^{-3}$$

and

$$\langle 1 - \frac{\sigma_{CB}^{GNN}}{\sigma_{CB}^{ATLAS}} \rangle = 10.56 \pm 0.008$$

using two different evaluation metrics is found for the final model.

The relative improvement was found to be robust across different detector regions although and largest for lower energy electrons.

The relative performance over the currently implemented method is found to increase with pile-up, resulting in expected greater relative performance in future high-luminosity runs.

Two different model architectures have been tested, but only one lead to improvements over currently implemented methods. Suggestions for additional model improvements are given and further testing of the model using real data and $Z \to \mu\mu\gamma$ can be achieved with only little additional development.

Acknowledgments

I would like to bring a big thanks to Troels Christian Petersen, who has been an excellent supervisor, friend and ally throughout the entire thesis from model development to the final writing process.

I would also like to thank Malte Algren and Rasmus Faldt Ørsøe whose works have been the starting block for this thesis and who both spent a great amount of time helping and giving advice during especially the early stages of the model development.

A big thanks also to Kaare and Leon for allowing me to come in and vent in your office and to Jason for counsel assistance with my further prospects within academics.

Finally, to my family who have been patient, supportive and understanding during times, where I have not been able to be present as much as I wanted.

MSc Thesis,
Niels Bohr Institute, University of Copenhagen

Thesis submitted 8th of July 2022 for the completion of the degree Master of Science (MSc) in computational physics.

LaTeXtemplate: A Tufte-Style Book tufte-latex.github.io/tufte-latex/

First printing, July 2022

Contents

1 Introduction

The aim of this thesis is to further the range of machine learning (ML) implementations at the Large Hadron Collider (LHC) experiment called A Toroidal LHC Apperatus (ATLAS) to include Graph Nueral Networks (GNN). There are several benefits inherent to a GNN implementation. The currently implemented method in ATLAS is a Boosted Decision Tree method which does not include any of the *raw* cell or track data, but is based solely on global event summary variables also called scalar variables.

Previous students have researched Convolutional Neural Network (CNN) implementations in an attempt to include *raw* cell data into the energy regression scheme. These theses have shown large potential gains in resolution compared the BDT. However, these methods require an up-sampling of the lower resolution calorimeter units which increases the calorimeter data size by roughly a factor 10. The CNN implementation also does not have a good way of including track data which have a variable size per event, and have resorted to including a fixed number of tracks disregarding the rest or 0 padding if fewer tracks are available.

A GNN implementation would solve both those issues as it can handle data input which is both of variable size and non-equidistant. At high rates of pileup one of the largest causes of lost resolution is the inability to include auxiliary track features. As pileup increases along with luminosity the benefit of a GNN implementation is expected to grow along with the increase in luminosity of run 3 and *high-lumi* run 4.

These first four chapters on particle physics (chapter 2), detector physics (chapter 3), reconstruction (chapter4) and machine learning (chapter 5), will be introductory chapters which aim to give the reader knowledge of the ATLAS experiment and the jargon used within high energy particle physic; as well as a foundation for understanding the motivation behind looking for new ML methods for implementation in *regression and classification*.

The following 2 chapters on the data-pipeline in chapter 6 and model architecture in chapter 7 will describe the data sets and the algorithms, that have been develop in this thesis.

The final 2 chapters will be a presentation of the results in chapter 8 and a summary of the most important conclusions along with an outlook into further applications of the model and possible improve-

ments in chapter 9.

We will unless otherwise stated work in natural units ($\hbar = c = m_e = \epsilon_0 = 1$) throughout this thesis, as this is common practice and simplifies mathematical expressions within the particle physics domain.

2 Particle physics

This chapter seeks to give the reader an insight into the particle physics necessary to understand the motivations and reasoning used in the rest of the thesis. This thesis is much more applied than it is theoretical in nature and therefore this chapter does not try to give a full understanding of particle physics at the energy frontier, but only cases relevant to the specific data used in this thesis.

2.0.1 The existence of elementary particles

Before the Standard Model (SM) The makeup of atoms; protons, neutrons, and electrons[1] were the smallest constituents of matter, these subatomic particles were the furthest physicist had been able to probe the matter that makes up the universe. In 1964 Murray Gell-Mann and George Zweig both independently came up with a theory, that could explain the interaction and properties of these strongly interacting particles. These theories both proposed that the subatomic particles protons and neutrons were made up of smaller particles with electrical charge 1/3 and spin 2/3 that of the electron in a symmetry scheme based on the mathematical SU(3) symmetry, and as such the theory of quarks were born.

In 1968 the linear accelerator proton-electron scattering experiments revealed signs of the inner structure of the nucleons[29]. We will return to how later in section 2.1.1.

[1] Which was a known elementary particle

2.1 Standard model

The behaviour and interaction of *elementary particles*[2] are currently best explained in the Standard Model (SM). The particles currently contained within the SM can be seen in Figure 2.1. The elementary particles are divided into two categories, fermions or the *matter particles*[3] and bosons, which are the *force carrier particles*.

[2] subatomic particles that aren't composed of other particles.

[3] although Z, W and Higgs bosons too are massive

Bosons and forces

There are 5 different bosons, plus one opposite charge W boson: the gluon which mediates the strong force, the photon which mediates the electromagnetic force and the Z and $W^{\pm}$ bosons which mediate the weak force. The Higgs boson is also considered a force carrier particle. However, it does not mediate any force. Instead the Higgs potential interacts with all particles of mass.

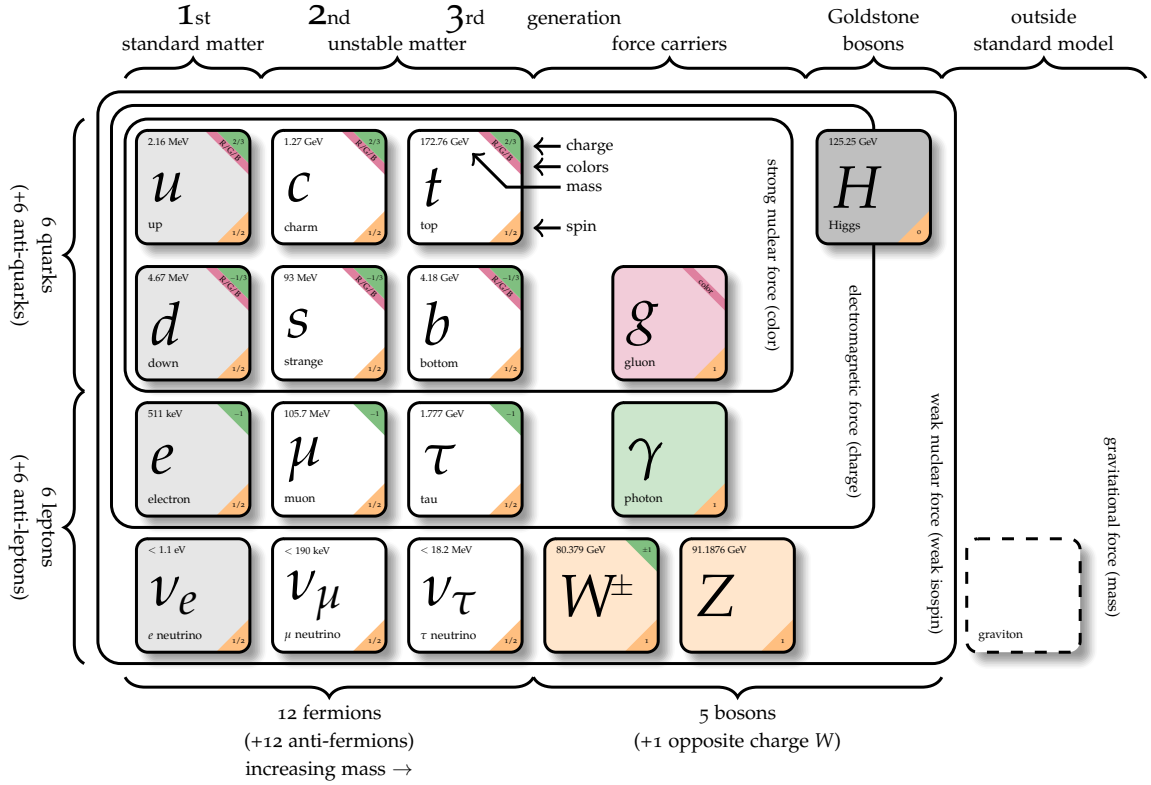


Figure 2.1: An overview of the different elementary particles ordered in their different categories, borders illustrating coupling. Created by David Galbraith and Carsten Burgard at CERN Webfest 2012.[72][59]. Values updated according to PDG[77]

The electromagnetic force and the weak force are unified in the *electroweak* theory which Sheldon Glashow, Abdus Salam and Steven Weinberg shared a Nobel Prize for in 1979. The strong force is described through the theory of Quantum ChromoDynamics (QCD). The gravitational force is not included in the Standard model, however the influence of gravitational forces are negligible at the scale of particles physics, and therefore we do not need to concern ourselves with it in this thesis.

Fermions

The fermions are subdivided into two types quarks and leptons. Quarks which are coupled[4] with the gluon, and leptons which are not. Quarks also cannot be observed freely but are bound in *hadrons* due to a phenomenon called *color confinement*, which is futher explained in chapter 2.3 .

For every elementary particle there also exists an anti-particle, except for the gluon, photon, Z boson and Higgs boson which are their own anti-particles, that is they turn into themselves under charge conjugation, the anti-particles are not portrayed in Figure 2.1 and are not of much interest for this thesis. It should however be mentioned that the familiar term electron are sometimes hijacked to loosely describe both electrons and their anti-particles positrons.

[4] coupling here is synonymous with interacting, the couplings or interactions are described through Yukawa coupling and the Yukawa potential.

Composite particles; Hadrons & Baryons

Composite particles are called hadrons and consist of 2 or more quarks, where a 2 quark particle is called a meson and a 3 quark particle is called a baryon. Anything that consists of more than 3 quarks is usually labeled an *exotic* variant of those two. Of the hadrons only protons are of interest in this thesis, since that is what is collided within the ATLAS experiment.

2.1.1 *Parton distribution functions*

Returning to discovering the existence of quarks, this was done by looking at the Parton Distribution Functions (PDF) in Deep Inelastic Scattering (DIS) events. We consider the proton-electron experiment mentioned in section 2.0.1. A schematic of the interaction can be seen in Figure 2.2.

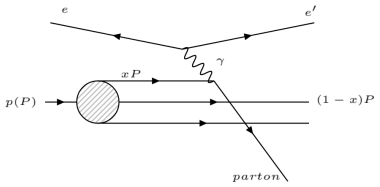


Figure 2.2: schematic representation of a deep-inelastic scattering event between an electron and a proton, the electron striking out a parton (quark or gluon) from the proton.

The electron interacts with one of the partons of the proton by an electroweak interaction i.e. through a photon. In the deep inelastic event that knocks out the partons, which results in showers in two directions, one in roughly the original directions of the proton and another for the parton. The momentum of the electron before and after the exchange can be measured, the energy momentum of the two *jets* can also be measured. Through complicated kinematics it is possible to determine which parton was hit (up/down quark, gluon etc.). Q^2 is related to the squared energy-momentum transfer (q^2) through.

$$Q^2 = -q^2 = -[(E - E')^2/c^2 - (\mathbf{p} - \mathbf{p'})] \tag{2.1}$$

where E and E' are the electron energy-momentum before and after, $\mathbf{p}$ and $\mathbf{p'}$ are the 3 momentum vector of the proton and the jet in the direction of the proton. Q^2 also represents the absolute value of the 4-momentum of the virtual photon[5] in the proton-electron collision, which *delivers* the energy to the parton.

[5] denoted γ in figure 2.2

Now if the proton consists of 3 quarks, we would expect to see that a DIS will knock out one of the quarks with $x = 1/3$, with x being the fraction of momenta pertaining to the knocked out quark. However, since the quarks interact with each other through the strong force what would have otherwise been a delta function at $x = 1/3$ now becomes smeared out, and this is what was seen in the early experiments.

There is a large discrepancy between the proton mass ($M_p \approx 938MeV$) and the sum of the quark masses which are approximately $M_{uud} \approx 9MeV$. In fact, we are off by a factor of about a 100. Furthermore, when increasing Q^2 we find that the PDF og the proton is pushed towards smaller values of x. This is due to what is sometimes called "sea-quarks". It turns out that the proton consist of 3 quarks namely two up and one down quark also called valence-quarks *and* additional quark anti-quark pairs, which are continuously produced and annihilated through radiation of virtual gluons, and as we increase the energy we are more likely to hit these particles.

In the ATLAS experiment we are not studying proton electron collision but instead proton-proton collisions. This produces considerably more complicated PDFs, but the concept remains the same. Figure 2.3 shows the Next to Leading Order (NLO) PDFs for different particles at two different Q^2 values. The figures show how a smaller x equates hitting either the *sea-quarks* or the virtual gluons.

These figures have been generated using data from several experiments, not only the ATLAS experiment.

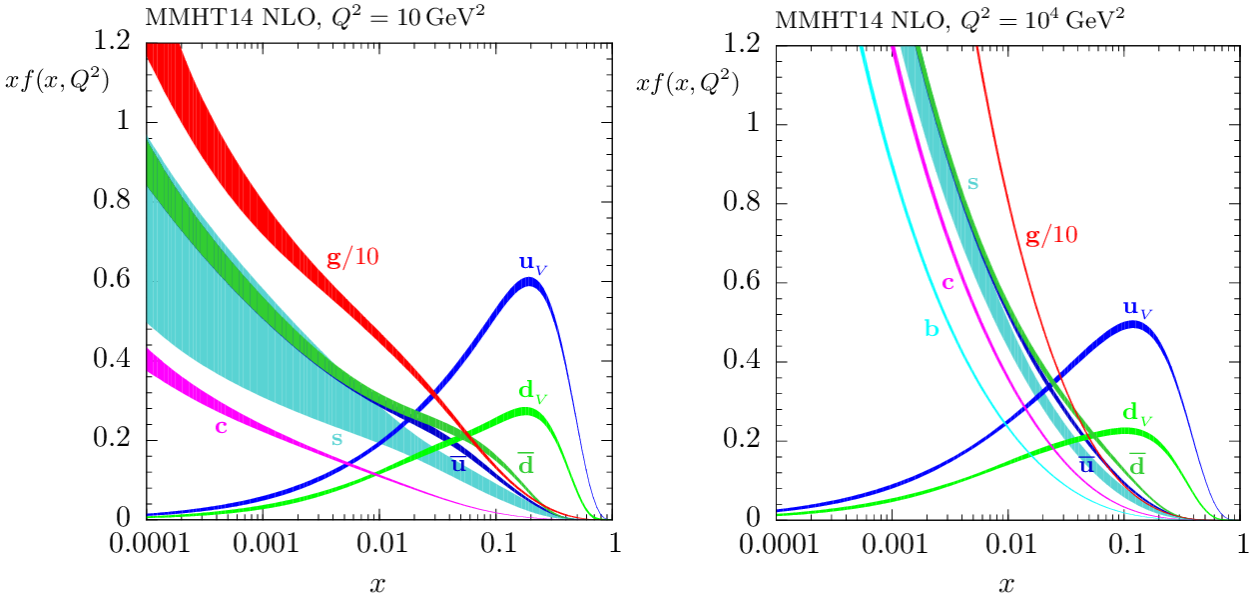


Figure 2.3: Showing the NLO PDFs at $Q^2 = 10GeV^2$ and $Q^2 = 10^4 GeV^2$ Figure from [39]. Note the factor x on the y-axis

2.2 *Decay chains and Feynman diagrams*

The decays chains which are of interest in this thesis are listed below.

$$\begin{aligned} H &\to \gamma\gamma^* \\ H &\to Z\gamma \\ H &\to \gamma\gamma \\ H &\to ZZ^* \Rightarrow eell \\ H &\to HH \Rightarrow bb\gamma\gamma \\ Z &\to ee \\ Z &\to \gamma\gamma \\ Z &\to l\gamma^* \\ Z &\to \mu\mu\gamma \end{aligned} \tag{2.2}$$

[6] H = Higgs, Z = Z boson, γ = photon, e = electron/positron, l = lepton, b = bottom quark, * = indicates off-shell particle

Section 2.5 will try to explain why we are interested in these specific decays. These different decay chains span a wide range of energies and final state particles, for a large part of them we have the inclusion of the Z boson. The Z boson has a very short mean lifetime, roughly $3 \times 10^{-25}\,s$, this leads to a width on the mass of the Z boson

through time energy uncertainty principle.[7][11].

$$\Delta T \Delta H = \frac{1}{2}\hbar \tag{2.3}$$

The Z boson mass appears to follow a Breit-Wigner distribution

$$\frac{dN}{dm} = \frac{N_{tot}}{\pi} \frac{\Gamma/2}{\Gamma^2/4 + (m - m_0)} \tag{2.4}$$

with mean $m_0 = 91.1876 \pm 0.0021\,\text{GeV}$ and *Full-width* $\Gamma = 2.4952 \pm 0.0023\,\text{GeV}$[77]

The Z boson is neutral and as a result the charge of the decay products too must be neutral following charge conservation [8], which leads to decay products consisting of particle and anti-particle pairs. [9]

These decays are not treated equally and some happen more frequently than others, the probability for a process to occur is calculated through the *cross-section*. The fraction of total decays that one decay channel constitutes is called the *branching ratio* and is determined by the width of the specefic *final state*, here the i'th branching ratio $Br_i = \Gamma_i/\Gamma$.

A selection of Feynman diagrams showing Higgs boson production is shown in Figure 2.4. Looking at (c), reading from left to right, we see an incoming quark and anti-quark pair annihilating in order to create either a W or Z boson which in term produces a Higgs through an effect known as Higgs-strahlung.

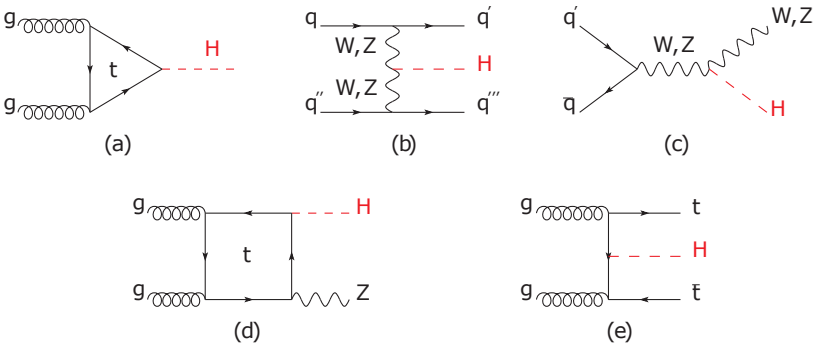


Figure 2.4: Figure from[77] (Cropped) Feynman diagrams showing Higgs production from (a) gluon fusion, (b) Vector-boson fusion (c) Higgs-strahlung, (d) associated production with gauge boson, (e) associated production with a pair of top quarks

2.3 *Quark Color confinement.*

As previously mentioned quarks are confined in hadrons, and can therefore never be observed on their own. The explanation for this confinement is to be found in the potential of the strong force, which is linearly increasing with the distance between the quarks.

This energy comes from what is usually called a gluon flux tube, the energy of this tube per unit length is roughly $k = 1GeV/fm = 0.2GeV^2$, and the potential can be considered to be linearly increasing with distance $V(r) = kr$. A small Coulumb term should be included but is considered negligible [8]. This means that if energy is applied

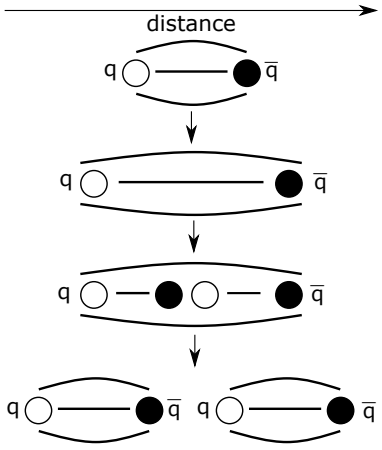


Figure 2.5: Illustration of a meson being *stretched* ie. provided energy and as result spontaneously creates a new $q\bar{q}$ pair which then form a new meson.

[7] The more controversial counterpart to the position-momentum uncertainty principle since time is not well defined and there does not exist a "time" operator.

[8] Here we are considering all charge not just electromagnetic charge.

[9] In some cases with additional neutrally charged particles, usually photons.

to quarks seperating them, the potential energy of the gluon flux tube increases linearly, until it becomes energetically advantageous to create new set of hadrons, at this point the potential energy will spontaneously convert into mass, i.e. new quarks, and new hadrons are formed. An illustration of this hadronization can be seen in Figure 2.5. It should be mentioned that there are two competing models for predicting hadronization. In this thesis we will be using the Lund string model, as it is the one used in Monte Carlo (MC) event generator PYTHIA[67], which is used for ATLAS' MC event generation.

2.4 *Particle interaction in matter*

In this project we will mainly be looking at end products of photons and electrons, however these electrons and photons will when originating from a collision in ATLAS will be very energetic (Magnitudes of MeV or larger).

2.4.1 *Bremsstrahlung*

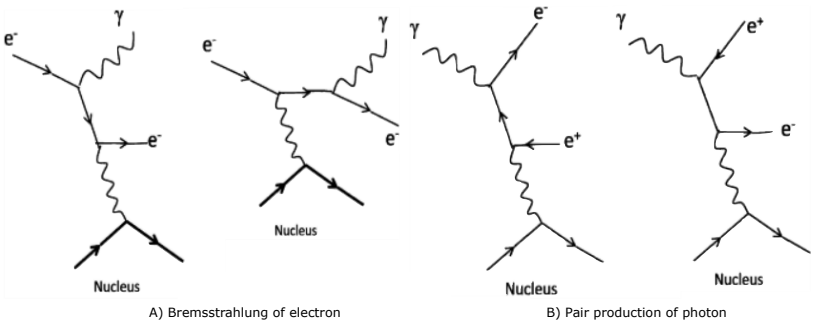


Figure 2.6: A shows Bremsstrahlung of an electron while B shows the photons pair-production of an electron positron pair. Figures from [50]

[10] π^0 also have some interaction through bremsstrahlung, but it is not a dominating effect.

The dominant effect for electrons and photons [10] at energies present in ATLAS are bremsstrahlung. Since photons are massless and with no charge their bremsstrahlung effect only occurs due to an effect called *electron-positron pair-production* where an electron and positron are created through a photon's interaction with the nucleus of a matter material, which leads to further bremsstrahlung from the daughter particles, creating a cascading effect that results in a shower of particles in the material. For an electron the average rate of energy loss is given through

$$-dE/dx = E/L_R, \tag{2.5}$$

where L_R is a function of Z (the atomic number) and n_a the number density of atoms in the medium. Integrating over the distance travelled one gets

$$E = E_0 exp(-x/L_R), \tag{2.6}$$

And it can be seen that the radiation length L_R is the average thickness of the material that reduces the mean energy of the electron

by a factor of e. For the photon the resulting *equivalent* expression becomes

$$I(x) = I_0 exp(-7x/9L_R). \tag{2.7}$$

The Feynmann diagrams of the processes can be seen in Figure 2.6 [50]

In Table 2.1 radiation lengths and interaction lengths for some materials relevant to ATLAS detector has been listed.

Material	Radiation length	Pion interaction length
Copper	1.436 cm	18.51cm
Lead	0.5612 cm	19.93cm
Tungsten	0.3504 cm	11.33cm
Silicon	9.370 cm	59.14cm
Iron*	1.757 cm	20.42cm

Table 2.1: Table of radiation lengths of materials relevant to the ATLAS detector.[77]. * representing steel

2.5 *Physics beyond the standard model*

So far, I have not explained why we are so interested in increasing available statistics for the Higgs or Z boson, which are the parent particles of the decays we are interested in. This next section will seek to amend that fact. I started of this chapter proclaiming that the SM is the most effective model we have for describing particle physics, the so-called status quo. However, it is not all-encompassing and there are areas that it fails to describe, for an example, how the gravitational force is tied in with the other forces. Therefore, we are looking for phenomena which can not be described by the SM so-called *new physics* or *physics Beyond the Standard Model* (BSM).

In the most general terms we could say that if A_{SM} is the theoretical amplitude of a specific final state from phenomena contained within the SM, and A_{BSM} is then the amplitude of contributions not contained within the standard model. Then we seek to answer

$$|A_{measured}| = |A_{SM} + A_{BSM}|^2 \stackrel{?}{=} |A_{SM}|^2. \tag{2.8}$$

While this equation is general, it does not provide us much in terms of *where* to look. This is decided upon by looking at competing theories, and areas, where they differ from the SM. Some of these competing theories which go beyond the SM predict the existence of a particle coined the *dilaton*, however, this theory predict no self-coupling in the Higgs particle. The reasoning behind this is complicated and grounded in the lagrangian for the standard model. Very briefly, the Higgs potential is of the form

$$V(\phi^*\phi) = \mu^2(\phi^*\phi) + \lambda(\phi^*\phi)^2 \tag{2.9}$$

If $\mu > 0$;the potential will be of the form seen in figure 2.7 which leads to the spontaneous symmetry breaking that gives mass to the matter particles and the massive bosons. In the resulting lagrangian

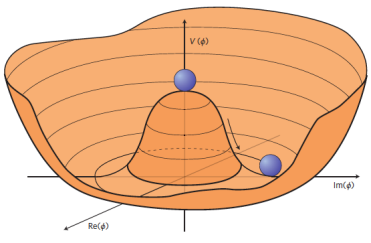


Figure 2.7: showing the higgs potential with $\mu > 0$.[28]

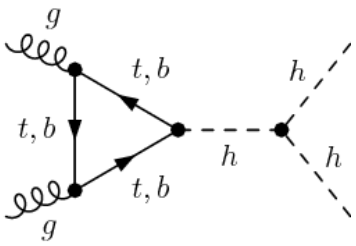


Figure 2.8: Example of a Higgs decay, with a point like self-coupling decay node, as predicted by Electroweak symmetry breaking. Figure from [24] (cropped)

[11] Only in MC as real data does not exist.

there is also a quartic term which the standard model predicts as self-coupling of the Higgs, however if this is the case then the dilaton particle cannot exist [33]. That means with enough statistics, one should be able to see the contribution from decays which include a Higgs self-coupling node like the one seen in figure 2.8.

The decays listed in 2.2 all include either the Higgs or the Z-boson, the Z-boson is also included in one of the Higgs particle decays, namely $H \to z\gamma$. Initially the goals for which data should be tested were high, it was hoped that the thesis could cover both MC and real data for $Z \to ee$, $z \to \mu\mu\gamma$ and $H \to \gamma\gamma$[11] The inspiration for these lofty ambitions were that they had been tested by Malte Alghren in his thesis about CNN performance in ATLAS [6], what was not considered was that his starting point for his thesis were a fully functional CNN model developed for ATLAS data. It would slowly over the course of the project become apparent, that developing a functional graph neural network along with the data pipeline to make it possible was enough on its own and therefore the $Z \to ee$ decay channel was chosen, as it is considered the easiest. However, the model has been constructed with both expanding to other decay channels and evaluating on real data as well as training in real data in mind, and not much further work needs to be done to test these scenarios.

2.5.1 *Energy resolution of the daughter particles.*

Both the Higgs and Z mass increase in precision with an increase in precision of the daughter particles, but while the Z-boson has a full-width of $\Gamma_Z = 2.4952 \pm 0.0023\,\text{GeV}$ the width of the H-boson is much smaller with an upper limit of only $\Gamma_H < 0.013 GeV, CL = 95\%$ [77], why an increase in resolution of the daughter particles of the Higgs decay will continue to bring about an increased resolution for the Higgs particle.

3 Detector physics

This section aims to give the reader an understanding of the mega experiment at *Conseil Européen pour la Recherche Nucléaire* more commonly known as CERN. While some explanation of different detector sub-units will be given, it will be no deeper than what is necessary to understand the data used in this project.

CERN is a research institution which houses numerous accelerators. The accelerators are linked such that, each accelerator accelerates a beam of particle bunches before injecting it into the next more powerful accelerator in the sequence. The final and most powerful accelerator in this sequence is the Large hadron collider (LHC). Most of the accelerators have their own experiment units or detectors.

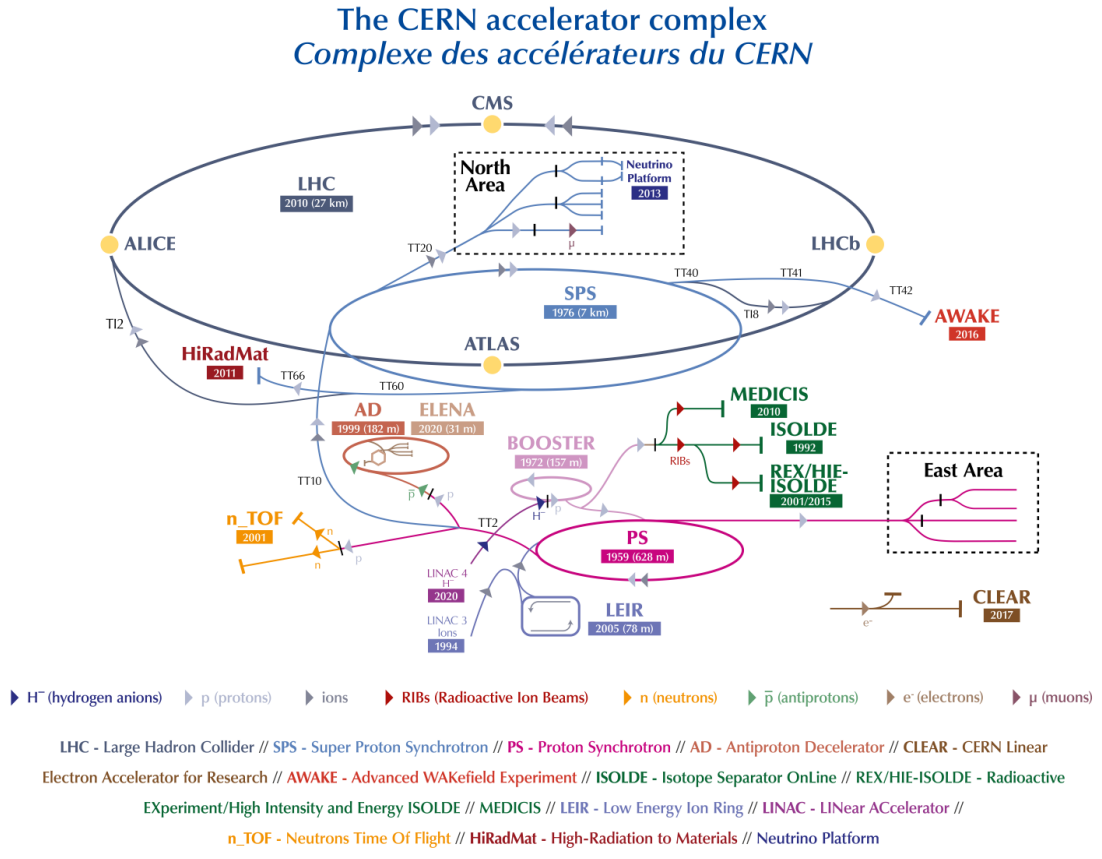


Figure 3.1: The accelerator complex at CERN showing the injections of the accelerators into the next one in the sequence.

The LHC has 4 main experiments: Compact Muon Solenoid (CMS), A Toroidal LHC ApperatuS (ATLAS), A Large Ion Collider Experi-

ment (ALICE) and LHC beauty (LHCb), where the accelerated particle bunches are put on collision course.

The inarguably most well known discovery made at CERN is the joint discovery of a neutral boson with a mass of about $125GeV$ in 2012 by ATLAS and CMS. [20][15]

3.1 *The Large hadron collider and ATLAS*

With a circumference of roughly 27 km the LHC is the largest and most powerful accelerator at CERN. The LHC has so far seen two *runs* with the third scheduled for startup later in 2022. In 2018 the second run finished and within ATLAS proton-proton collisions with Center of Mass (CoM) energies at $\sqrt{s} = 13TeV$ were studied. When in operation during run 2, *particles bunches*[1] collided every 25 ns in the center of the ATLAS collision chamber. The experiment reached a time integrated luminosity of $L_{int} = 146.9\text{fb}^{-1}$[3]. The time integrated luminosity is as the name suggest a time integration over the luminosity, luminosity is defined as

$$L = \frac{1}{\sigma} \cdot \frac{dN}{dt}, \tag{3.1}$$

where σ is the cross-section. Unfortunately an increase in luminosity also brings about an increase in *pileup* equating an increase in noise. Pileup is quantified by the mean number of interactions per crossing. Pileup is the amount of extra data unrelated to the interaction of interest. It can be split into two parts, in-time pileup and out of time pileup referring to whether the extra particles are from the same bunch crossing or an earlier bunch crossing. As luminosity will be increased in run three and drastically in run four the noise from pileup is considered one of the largest challenges for currently implemented algorithms.

[1] Particle bunches contain upwards of 10^{11} particles

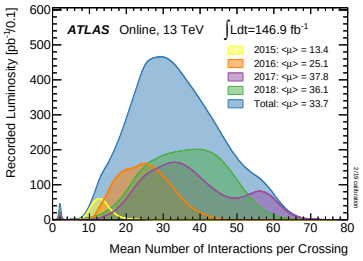


Figure 3.2: A plot showing the recorded luminosity as a function of the Mean number of interactions per crossing, which quantifies pileup. Figure from [48]

Detector and coordinate-system

The detector is shaped like a cylinder around the beam pipe. We define a coordinate system where z is in the direction of the beam pipe. x points inwards towards the center ofs the LHC and y pointing upwards. Due to the symmetrical nature of the detector, the azimuth ϕ is often used to describe the direction in the xy-plane. Furthermore, θ is used to derive the Lorentz boosting invariant feature η

$$\eta = -\ln\left(\tan\left(\frac{\phi}{2}\right)\right). \tag{3.2}$$

η is defined in the zy-plane and becomes infinite when directed along the z-axis and 0 along the y-axis. See Figure 3.3

The detector consists of five submodules which can be further subdivided. Four of these modules are, from closest to the beam pipe to farthest, in the xy-plane: The Inner Detector (ID); the Electromagnetic CALorimeter (ECAL); the Hadronic CALorimeter (HCAL) and

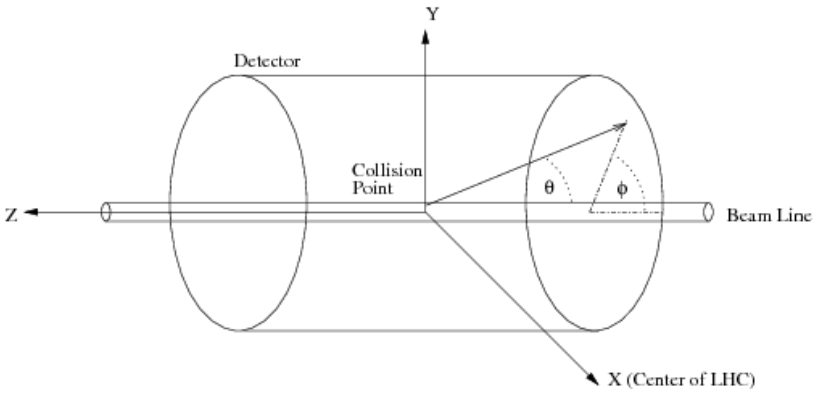


Figure 3.3: Diagram showing the coordinate system used to describe the ATLAS experiment. Figure from [64].

the muon spectrometer. The final module is the ForwardCALorimeter(FCAL), which covers the very forward η region $3.1 < |\eta| < 4.9$.

3.2 *Inner detector*

The purpose of the inner detector is the track reconstruction of charged particles. We differentiate between two different areas of the ID, the barrel which surrounds the beampipe and then the two endcaps which are perpendicular to the beampipe and situated on both ends of the barrel unit. This configuration gives coverage in $\eta < 2.5$. The ID tries to minimize the amount of energy deposited in the unit such that the calorimeter will be able to most accurately reconstruct the energy of the created particles. It consists of a Silicon pixel detector, a SemiConductor Tracker (SCT) and a Transition Radiation Tracker (TRT). The two semiconductor based modules achieve low amounts of deposited energy by having a small band gap between the valence band and the conduction band ($1.21eV$)[36]. This results in a cost of $3.6eV$ in order to liberate an electron for readout. For the TRT, which uses a noble gas, the price to liberate an electron is $\sim 30eV$

A Solenoid Magnet situated between the ID and the HCAL covers the ID in a magnetic field of $2TeV$, which bends the charged particles, allowing for a reconstruction of the path or *track* taken by the particle. Charge and transverse-momentum estimations can then be made through,

$$p_T = \frac{r}{q \cdot B}, \tag{3.3}$$

where r is the trajectory radius of the particle

3.2.1 *The Pixel detector*

The silicon pixel detector consists of semiconductors placed in 4-layers in a circular pattern surrounding the beam pipe as seen in figure 3.4. The units are slightly angled compared to the tangent of the beampipe cylinder and are overlapping in order to ensure full coverage in ϕ. The detector consists of 1736 modules plus an additional

288 modules in 3 disks in the endcap, totaling 92 million pixels. The 3 outer layers have pixels of size $50 \times 400 \mu m^2$. The newer innermost layer (The Insertable B-Layer (IBL)) have pixels of size $50 \times 250 \mu m^2$ and was intended to better identify *"long-lived"* particles e.g. b/τ. A schematic showing the the different layers in the pixel detector can be seen in figure 3.4

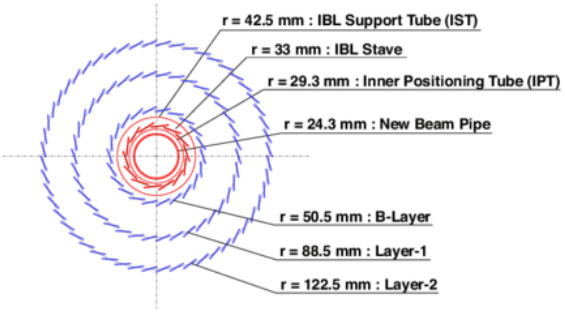


Figure 3.4: A schematic showing the construction of the pixel detector [58]

3.2.2 *The SCT*

The SCT, like the pixel detector, uses semiconductors. It consists of 4088 modules assembled from 4 rectangular single sided silicon micro-strip sensors, two pairs are *daisy-chained*[2] together, resulting in a sensor of approximately 12 cm in length, and an identical pair is then placed in a back to back configuration with a stereo angle of 40 mrad. This is done to save on number of readout channels (totaling ∼6 million), whilst keeping good resolution in the azimuth, at the cost of precision in the z-axis. Saving number readout channels reduces both mass in the ID and monetary cost.

[2] Nomenclature from electrical engineering meaning that the units are wired in sequence

The sensors are situated in 4 barrel layers surrounding the pixel detector and 9 disks in each endcap with units perpendicular to the beam-axis. In the barrel layer the modules are placed with a slight angle to the tangent of the beam-pipe cylinder, and overlapping with a few millimeters in order to ensure full coverage in the azimuth [18]. A schematic of the SCT can be seen in figure 3.5

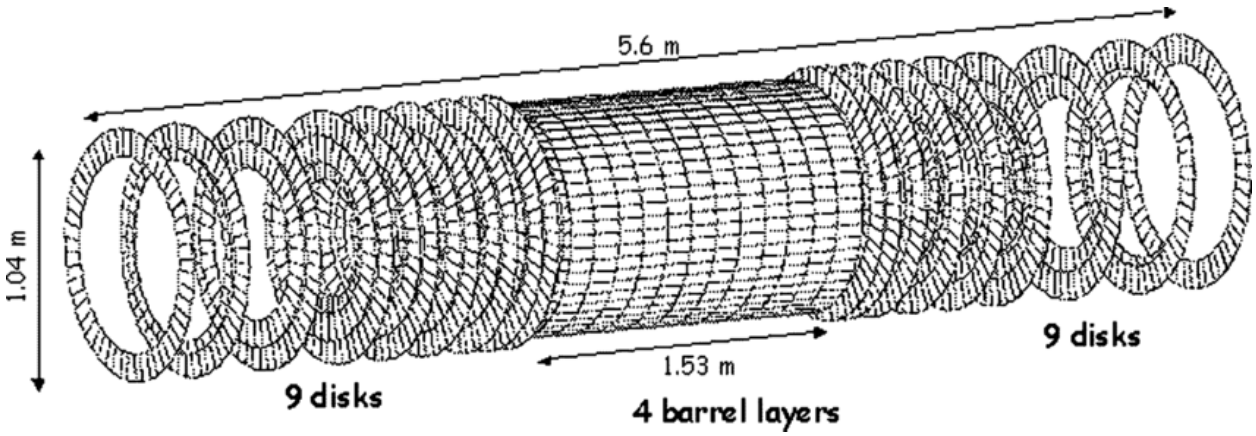


Figure 3.5: A schematic of the ATLAS SCT. Figure from [65]

3.2.3 *The TRT*

The final part of the ID is the TRT. It is built with less precision but gives more continuous readout of a track averaging ∼35 hits per track. It consists of hollow straw-tubes with a 4 mm diameter with centered 0.03 mm gold-plated tungsten-wire. A high voltage is induced between the tube and the center wire and a gas mixture of $Xe/Ar/CO_2/O_2$ flows through the tube. A charged particle will ionize the gas which creates a measurable current that can be readout. The TRT has considerably less readout channels when compared with the two semiconductor based modules with only 350.000 readout channels. Like the other modules, it is split into a barrel and endcap region, with 50.000 straws of length ∼144 cm in the barrel and 250.000 shorter straws of length ∼39 cm in both endcaps. The TRT data is especially used for particle identification tasks [71]. An overview of the specifications of the ID can be seen in table 3.1

	Intrinsic accuracy (μm)	readout channels	Average # hits†
Pixel detector	10×115 in $(R-\phi)\times z(R)^*$	80.4M	4
SCT	17×580 in $(R-\phi)\times z(R)^*$	6.3M	8
TRT	130 in $(R-\phi)$	0.35M	36

Table 3.1: Summary of ID specifications. (†per track.) (* in endcap.) [4]

3.3 *The calorimeter units*

While the ID tries to track the particles without interference, the calorimeters are sampling detectors, which mean they are built to ensure that the particles are completely stopped[3] within the calorimeter units. This happens primarily through *bremsstrahlung* effects which were discussed in section 2.4. The calorimeters consist of an active and a passive material .The particles passing through the calorimeter interact with the passive material creating showers of particles which are measured in the active material allowing for energy estimations. Once again the structure of the calorimeter units are split into a barrel and endcap part.

[3] deposits all their energy

3.3.1 *The electromagnetic calorimeter*

The ECAL ensures the deposit of energy from *lightweight* particles such as electron and photons (massless), while heavier particles along with neutrinos [4] mostly pass through without detection. The barrel region covers an η area between $0 \leq |\eta| < 1.475$ and the endcap covers $1.375 < |\eta| < 3.2$. The active material in the ECAL Liquid Argon (LAr) and the passive material or absorber material is lead[5]. The absorbers have an accordion shape and are interleaved with readout electrodes. This allows for several active layers in R, three in the precision region ($|\eta| < 2.5$) + a pre-sampler ($|\eta| < 1.8$) all shown in Figure 3.6.

[4] neutrinos are for the most part not detectable in the ATLAS experiment

[5] See table 2.1

The crack region

Output channels as well as power input and cooling have to enter/exist the detector somewhere. This is done within the $|\eta|$ range of roughly $|\eta|_{crack}[1.37 - 1.52]$. Here, the resolution is lower than other in areas of the detector. There is also a larger amount of energy escaping the detector undetected in this region, which is why many experiments avoid using data from the crack region entirely.

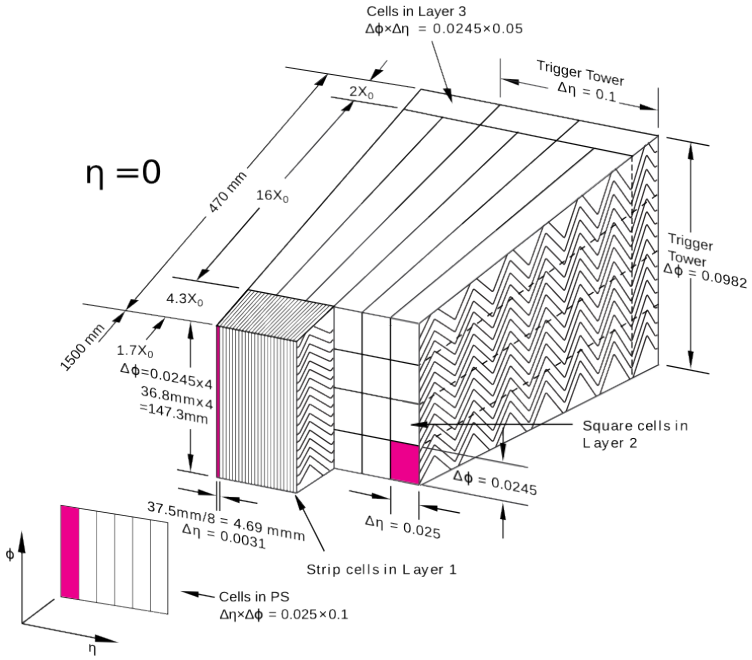


Figure 3.6: Schematic of the ECAL barrel module, showing the accordion shape along with the differing granularities of the 4 different layers.[45][4]

3.3.2 *The hadronic calorimeter*

The central part of the hadronic calorimter system is called the Tile calorimeter as it is constructed of tiles of steel plates[6] and plastic scintillator tiles. It consist of a barrel and extended barrel part. The barrel operates in < 1.0 and the extended barrel in $0.8 < |\eta| < 1.7$. A schematic of the hadronic tile calorimeter can be seen in 3.7

The endcap part of the hadronic calorimetry system is a Copper/liquid-argon sampling calorimeter with a flatplate design. It consists of two wheels in each endcap (4 total), it operates in the $1.5 < |\eta| < 3.2$ range. [4]

An overview of the differing granularities of the central calorimetry system is given in table 3.2.

[6] See table 2.1 for information about the Pion (Representing hadrons as a whole and) interaction length with iron substitution for steel alloy

3.3.3 *The muon system (MS)*

The final layer of the detector is not of much interest in this project, but should be mentioned briefly. The only particles to reach the Muon system are muons and neutrinos due to their low interaction rate. Neutrinos continue without interacting the in the (MS)[7]. The

[7] presence of neutrinos is measured almost solely on missing transverse momentum

Table 3.2: Granularities of the layers within the central ($|\eta| < 2.5$) calorimeters. The differing granularities pose a problem for a CNN algorithm, however using a GNN circumvents this problem, more on this in chapter Graph Neural Networks. Table borrowed from previous master student Malte Algren, input data originally from ATLAS codebase

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

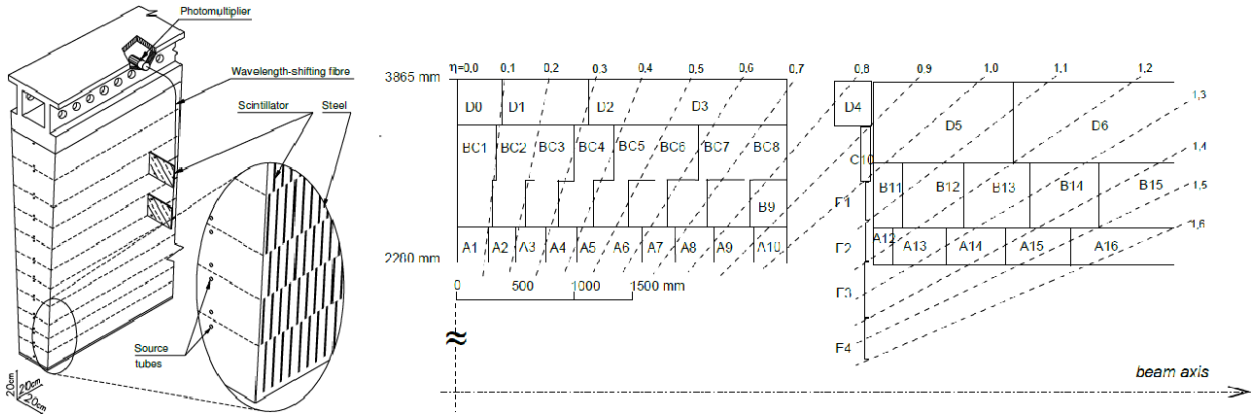


Figure 3.7: A schematic of the hadronic Tile calorimeter. Figure from [7]

muons are bend in ϕ using a 3.5 T magnetic field which allows for a momentum estimate through eq. 3.2. The MS operates in $|\eta| < 2.7$.

4 Reconstruction

Now that we are familiar with the detector it makes sense to look at how this data is processed at the ATLAS collaboration. This section will give a brief overview of how ATLAS processes the data determining clusters in the calorimeter, reconstructs tracks in the inner detector, and produce particle identification as well as energy regression. The section will focus on the reconstruction of electrons and photons.

4.1 Triggering system

Before even starting the reconstruction the data has to be slimmed down to only contain possibly interesting events. The first trigger (L1) system is a hardware implementation that works in real time on a subset of detector information and reduces the event rate to a design maximum of 100kHz from up to 40 MHz[2].[1] The data is then passed on to the L2 trigger and the High Level Trigger (HLT), which is a software implementation, that makes use of about 40 - 50 thousands processing units. It reconstructs on either full-detector information or uses Region of Interest (RoI) information identified in the L1. The HLT stores the information for offline analysis at a rate of about 1 kHz. [76]

[1] Based on the 2015 proton-proton collision data with event rates in run 2 being even higher

4.1.1 Calorimeter clustering

Clusters of energy depositions are found in topologically connected calorimeter cells. These clusters are called topo-clusters. The process starts with a cell significance ς^{EM}_{cell} breaking a predefined noise threshold and becoming a *proto-cluster* initializing cell. The requirement is

$$|\varsigma^{EM}_{cell}| = \left| \frac{E^{EM}_{cell}}{\sigma^{EM}_{cell}} \right| \geq 4, \tag{4.1}$$

where E^{EM}_{cell} is the cell energy and σ^{EM}_{cell} is the expected cell noise, which is a combination of known electronic noise and pile-up noise expected from the instantaneous luminosity in run 2. The pre-sampler and first LAr EM layer are excluded from the generation of *proto-clusters*. Neighboring cells surpassing a significance of $|\varsigma^{EM}_{cell}| \geq 2$ are collected into the cluster and become seed cells for the next iteration. If *proto-clusters* share a cell with significance above the threshold the *proto-clusters* are merged. A crown of nearest neighbors is then added

to the clusters independently of their energy. *Proto-clusters* can be separated if they contain two or more local maxima[2]. [27]

[2] The requirements for a local maxima are at least four neighbors of which none have a larger signal and a $E_{cell}^{EM} > 500MeV$

4.2 *Track finding*

First the raw ID and TRT data are turned into space-points, through a clustering of the readout. Hereafter, seeds are generated; seeds are combinations of three space points. This maximizes the candidates while still allowing for a rough momentum estimate. The seeds are extended with additional space-points that are compatible with a perfect helical trajectory in a uniform magnetic field. Multiple track candidates can occur per seed. Following this is a stage of ambiguity solving, which seeks to deal with track candidates overlapping and using the same clusters. The algorithm also seeks to identify *merged clusters*, i.e. clusters containing charge deposits from multiple particles, and *shared clusters* which are clusters used in multiple track candidates that do not fulfill the requirements for a *merged cluster*. The tracks are scored on different features such as $\chi^2 - fit$, penalizing tracks with a bad fit.

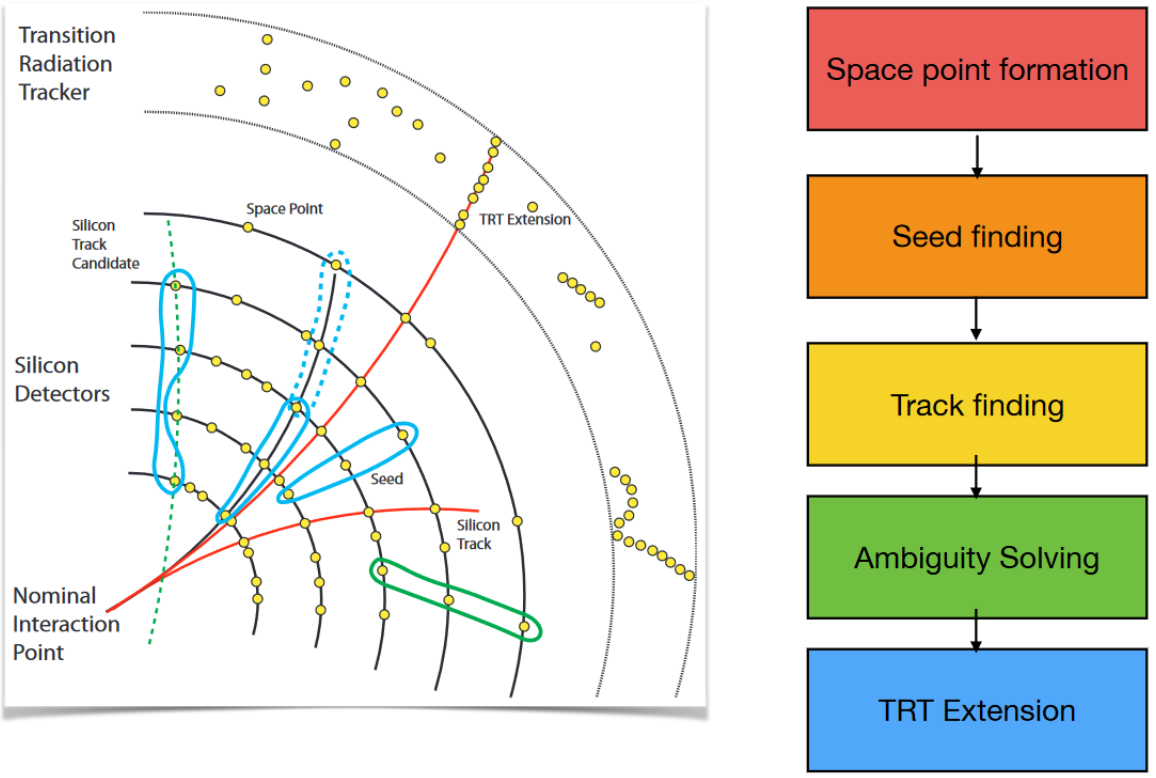


Figure 4.1: Diagram showing a simplified overview of the track reconstruction algorithms and workflow. Figure from [35].

The ambiguity solver rejects tracks that do not meet the following criteria, solving for higher scoring tracks first.

- $p_T < 400MeV$,

- $|\eta| < 2.5$,
- Minimum of 7 pixel or SCT clusters,
- Maximum of 1 shared pixel cluster or two shared SCT clusters on the same layer,
- no more than a total of two *holes*[3] in the combined pixel and SCT,
- no more than one hole in the pixel detector.
- and two additional requirements to the transverse impact parameter and the longitudinal difference.[4]

[3] holes are expected but missing clusters

[4] more detailed description of these parameters can be found in [19] p.7

The final step extends the track into the TRT region of the detector [19]. An overview of the track reconstruction workflow can be seen in figure 4.1.

4.3 *Matching clusters*

The reconstruction starts from determining energy clusters in the calorimeters. Energy deposits that can be matched with an ID track consistent with the characteristics of an electron originating from the interaction point are classified as electrons. Clusters that cannot be matched to an ID track are labelled as an unconverted photon. Converted photons are a bit more complex and can be determined in two ways. As a cluster matched with a track consistent with the characteristics of a photon conversion in the ID material, or matched to a two-track vertex. There are also rules for how the reconstruction handles ambiguities such as clusters fulfilling both electron candidate and converted photon requirements. [1]. The clustering matching algorithm seems to have some inaccuracies which causes *False* labels even in MC, which use the same algorithm for matching clusters to tracks and thus obtaining the truth energy label. More on this in section 6.2.1.

5 Machine Learning

Since it is *still*, at least for a little while longer, possible to complete an education within physics without ever touching upon machine learning, I will start off by describing some core elements of a machine learning algorithm, how they came about and what kind of tasks they excel at. The chapter will be based on the publicly available textbook *Deep Learning* by Goodfellow et al. [34]. The chapter strives to use the notation described by the book in the preliminary pages.

5.1 Supervised learning

Within machine learning there exists two distinctly different branches, that is *supervised* and *unsupervised* learning. The difference between the two is the existence of labels or a target. A supervised learning algorithm will make a prediction which can then be compared to a label of truth or pseudo-truth[1]. Unsupervised learning does not need any labels but is instead used to order or categorize the data based on structures inherent in the data [2].

[1] More on pseudo-truth labels for training in data in 6.3.2

[2] Examples of unsupervised algorithms are clustering and autoencoders.

For supervised learning, given an input-space X and an output-space Y, we wish to determine a function capable of transforming an input $\mathbf{x}^{(i)} \in X$ to the corresponding $\mathbf{y}^{(i)} \in Y$, for all inputs, that is the function $f(\mathbf{x}; \boldsymbol{\theta})$ (denoted h for hypothesis) takes $h : X \to Y$. In a probabilistic setting we assume the existence of an underlying joint probability distribution $P(\mathbf{x}, \mathbf{y})$ and wish to determine h such that it matches the conditional distribution $P(\mathbf{x}|\mathbf{y})$. According to the principle of risk minimization originally from [73], the optimal function h is the one that minimizes the *Risk* defined as,

$$R(h) = \mathbb{E}[\ell(h(\mathbf{x}, \mathbf{y})] = \int \ell(h(\mathbf{x}, \mathbf{y}))dP(\mathbf{x}, \mathbf{y}) \tag{5.1}$$

Where ℓ is the *loss function*, that describes the distance between the "truth" $\mathbf{y}$ and the by h predicted value $\hat{\mathbf{y}}$.

Unfortunately, we do not know the true underlying probability distribution, as such we estimate the risk through the empirical risk, henceforth called loss and denoted $\mathcal{L}$.

$$R_{emp}(h) = \frac{1}{N}\sum_{i=1}^{N} \ell(h(\mathbf{x^{(i)}}, \mathbf{y^{(i)}})) \tag{5.2}$$

Requiring a finite set of example data-target pairs. $S = \{\mathbf{x^{(i)}}, \mathbf{y^{(i)}}\}_{i=0}^{N}$ independently drawn from $P(\mathbf{x}, \mathbf{y})$.

5.1.1 *Over- and underfitting.*

While it for a modern computer is quite a simple matter to find the minimizing function $h^* = argmin_h(\mathcal{L}(h, S))$ it is unfortunately not what we are looking for. It is easy to imagine a model that has such high *complexity*[3], that it is simply able to memorize every input-output pair which would lead to minimizing the loss. However the real objective is not a minimization of the empirical risk but rather the true risk. While one could set out to determine the error between empirical risk and the true risk, often called the *generalization error*

$$G(h) = R(h) - R_{emp}(h), \tag{5.3}$$

there is in most cases a much more practical method. That is splitting the data into independent and identically distributed (iid) sets one for training and one for validation. [4]. This grants us a tool with which we can inspect this generalization and balance the model optimally between over- and underfitting. Figure 5.1, shows how increasing the complexity of a model will lead it to learn the statistical fluctuations of the finite training dataset. However too *simple* of a model will not be able to capture all the underlying structures of the data.

When we apply this method we run into the issue of once again being biased towards the train-validation set combination. This is why we hold out on a test set, which will never be included as part of the training or optimization of the model, for unbiased evaluation.

We can estimate the bound of the generalization error through the Probably Approximately Correct (PAC) theory and the Hoeffding's inequality to be,

$$G(h^*) = R_{emp}(h^*) + \sqrt{\ln\left(\frac{2M/\delta}{2N}\right)}. \tag{5.4}$$

[40][53], where N are the data points available in the training, h^* is the best model among M models tested. M increases with the complexity of the model as more learnable parameters increase number of feasible models. δ is the confidence of the model. it can be seen that complex models are possible as long as we have large amounts of data.

[3] Read many trainable parameters.

[4] We are actually splitting the data into a training, validation and a test-set

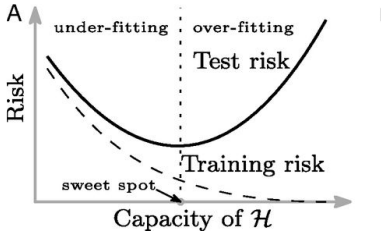


Figure 5.1: Showing the diverting nature of the risk in the training and test iid sets, often called the *bias variance tradeoff*. Figure is a snippet of a figure from [9], which questions whether this holds for large models such as deep neural networks

5.2 *Neural Networks (NN)*

With the most basics of fundamentals in place, the next section will relate to the history and theory of NNs, along with more detailed explanations of the models related to this project.

5.2.1 *Early algorithms*

The earliest algorithms were a linear combination of weights on a feature space $f(\mathbf{x}, \mathbf{w}) = w_1 \cdot x_1 + ... + w_n \cdot x_n$. The resulting number could then be used as a binary classifier[5] depending on whether the

[5] A binary classifier simply splits data points into two different groups e.g. healthy v. sick, cat v. dog or car v. not-car etc. (The last example is called a one v. all)

resulting number was positive or negative. The weights initially had to be set by a human operator. The two next natural evolutions of the science were the inclusion of algorithms that could automatically set the weights and algorithms that return a real number allowing for regression tasks.[6]

The *training* of the weights were accomplished using an algorithm called *stochastic gradient descent*[61][44]. This algorithm or modified versions of it are still the status quo for training algorithms within machine learning today.

The deep learning field has over time seen increasing and waning interest, as a result of scientific breakthroughs, unrealistic claims and the failure to deliver on those claims, as well as the increase in computation hardware and data availability.

The deep learning algorithms are, when compared to other machine learning counterparts, computationally expensive and usually requires a large amount of data for training.

[6] Regression task are simply a task where instead of trying to group the input, we wish to use the input features to determine a related feature not part of the input features e.g. housing prices or survival rates.

5.2.2 *Alternatives to deep learning and currently implemented methods*

ATLAS has historically opted for *Likelihood* based algorithms or *tree based* algorithms. One could, with good reason, ask oneself if these algorithms have been around for so long, why have they not previously been used for energy regression tasks at ATLAS. The answer is in part the conservative nature of large organizations as well as the maturity or rather immaturity of the deep learning field at the time when the Likelihood method was chosen, as well as limitations of the hardware when considering the choice between Boosted Decision Trees (BDT) and more complex models deep learning models.

The choice of using tree based algorithms seems very reasonable as these algorithms still in part dominate the machine learning space especially when it comes to tabulated data.[7]

However, improvements in hardware mean a push for the implementation of deep learning methods that while more resource heavy and harder to implement should be able to achieve equal or better performance to these "simpler" methods. Especially, when we wish to include more "raw" detector data such as the calorimeter cell data, more on the specific data structure in Chapter 6.

[7] Tabulated data being different key features with single values each, an example of non-tabulated data is image data.

5.2.3 *Components of neural networks*

Loss function

The loss function which was briefly mentioned in section 5.1, as a function that describes the distance between the predicted values $\hat{\mathbf{y}}$ and the true/labeled target values $\mathbf{y}$. There are a plethora of different functions which fit this description, the most common examples are, the Mean Absolute Error (MAE) [62]

$$\mathcal{L}_{MAE} = \frac{1}{N}\sum_{i=0}^{N} |(\hat{\mathbf{y}}^{(\mathbf{i})} - \mathbf{y}^{(\mathbf{i})}|, \tag{5.5}$$

or the Mean Squared Error (MSE).[63]

$$\mathcal{L}_{MSE} = \frac{1}{N}\sum_{i=0}^{N} ((\hat{\mathbf{y}}^{(\mathbf{i})} - \mathbf{y}^{(\mathbf{i})})^2 \tag{5.6}$$

Network, layers & neurons.

In essence, a NN is simply a function of learnable or trainable parameters. Using a network of the type feed-forward NN coined multilayer perceptron, it consists of layers of *neurons* which are fully connected.[8]

[8] Fully connected refers to each output of the previous layer being fed into all neurons of the consecutive layer.

The output of the neuron is a real number. The function for the neuron i in layer l of the feed-forward NN can be defined as,

$$f_i^{(l)} = \sigma^l \left(b_i^{(l)} \sum_{j=1}^{n(l-1)} W_{ji}^{(l)} f_j^{(l-1)}(\mathbf{x}) \right), \tag{5.7}$$

where $n(l)$ is the number of neurons in layer l and $\sigma : \mathbb{R} \to \mathbb{R}$ is an *activation function* which introduces non-linearity.

The layer can then, using vectors and matrices, be described as,

$$f^{(l)} = \sigma^l \left(\mathbf{W^{(l)}}^\top f^{(l-1)}(\mathbf{x}) \right), \tag{5.8}$$

where $\mathbf{W^{(l)}}$ is the weight matrix of layer l, consisting of real numbers with shape $n(l-1) \times n(l)$ and b^l is the vector of biases.

The chain rule allows for the full network to be seen as a composition[9] of the layers.

[9] combining the function using output of following function as input to the other

$$f(\mathbf{x}) = \left(f^{(L)} \circ f^{(L-1)} \circ ... f^{(2)} \circ f^{(1)} \right) \tag{5.9}$$

An illustration of a fully connected feed forward neural network of the type also used in this thesis can be seen in figure 5.2

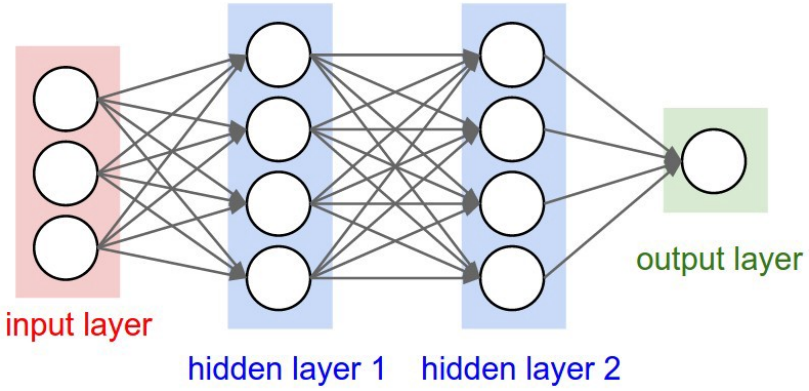


Figure 5.2: A classic illustration example of a fully connected neural network with two hidden layers. Figure from [37].

Activation functions

The *REctified Linear Unit* (ReLU) is one of the most popular activation functions it is defined as,

$$f(x) = max(0, x). \tag{5.10}$$

It was first introduced by Hahnloser et al. in [38], a variant of it coined LeakyReLU,

$$LeakyReLU(x) = max(\alpha x, x), \tag{5.11}$$

where $\alpha < 1$, is currently one of the most widely used activation functions. The sigmoid function,

$$\sigma(x) = 1/1(+e^{-x}), \tag{5.12}$$

is also widely popular, especially as the final output layer in a binary classifier, where the output between 0 and 1 can be a proxy for model confidence or probability for the data point to belong to the signal group.

As previously stated these activation functions introduce non-linearity to the network, without them the network would just be a composition of summations over linear functions which would result in a linear separation line.

Figure 5.3 show graphical representations of the activation functions mentioned in this section.

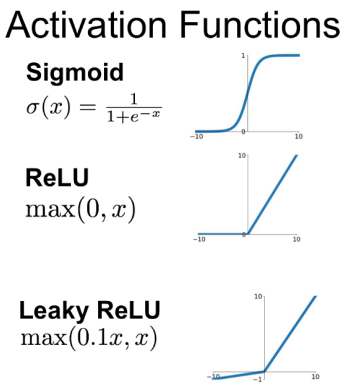


Figure 5.3: Plots of some of the possible activation functions for introducing non-linearity, snippet of figure from [42].

5.2.4 *Training the network*

Backpropagation

We now have all the necessary components of a NN. However missing still, is a method in order to determine the learnable or trainable parameters of the network.[10]

[10] The weights and biases.

This is done using backpropagation, which takes advantage of the chain rule in order to determine the gradient of the network loss described by equation 5.9, that is calculate,

$$\begin{aligned}\nabla_x \mathcal{L} =& (\omega^1)^\top \cdot (f^1)' \ldots \circ (\omega^{L-1})^\top \cdot (f^{L-1})' \\ & \circ (\omega^L)^\top \cdot (f^L)' \circ \nabla_{(f^L)} \mathcal{L},\end{aligned} \tag{5.13}$$

where f^L is the output of the final layer, and ω is the collection of both weights and biases. It makes sense to introduce the partial gradients of layer l, this also accentuates the fact that this process is best calculated iteratively starting from the output layer and moving backwards, hence the name.

$$\begin{aligned}\delta^l =& (f^l)' \circ (\omega^{l+1})^\top \ldots \cdot (f^{L-1})' \\ & \circ (\omega^L)^\top \cdot (f^L)' \circ \nabla_{(f^L)} \mathcal{L}\end{aligned} \tag{5.14}$$

.

We can then use this to define the gradient of weights and biases in layer l.

$$\nabla_{w^l}\mathcal{L} = \delta^l (f^{(l-1)})^\top \tag{5.15}$$

[11] And biases, however biases do not depend on previous input and just uses 1 instead.

The weights[11] are then updated according to this gradient, pushing it in the opposite direction of the gradient. The k'th neurons weight is updated through

$$\begin{aligned} w_{kl} \leftarrow \Delta w_{kl} &= -\lambda \frac{\partial \mathcal{L}}{\partial w_{kl}} \\ &= -\lambda o_k \delta^l \end{aligned} \tag{5.16}$$

Where o is the output of the neuron. The amount of pushing for each step is determined by λ, which is called the learning rate and is one of the most important *hyperparameters* of the network. This method of updating weights is called gradient descent. The stochastic part of *stochastic gradient descent* (SGD) enters, when updating the weight several times per epoch[12] on what is called mini-batches.

[12] An epoch is one full run through all the available data

While the model in this project does not employ the original SDG algorithm, two more modern variants of it have been tested, these optimizing algorithms are known as *optimizers*

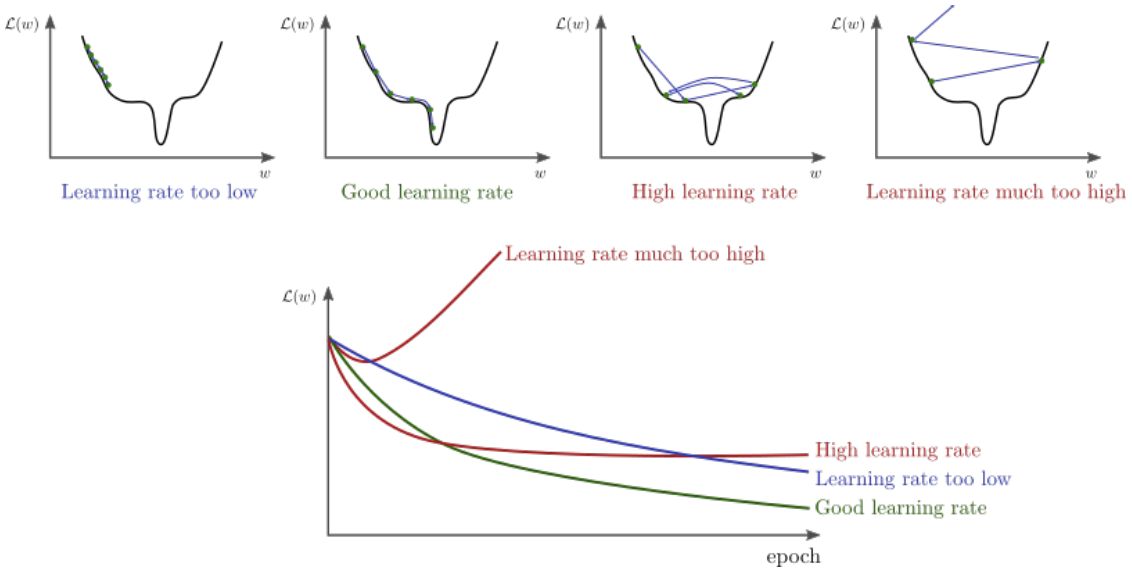


Figure 5.4: Examples of different levels of learning rates. Figures from [22]

Learning Rate

The learning rate, which was mentioned in the previous section, is as stated one of the most, if not the most, important hyperparameter in most machine learning algorithms. Too high of a learning rate will result in the algorithm not being able to accurately converge to a solution and might even cause divergence, while too low of a learning rate can result in the algorithm getting stuck in local minima and extremely slow convergence times. Examples of the different learning rates (in a 2d learning space) can be seen in Figure 5.4

Learning rate schedulers

Instead of having a fixed learning rate, it is often favorable to use a learning rate scheduler to change the learning rate over the course of training. In [68], they argue that cyclical learning rates are beneficial, especially because they allow quicker escapes of *saddle-points* in the loss landscape. A saddle point will have a very small gradient which slows down the learning. Increase the learning rate with a cyclical learning rate to quickly *escape* these saddle-points. It is also argued that if one's ranges straddle the optimal learning rate, then one will, for some time, use the optimal learning rate or a near optimal learning rate. Examples of cyclical learning rates can be seen in figure 5.5

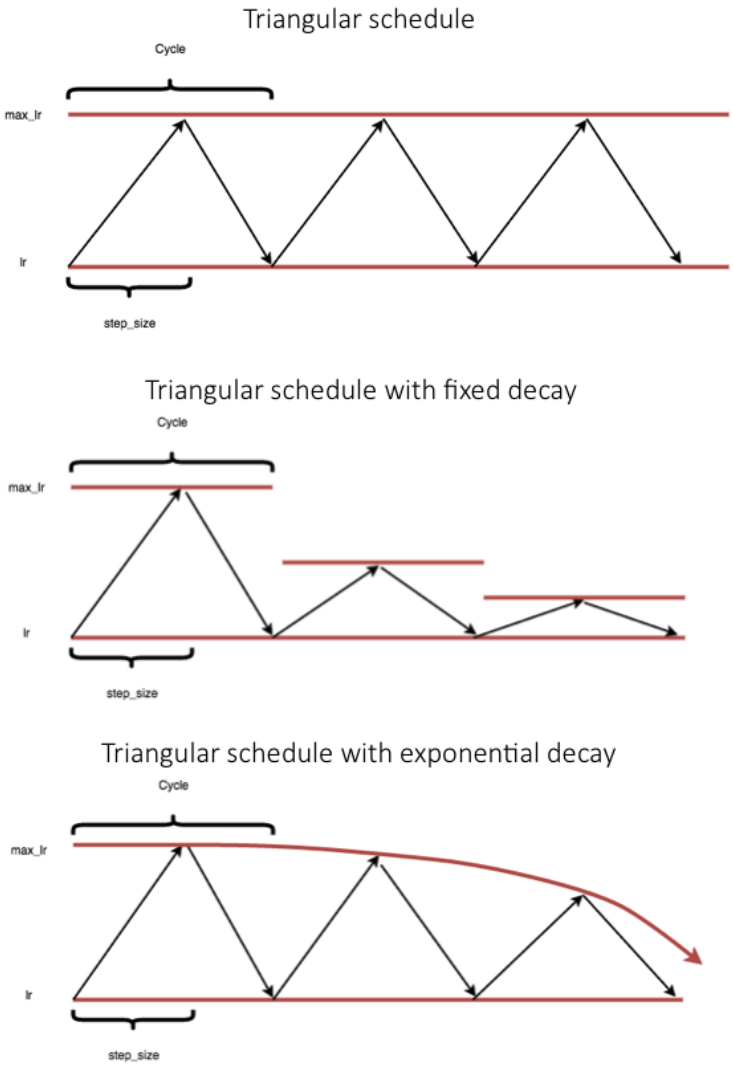


Figure 5.5: Examples of different learning rate schedulers[41].

5.3 Convolutional Neural Networks

As this work builds upon previous students work with Convolutional Neural Networks (CNN), a brief description is required before moving on to Graph Neural Networks (GNN). The section will to a large degree make use of graphical representations when posssible.

CNNs as they are used today were first introduced in 1989 [46], but it did not gather much immediate attention due to the limited computation power of the time and the large amount of data that a CNN requires.

A CNN takes a regular input $\mathcal{X}$, usually an image, and maps a learnable weight matrix called a *kernel* to $\mathcal{X}$. The kernel can be thought of as a *neuron* with dimensions $m \times n$. The input is of constant size $H \times W \times D$, *(height, width, dimension)*. D is the *depth* of the data or usually when considering image data, the number of channels [13]. An example of a 3×3 kernel can be seen in figure 5.6.

[13] most commonly D = 3 for red, green and blue color channels

5.3.1 Convolution layers

The name *Convolution* is in most cases a misnormer as the function most implementations of CNNs use [14] is the cross-correlation function.

[14] Including pytorch

Cross-correlation

$$S(i,j) = (I \star K)(i,j) = \sum_m \sum_n I(i+m, j+n) K(m,n) \tag{5.17}$$

Convolution

$$S(i,j) = (I \star K)(i,j) = \sum_m \sum_n I(i-m, j-n) K(m,n) \tag{5.18}$$

The difference between them being only the signage before m and n. The result is a flip in both axis of the matrix. Eqn. 5.18 is commutative, while Eqn. 5.17 is not. In the rest of this thesis convolution will mean Cross-correlation and refer to the expression in Eqn. 5.17

This operation is much easier to comprehend through visualization with a graphic. Figure 5.6 shows a 3×3 kernel on a $5 \times 5 \times 1$ input. The kernel will then glide over the image with a predetermined stride[15]. In this case the stride is 1, which results in an image of size $3 \times 3 \times 1$.

[15] *stride;* the step size of the sliding *pane* as it goes from left to right and top to bottom much like an old typewriter.

One can perform what is called *padding* (usually *zero padding*), by putting a border of arbitrary values (zero) around the original image, as not to reduce the size of the image.

One convolution layer can consist of several k kernels and the number of learnable parameters is given by.

$$N_{params} = ((C_{in} \times W_{kernel} \times H_{kernel}) + 1) \times k \tag{5.19}$$

Increasing k increases the number of feature maps outputted from the layer. Modern CNN algorithm usually employ several kernels and several convolution layers in the model. [75]

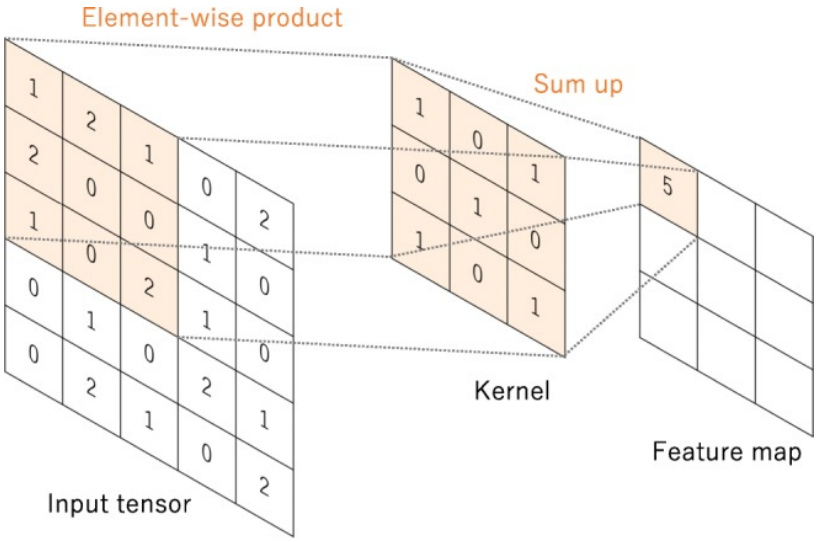


Figure 5.6: An illustration of the *convolution* operation using a 3×3 kernel using a stride of 1 on a $5 \times 5 \times 1$ image. (1 channel could for an example be a grayscale image.) Snippet of a figure from [75]

Pooling layers

In order to decrease the dimensionality of the model as not to fill out the computer memory with parameter weights, a down-sampling operation is often included, one such operation is pooling. A pooling operation has no learnable parameters, it is much like the kernel defined by its height, width and stride. The operation used in the pooling layer is also a hyperparameter. Some common operations are max pooling, which takes the highest value within the *filter*, and mean pooling which passes on the mean. Figure 5.7 illustrates a max pooling operation with dimension 2×2 and a stride of 2 which reduces the size of the feature map by a factor of 2. Aside from the decrease in computational cost, pooling also introduces translational variance. [75]

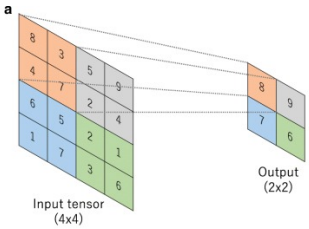


Figure 5.7: Max pooling operation with dimensions 2×2 and a stride of 2 figure from [75].

5.4 Graph Neural Networks

The idea of Graph neural networks was first introduced in 1997 [69] in an attempt to improve neural networks capabilities when dealing with complexly structured data, and varying input sizes[16]. GNN combines the mathematical theory of graphs with neural networks in order to *generalize* the function of CNN to data that is variable in number of inputs and non equi-distant between data-points.

5.4.1 Graphs; a mathematical concept.

Graphs are a mathematical concept described by the aptly named graph theory. This project (luckily) does not require a full understanding of this theory, however some basic knowledge of the defining features of a graph and the notation and nomenclature is necessary in order to explain the function and benefit of GNNs.

A graph is denoted $G = (V, E)$ where V is a set of *vertices*, equivalent to *nodes* or *datapoints*, and E a set of *Edges*. An edge is defined

[16] It is also considered a solution to data in non-euclidean space, although that is not the focus of this project.

by a connection between two endpoints $e = u, v$, here u is said to be a neighbor of v. Graphs can be directed and undirected, a directed graph means that all edges have a direction $e = u, v$ does not necessarily lead to $e = v, u$. Undirected meaning that all edges go both ways, that is if u is the neighbor of v then v is necessarily also the neighbor of u. $d(v)$ is the number of connections to v.

The Adjacency matrix for a simple graph $G = (V, E)$ without self connections can be described through

$$A_{i,j} = \begin{cases} 1, & \text{if } \{v_i, v_j\} \in E \text{ and } i \neq j, \\ 0, & \text{otherwise.} \end{cases} \tag{5.20}$$

An illustration showing a 2d graphical representation of a graph structure and illustrating the difference between direct and undirected graphs can be seen in figure 5.8.

There are two different ways in which one can describe attributes of the graph, *spatial* and *spectral*. The spectral method uses the graphs laplacian eigenbasis, and is dependent on graph structure. The spatial approach, which is the one used in this project, works through operations on *spatially*[17] close neighbors. [47]

[17] Spatially does not necessarily mean in euclidean distance

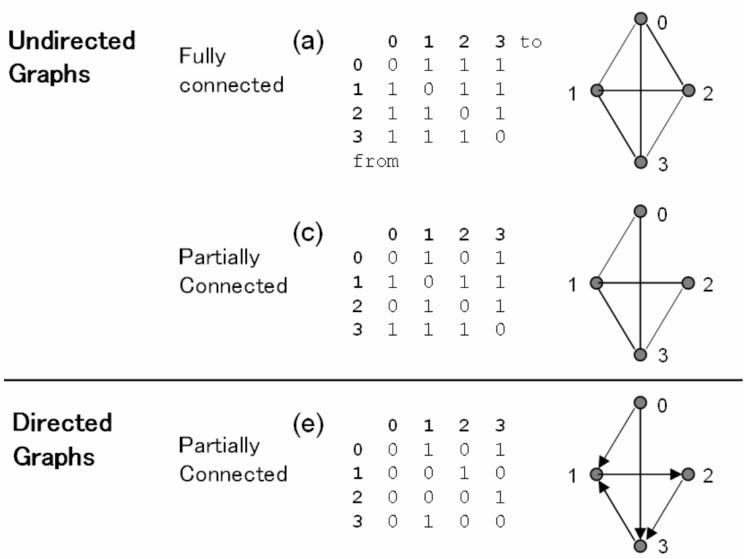


Figure 5.8: Figure from [12] (Cropped), which illustrates the difference between directed, undirected and fully connected versus partially connected graphs.

5.4.2 *Learnable graphs through message passing schemes and updating functions*

In [31] a general framework for supervised learning on graphs was proposed, it is based upon two *phases*, a message passing phase and a readout phase. The message passing phase can be run for T timesteps and is constructed using two sub-functions, a message passing function denoted M_t and a message updating function denoted U_t, where t denotes a given timestep. m_v^t is the message at time t for the specific node v and h_v^t is the hidden state again at time t for node v. The

updating of messages can then be expressed as

$$m_v^{t+1} = \sum_{w \in N_v} M_t(h_v^t, h_w^t, e_{vw}), \tag{5.21}$$

with e_{vw} representing the features of the edge between node v and w. which leads to the updating of hidden states through

$$h_v^{t+1} = U_t(h_v^t, m_w^{t+1}) \tag{5.22}$$

There are many different message and updating functions. The requirements on the functions are that they are differentiable and contain learnable parameters.

The readout phase consists only of the readout function, which is expressed as,

$$\hat{y} = R(h_v^T, |v \in G) \tag{5.23}$$

The readout function looks at the entirety of the final graph state and produces some sort of output.

In this thesis the Dynamic Graph Convolutional Neural Network (DGCNN) method described in [74] is used.

The edge features are described as

$$e_{ij} = h_\Theta(x_i, x_j), \tag{5.24}$$

where x_i is the feature vector of a node in the graph.

with h_Θ being a non-linear function of learnable parameters h_Θ : $\mathbb{R}^F \times \mathbb{R}^F \rightarrow \mathbb{R}^{F'}$

The update of x_i can be described through

$$x_i' = \square_{j:(i,j)\in\mathcal{E}}, \tag{5.25}$$

where $x_j : (i,j) \in \mathcal{E}$ are the neighboring nodes of the *central* node x_i, and $\square$ is a symmetric aggregation operator. The method allows for a choice of (h and $\square$) and in this thesis we are using a fully connected multi layer perceptron (MLP) as the non liner function h_Θ and we use mean as the aggregation operator. Figure 5.9 shows a graphical representation of the EdgeConv operator

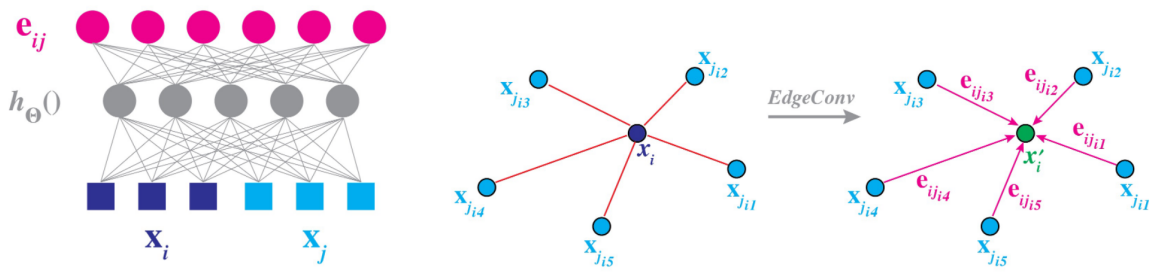


Figure 5.9: Figure from [74], which shows the EdgeConv operator here with a single fully connected layer as h_Θ and 5 neighboring nodes to the central nodes, with 3 features in each of the nodes.

It is relevant to note here that, when $x_i ..., x_n$ represents images on a regular grid and the graph has connectivity which can be described as a patch of fixed size each pixel, with the choice of $\Theta_m \cdot x_j$ as the

edge function and using the sum as the aggregation operator the standard convolution operation described in 5.3.1 is found.

$$x'_{im} = \sum_{j:(i,j)\in\mathcal{E}} \theta_m \cdot x_j, \tag{5.26}$$

with $\Theta = (\theta_1, ..., \theta_M)$ encoding the weights of M different filters each filter having the same dimensionality as x.

In this way the Edgeconv operator on the graph structure can be seen as a generalization of the convolutional method [74].

6 Data Pipeline

6.1 xAOD & DxAOD

The data from the ATLAS experiment is in the order of petabytes. This size introduces many additional challenges compared to experiments with data of a size that can be handled by a normal personal computer. The software framework used to access, reconstruct and simulate ATLAS data is called *Athena* [17]. This project will not go into depth about the ATHENA framework as it has not been a focus of the project. The data used in this thesis, before transformation into graph format, has been procured by previous students Lukas Erhke and Daniel Nielsen with additions from Malte Algren [6], leaving only a few changes to be made for this thesis[1].

[1] such as generating/including [x,y,z] coordinates for the track features

By far the largest part of data simulated or recorded at ATLAS is irrelevant to any specific analysis. Since computation resources are not infinite it is of great importance to create a *derivation* of the full data, which only includes events and features relevant to the specific analysis.

The largest data files that one can use for analysis are called *xAOD* these are meant to contain a wide amount of data for different types of analysis, they can be accessed through the CERN managed database Rucio[2]

[2] Given that you have the authority to do so, data from CERN is *not* publicly available.

The *Derivation Framework* (DF) aims to bring down the size of the data to orders of *gigabytes*, these files are called *DxAOD*. The smaller size allows for local storage and manipulation at tier 3 computers.[3] [23].

[3] tier 3 computers are local supercomputers belonging to any ATLAS collaborator institution

While this model has been developed with usage across different decay channels in mind there has not been time to implement these, therefore only the *DxAOD* called *EGAM1* has been used. This file is created with cuts optimized for, $z \Rightarrow ee$ central electrons. This DxAOD selects central electron pairs based on the Likelihood functions with minimum energy requirements. [26]

6.1.1 DxAOD to Graphs

The EGAM1 ensures that the files are in a workable format, but they still contain more information than what is used by the model, and more importantly the format is currently in a root format. The first step is what is called the *NTuple production.* The NTuple production uses the Athena framework to extract only the necessary information

from the *DxAOD* files, aswell as generate new features if needed. This script has in large part been inherited from previous students, however some additions and modifications have been made in order to extract different track features as well as a rework of the ΔR calculation, which had a faulty implementation of circularity in ϕ.

Initially calculations of (x,y,z) coordinates at the endpoint of the inner detector calculated by using the momentum of the track was implemented, then disregarded in favor of using $\eta, \phi and R$, however towards the end of the project the (x,y,z) coordinates were reimplemented this time instead of using a generated coordinate at the point of entry to the calorimeter the coordinate of the last measurement was used[4]. The different benefits and problems with using either are discussed in section 6.3.1

[4] This was mainly done as the enetry to the calorimetry x,y and z coordinates were found using an approximate method.

The NTuple production takes .root files as input and outputs .root files. The data has now been slimmed down sufficiently and all needed features have been produced.

The next piece of code takes the .root output files of the Ntuple production, where lists of features are ordered in event level structures, and outputs the data in graphs representing each *event*, along with the relevant event wide *scalar features*, ready for prediction or training by the model.

This script has been developed specifically for this thesis with inspiration from the graph production from [78] as well as selection criteria also implemented in [6].

- Event selection.
- Gather statistics for rescaling/normalization.
- Rescale/normalize
- Order things in graph format [5]
- Save to a data format that allows for *"easy"* reading. [6]

[5] Nodes containing track, probe or cell features.

[6] here a pickle format is used, although it seems this is generally disparaged though mostly due to security concerns. The gold standard would probably be to structure the data in an SQL database such as sqlite.[70]

The statistics for the choice of rescaling/normalization scheme is gathered first. Hereafter the data that passes a selection cut, is simultaneously organized into nodes and rescaled and the set of nodes relating to the same *event* is gathered in a graph, the *scalar features* of the same event are collected and rescaled. The energy *labels* and accordion energy are also collected before everything is saved in a pickle format.

The script allows for parallelization and is written such that the files only need to be opened twice, once for gathering statistics and once for everything else.

The final step before the data is ready for training is splitting it into a training, validation and test set. The data is distributed as 70%, 20% and 10% for training, validation and test respectively.

An overview of the entire process and which parts have been developed specifically for this project can be seen in figure 6.1.

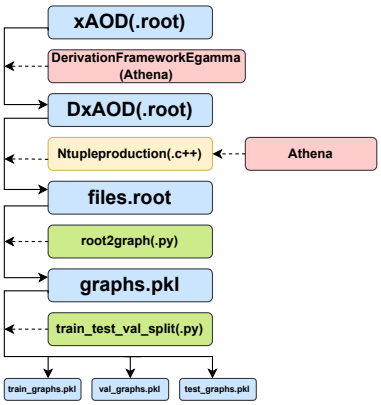


Figure 6.1: Overview of the data pipeline, blue indicates data-files, red indicates CERN produced code, green indicates code developed for this thesis, yellow indicates code developed by CERN, previous students and modified for this thesis.

6.2 *Pre-processing*

A common saying in the machine learning space is *garbage in garbage out.*. This expression is mostly used to warn followers of machine learning against dataism[7] and poorly labelled data. While this section will cover handling of mislabelled data it is also going to cover scaling and transformation of input data which are necessary methods to avoid exploding or disappearing gradients in large neural networks.

[7] blind faith in complex algorithms

6.2.1 *Mislabelled data*

Even though we are training on Monte Carlo generated data, inspection of discrepancies between the $Ecalib^{(BDT)}$ and E_{truth} revealed two possible labelling flaws in the Monte Carlo algorithm. The first one being a mislabelling, where the particle pair in the same event are given the exact same E_{truth}, which statistically would be exceedingly unlikely. These are searched for and removed.

The other group of discrepancies is more speculative, but the suspicion is that the cluster matching algorithm sometimes matches a daughter electron instead of the true initial electron for early decays. This leads to the ATLAS BDT algorithm guessing much lower values compared to the MC truth label. In an attempt to not confuse the algorithm these events are sorted out by making a cut on the ATLAS BDT classification score through

$$keep\,event : |1 - \frac{E_ATLAS^{(BDT)}}{E_truth}| < k, \tag{6.1}$$

where k is a tuneable hyperparameter. the result of the selection criteria can be seen in figure 6.2

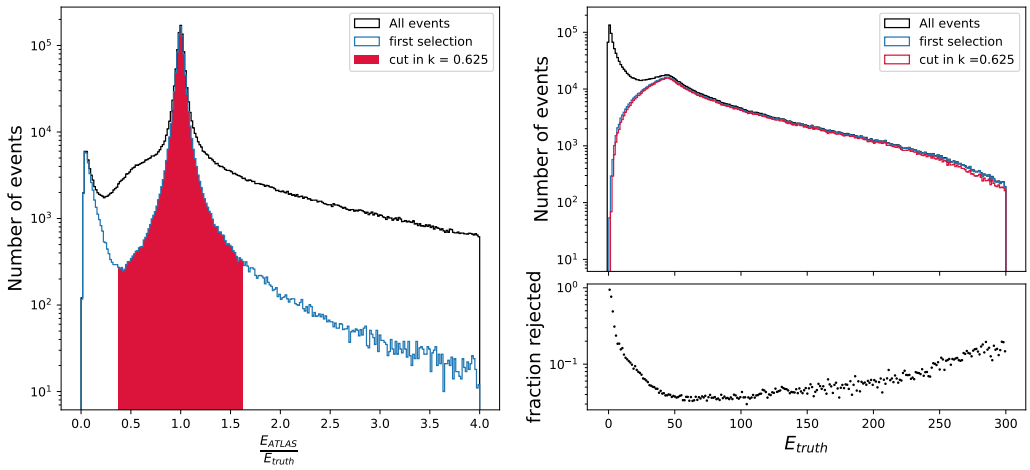


Figure 6.2: On the left we can see the distributions of the ATLAS BDT prediction result divided by the truth label energy, on the right the resulting energy distributions and the fractions of rejected in the different energy bins.

, this selection should not give an unfair advantage to the model as we are removing the most poorly reconstructed ATLAS BDT events,

which should greatly benefit the ATLAS BDT's results.

A decision was made by previous students to keep the selection symmetric, however one could also consider to keep the selection one-sided, as there are no good explanations as to why we should discard events where the ATLAS BDT guesses the energy to be too high.

It is also required that the candidates pass the Likelihood Loose. Another selection criteria which is unique to this thesis when compared to earlier projects is that the p_track container is not empty, as we use those features where previously the p_ container had been used.

6.3 *Features*

There are a lot of different considerations to be made when deciding on the features which are to be included in the model, and a large part of it depends on the structure of the model. However, there has not been any time to start a feature importance analysis. Therefore, the selection of features builds on in office discussions as well as drawing inspiration from previous theses.

In a frustration with the lack of readily available information about the features, the table of features seen in table (6.1, 6.2 & 6.3) has been created. It does not only include a short description the name of the features as they might appear in papers but also the name with which they are generated through the Ntuple production such that future students might have an easier time finding the correct features.

6.3.1 (x, y, z) *vs* η, ϕ, R

During the data production it was discussed whether η, ϕ, R were not a better measure of relation/distance between the data points, these would also seem more familiar to researches at ATLAS as the (x, y, z) are hardly used. The η, ϕ, R where therefore tested meaning that the distance between the graphs were no longer in terms of euclidean distance but some arbitrary distance which better corresponded to the expected connection between the data points. Unfortunately a naive implementation did not lead to improved performance. The reasons for this could stem from the following issues which would have to be taken into consideration if η, ϕ, R where to be used.

- Differing scales

  unlike the (x, y, z) coordinates which are naturally all on the same scale, the η, ϕ, R are all on different scales. This might be fixed using a MinMax scaler see (section 6.3.3).

- The cyclical nature of ϕ

  Since phi is cyclical in nature one would have to take that into consideration when calculating the distance to the neighboring

datapoints. There currently was no option to do this using the pytorch geometric library. It would therefore require replacing the function for calculating the neighbors with a custom function.

The belief is still that a correct switch to η, ϕ, R might bring about increased performance. The option of weighting the distance calculation so that R for an example might be weighted less than η and ϕ even after getting them on the same scale, could also be an interesting feature.

A counterargument against seeking out such an implementation is that it only affects the initial graph and already after one EdgeConv the distance is some unknown distance in 3-dimensional space, why the effect might not be all that apparent.

6.3.2 *pseudo-truths for training in real data.*

While there was not enough time to generate, evaluate or train on real data the pipeline was constructed with the possibility of training in real data in mind, as that had shown promising results in earlier theses. A pseudo truth label can be generated by using Eq.7.4, picking one of the electrons in the electron pair, and determining the energy of the remaining electron through the ATLAS BDT energy, the invariant mass is then set to the mean Z-mass ($M = 91.19GeV$). The energy of the chosen electron can then be determined, and the energies can be used as pseudo truth labels. It is noted that the Z mass is defined by a Breit-wigner distribution with a width of $\sim 2.5GeV$ which means that this energy is just an approximation. The approximation is considered good enough when used in combination with a cut on the invariant mass of the electrons. This method will inevitably have a dependency on the ATLAS BDT model, which the severity of should be checked upon implementation.

Scalar features		
Name	code ID	Description
$\langle\mu\rangle$	averageInteractions-PerCrossing	Average Proton-proton interaction per crossing
$n_{vertexReco}$	NvtxReco	Number of reconstructed vertices
n_{tracks}	p_nTracks	Number of reconstructed tracks
R_{HAD}	p_Rhad	Ratio of energy in hadronic calorimeter
R_{HAD1}	p_Rhad1	Ratio of energy in first layer of hadronic calorimeter.
f_{TRT}	p_TRTTrackOccupancy	Fraction of detector parts (straws) with hits in TRT.
E_{top40}	p_topoetcone40	Energy (in EM calorimeter) in cone of $\Delta R < 0.4$ around cluster barycenter excluding electron candidate (topological) energy.
E_{TG3}	p_fTG3	fraction of energy in crack region (tile-gap) scintillator detectors.
$\phi ModCalo$	p_phiModCalo	relative ϕ w.r.t. the cell edge of Layer 2 in ECAL (assumes constant cell size.)
$\eta ModCalo$	p_etaModCalo	relative η w.r.t. the cell edge of Layer 2 in ECAL (assumes constant cell size.)
E_{f0}	p_foCluster	fraction of energy in pre-sampler.
$\eta_{Cluster}$	p_etaCluster	η of electron candidate (topological) cluster
R12	p_R12	ratio between energy deposited in EM calorimeter layer 1 and 2.
$\Delta\phi_{TH3}$	p_dPhiTH3	Relative position of ϕ in a cell. $mod(2\pi + \phi, \phi/32) - \pi/32$
η_{Index}	p_cellIndexCluster	η cell index of cluster in layer 2
E_{acc}	p_eAccCluster	Total energy deposited in layer 1-3 of the ECAL
η	p_eta	η as determined by the tracking
$\Delta\phi_2$	p_deltaPhiRescaled2	Difference in ϕ between extrapolated track and the barycenter of the cluster $\phi_{cluster}$
$poscs_2$	p_poscs2	Difference in eta between the most energetic cell $\eta_{mostEcell}$ and $\eta_{cluster}$ the barycenter of the cluster $2 \cdot \eta_{cluster} - \eta_{maxEcell}/0.025 - 1$
p_T^{track}	p_pt_track	p_T^{track} estimated from the tracking algorithm †

Table 6.1: An overview of all the different scalar input features used in the final model.

Cell features		
Name	code ID	Description
x_{cell}	p_cell_x	x location of the calorimeter cell.
y_{cell}	p_cell_y	y location of the calorimeter cell.
z_{cell}	p_cell_z	z location of the calorimeter cell.
E_{cell}	p_cell_energy	Total energy measured in the calorimeter cell.
ΔR_{cell}	p_cell_dR	Distance from the calorimeter cell to the barycenter of the topological cluster.
$Area_{cell}$	p_cell_area	surface area of the calorimeter cell.
t_{cell}	p_cell_time	time of calorimeter cell activation relative to bunch crossing.

Table 6.2: An overview of all the different cell input features used in the final model.

Track features		
Name	code ID	Description
x_{track}	(p_)tracks_x	x location of last track measurement in the ID.
y_{track}	(p_)tracks_y	y location of last track measurement in the ID.
z_{track}	(p_)tracks_z	z location of last track measurement in the ID.
p^T_{track}	(p_)tracks_pt	The derived momentum of the track
ΔR_{track}	(p_)tracks_dR	Vicinity of track particle to the candidate particle. $\Delta R = \sqrt{(\phi_0 - \phi)^2 + (\eta_0 - \eta)^2}$
$\frac{d0}{\sigma_{d0}}_{track}$	(p_)tracks_doSig	The track $\frac{d0}{\sigma_{d0}}$, where do is the signed transverse distanve between the point of closest approach and the z-axis where σ_{d0} is the accompanying uncertainty
n_{SCT}	(p_)tracks_scthits	Number of SCT hits for the track
z_0	(p_)tracks_zo	longitudinal distance between point of closest approach and the z-axis.
θ_{track}	(p_)tracks_theta	θ of the track at reconstructed vertex.
η_{track}	(p_)tracks_eta	η of the track at reconstructed vertex.
ϕ_{track}	(p_)tracks_phi	ϕ of the track at reconstructed vertex.
$vertex_{track}$	(p_)tracks_vertex	The index of the track vertex. (ordered by transverse momentum)
n_{pixel}	(p_)tracks_pixhits	Number of pixel hits for the track.
$\frac{\Delta P}{P}_{track}$	(p_)tracks_dPOverP	Change in momentum over the track.
q_{track}	(p_)tracks_charge	Charge of the track.

Table 6.3: An overview of all the different track input features used in the final model. Note the two different types of *containers* for the different track types

6.3.3 *Data distributions*

The following distributions are made using the test data-set, which should be a sufficient amount to represent the full distributions, as we also use this assumption when checking the results.

These distributions were used as an analytic tool to not only see if the data representations are as expected, but also to see which transformation methods should be implemented for which features.

Below in figures 6.3,6.4,6.5,6.66.7,6.8 and 6.9. The log distribution of the features are shown before and after rescaling using Scikit learns [57] RobustScaler for all features except the energy type features E_{cell} , $p_{t,tracks}$ and p_{t,p_tracks}. The QuantileTranformer also from Scikit learn is used in this case as an attempt to combat the extremely long energy tails. The QuantileTranformer is non-linear and as such might ruin linear correlations between the features, which is why it is only used on a small subset of the features.

The figures symbolize one of the difficulties with working with ATLAS data, there are many features, and they might all need different treatments in terms of rescaling. Most of the distributions even after rescaling would leave some machine learning enthusiast hesitating to include them in their model as some of them span quite large ranges and still have persistent outliers. Another rescaling which one could consider implementing is the MinMax scaler which forces the features to lay within a certain range; [-1,1] and [0,1] are common and preferred ranges for neural networks. All figures have frequency on the y-axis and the feature relevant unit on the x-axis.[8]

[8] There is no easy way to apply the axis labels or even figure out what units the different features are entered with. During some *introduction days* I attended virtually with the ATLAS collaboration it was mentioned that the easiest way to figure out the contents of different features was to confer with the code...

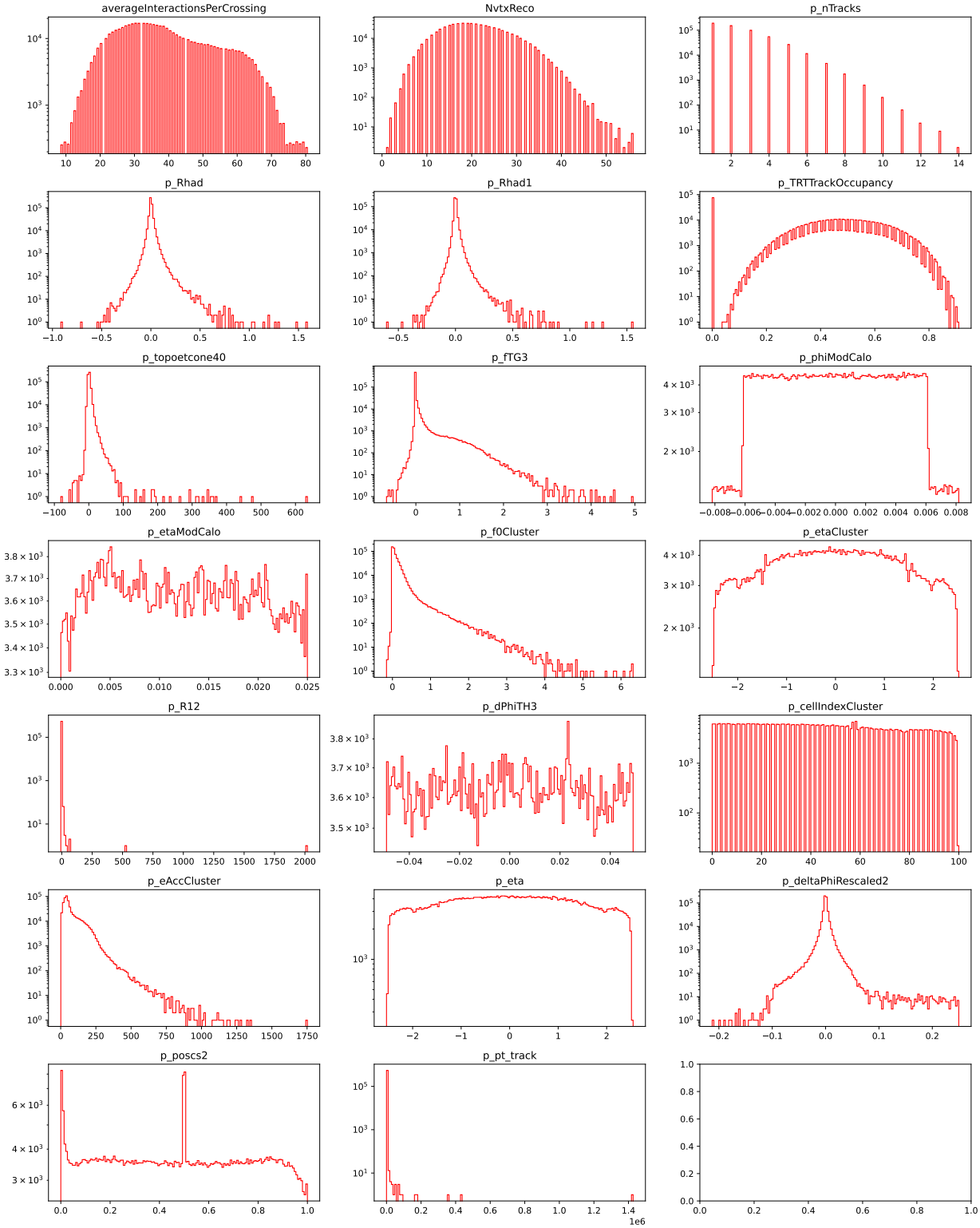


Figure 6.3: Histograms of all the different scalar variable distributions which are included in the final model, with the y-axis log scaled. See also 10.1

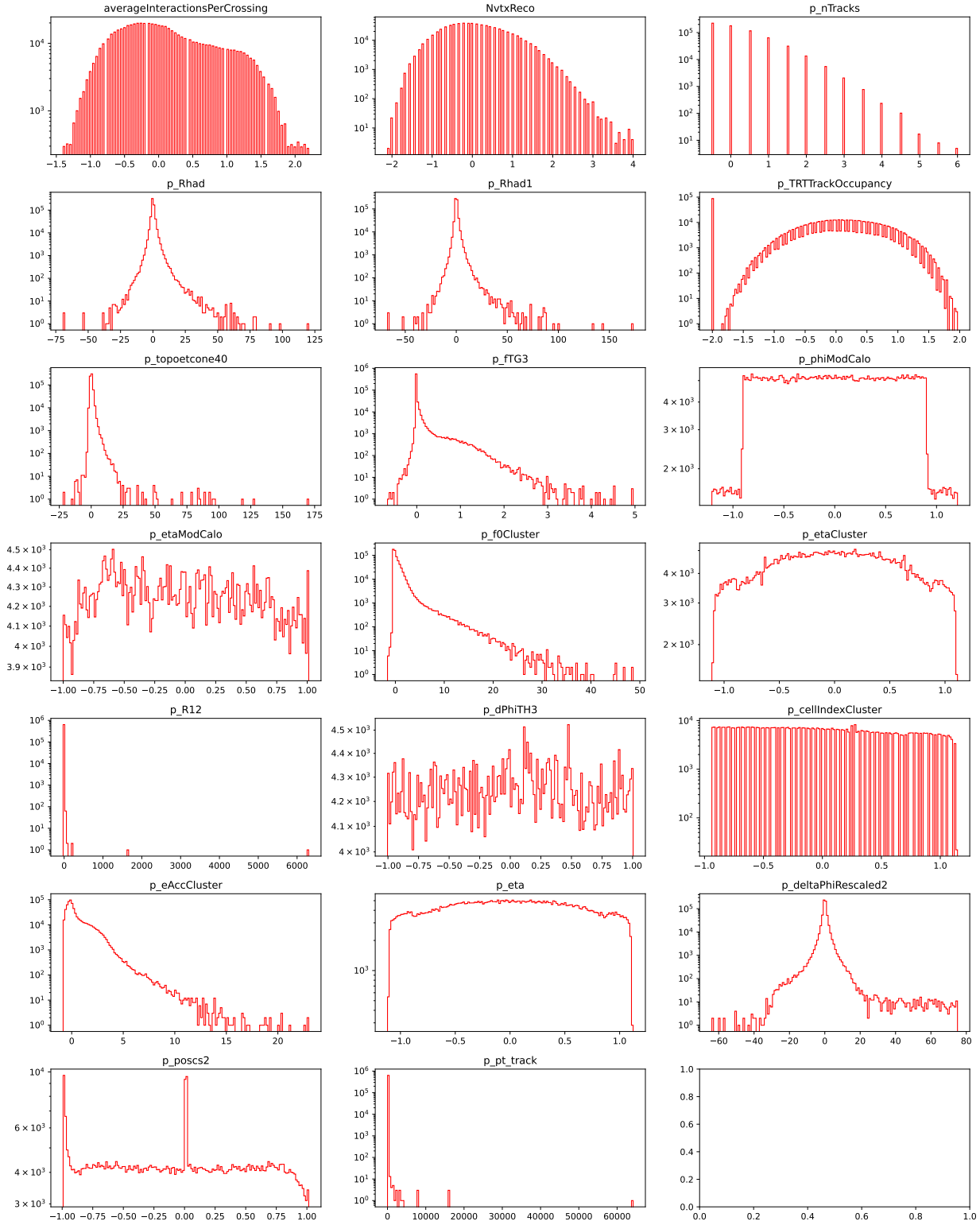


Figure 6.4: Histograms of all the different scalar variable distributions which are included in the final model, as they appear after transformations with the y-axis **log scaled**. See also 10.2

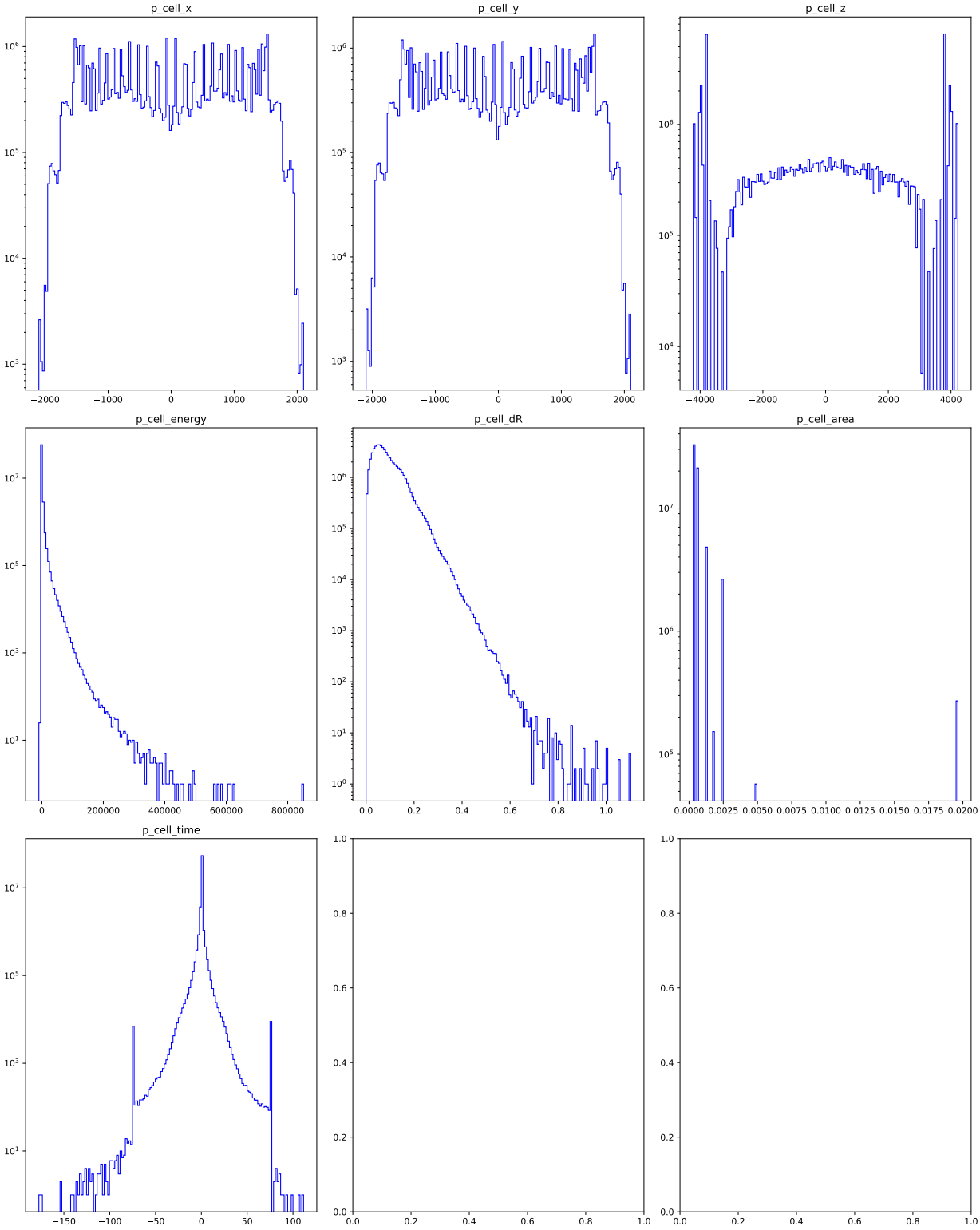


Figure 6.5: Histograms of all the different cell variable distributions which are included in the final model, with the y-axis **log scaled**. See also 10.3

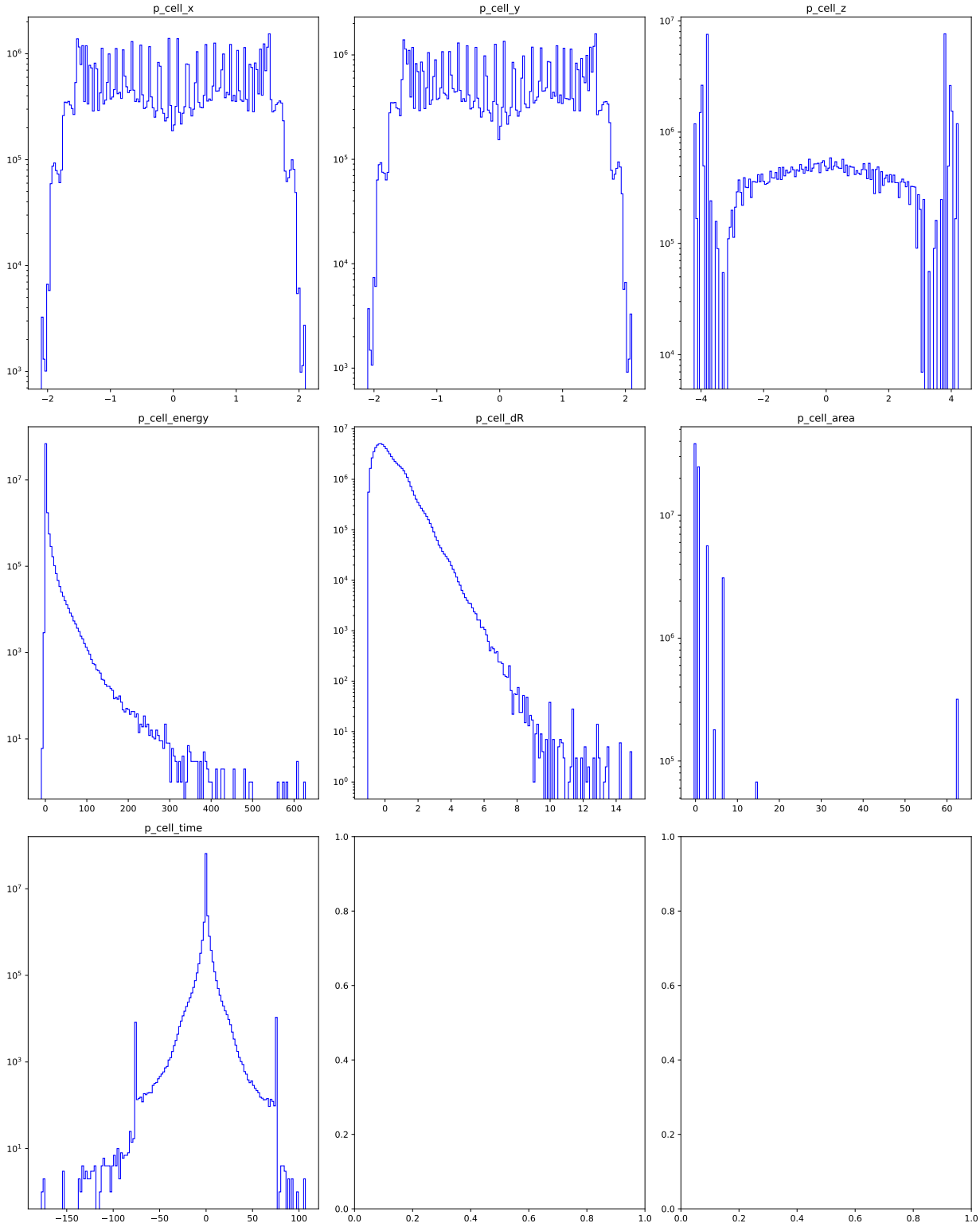


Figure 6.6: Histograms of all the different cell variable distributions which are included in the final model, as they appear after transformations with the y-axis **log scaled**. See also 10.4

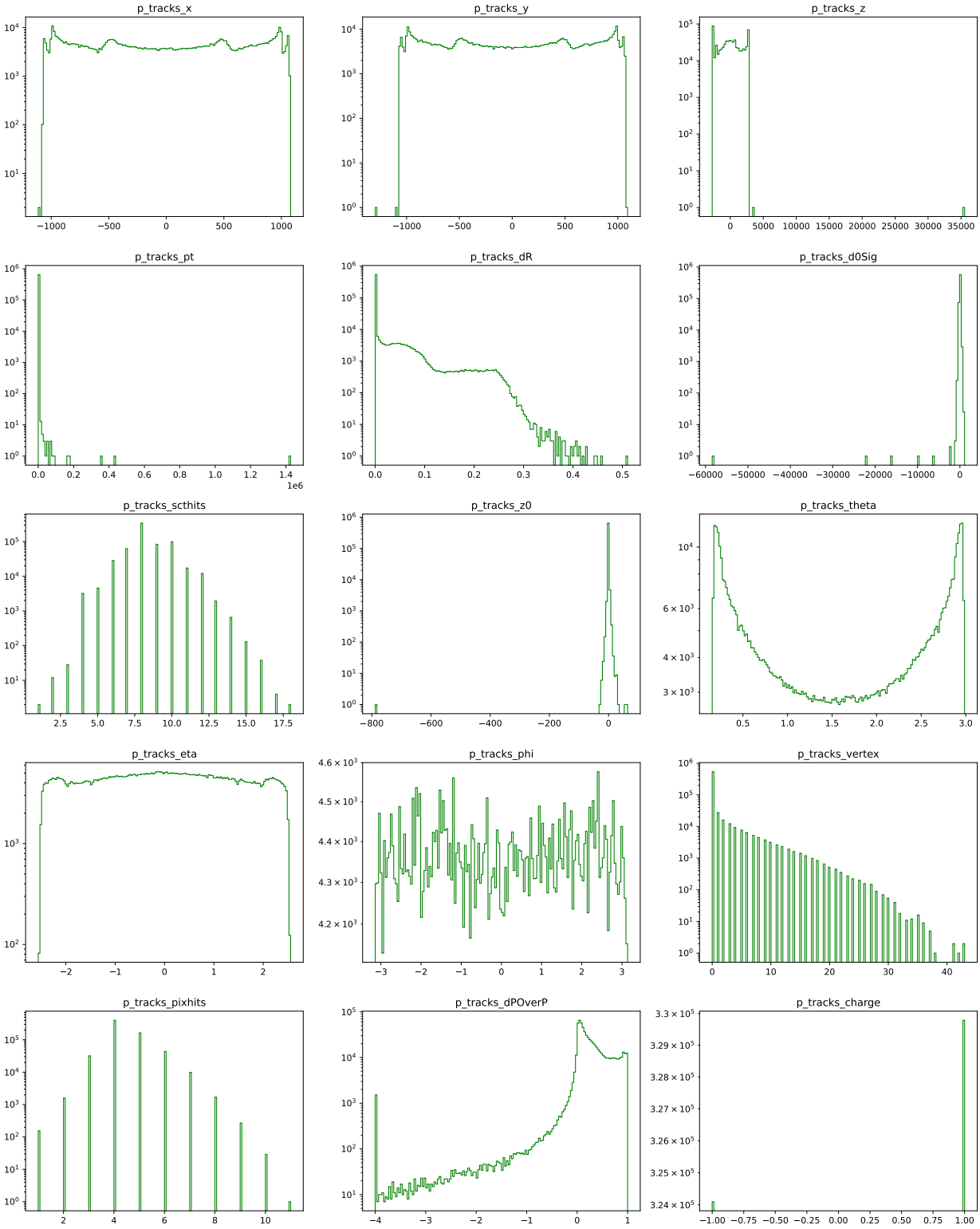


Figure 6.7: Histograms of all the different probe track variable distributions which are included in the final model, with the y-axis **log scaled**. See also 10.5

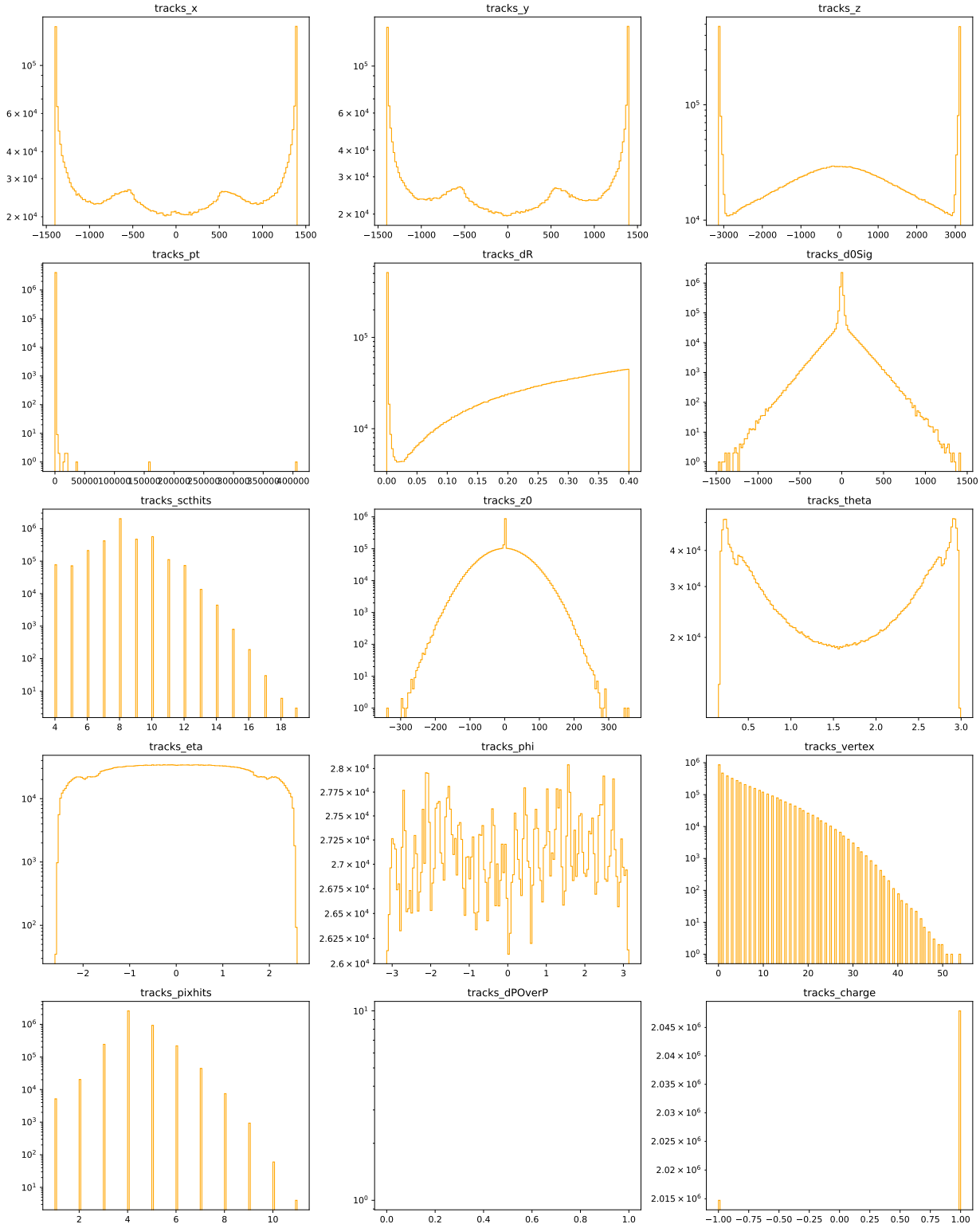


Figure 6.8: Histograms of all the different track variable distributions which are included in the final model, with the y-axis **log scaled**. See also 10.6

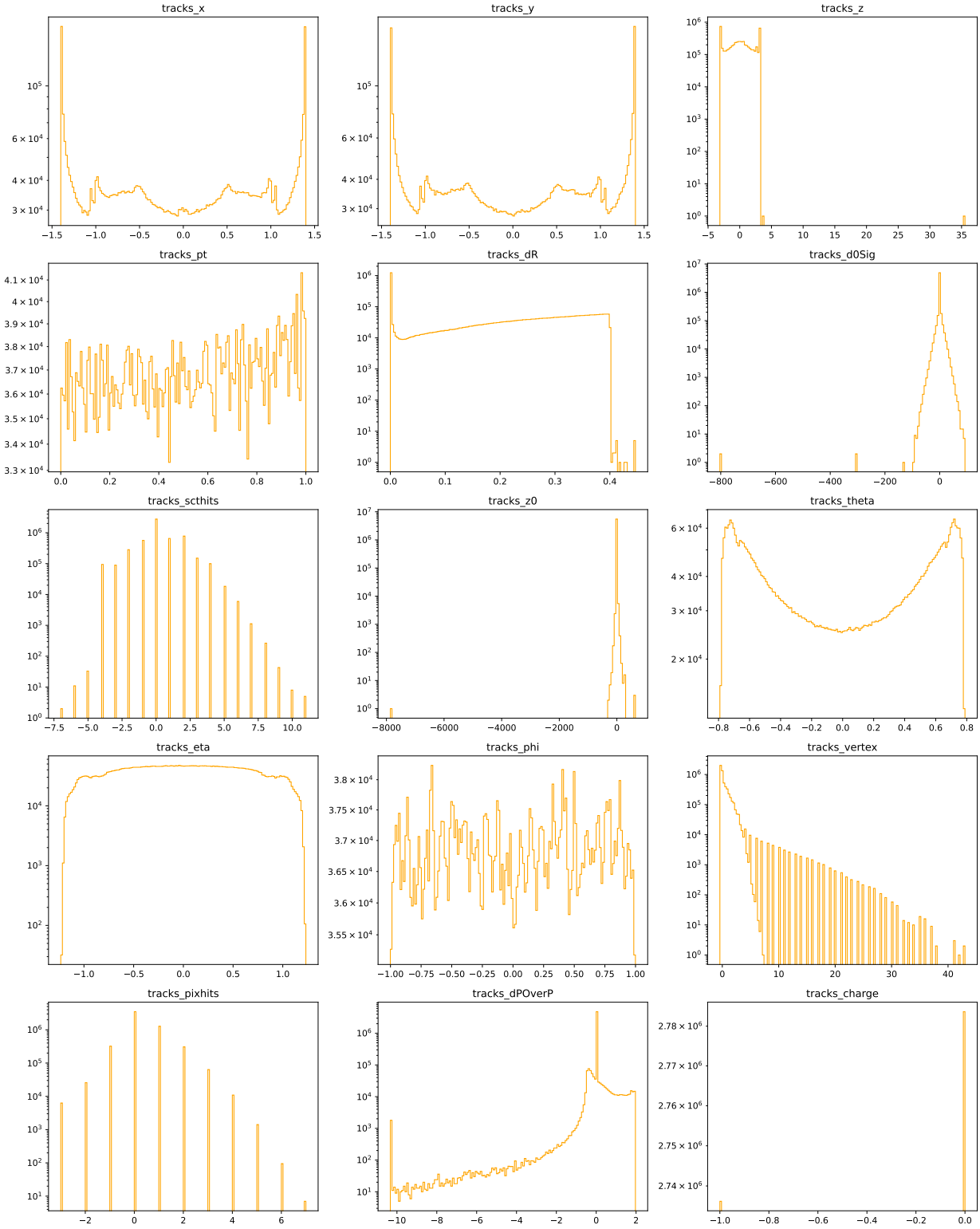


Figure 6.9: Histograms of the combined track distributions for both tracks and probe tracks as they appear in the graphs, that is after transformations, with the y-axis **log scaled**. See also 10.7

7 Model Architecture

Setting out to construct a GNN, which is considered a relatively new deep learning method, that can reconstruct energy in an experiment as complicated as the ATLAS experiment is quite an arduous task. There are only few papers that combine the ATLAS experiment and GNN's the ones I were able to find are [43], [25] and and [60]. The latter being the paper which most closely relates to the methods implemented in this thesis. In [60] they implement a DGCNN called particle net, which like the method in this thesis treats each particle as a node. The network structure is almost identical to the one found in this thesis, though it is used for classification tasks and the input features are different. The paper was found rather late in the process, it is therefore remarkable how similar the two network structures turned out regardless. The only method directly inspired from the ParticleNet paper is linearly increasing the node features in the generated graph layers as a function of the layer number.

One of the largest challenges with the development of this model is the mixed data input[1]. In this thesis two different models were constructed which take different approaches to the issue of mixed data types.

[1] Data from the calorimeter and the data derived from the tracks

The contents of the following section is mostly based on in office discussions and tinkering with the model, therefore there will, when compared to the rest of the thesis be noticeably fewer citations. While it would be preferrable to support every single decision made in the model development process with a link to an article stating the expected performance gain from a certain implementation or proving the correctness of the method. These unfortunately for the most part do not exist or do not often generalize well across different data-sets. Therefore, the search for the best or even a functioning model is based on trial and error and chasing results. If a result plot or even a simple number were to be included for the entire history of the model development this thesis would quickly grow to an unwieldy size, not to mention the extra time it would take to carefully log and create graphical representations whenever a change was implemented, as such I have taken the liberty to include descriptions of tested methods and state whether it led to and increase in performance or not without providing much further proof of testing. Under any circumstance it would make sense to go back and test some of the applied methods again now that a functional model has been developed.

7.1 Performance metrics

The purpose of the loss function is to provide a *smooth* estimate of the error between a label and a prediction, however it does not carry much meaning beyond that, therefore it is necessary to come up with a different method for evaluating the final model.

7.1.1 Performance metric in MC data

For MC simulated data this is done with the Relative Inter Quantile Range (ReIQR), which is inspired from [1]. First the Relative Error (RE) is calculated through.

$$RE = \frac{E_{pred}}{E_{truth}}. \tag{7.1}$$

This calculation is done with energy predictions E_{pred} from both the GNN model and E^{BDT}_{ATLAS}. This will produce Gaussian distributions around 1 for both models. The smaller the width the more precise the model. Hereafter, the effective InterQuartile Range (eIQR) is calculated through.

$$eIQR = \frac{P_{75}\left(RE\right) - P_{25}\left(RE\right)}{1.349}. \tag{7.2}$$

Once again this is done for both models. The factor 1.349 is a normalization term due to the gaussian nature of RE. For a gaussian distribution the 75th percentile corresponds to $\mu + 0.6745$ likewise the 25th percentile corresponds to $\mu - 0.6745$ Therefore the IQR represents 1.349σ. The normalization term cancels out in the final evaluation metric. The choice of 25th and 75th are completely arbitrary,[2] and one could consider any pair of percentiles. The choice of 25th and 75th is based on previous theses.

[2] although they should fullfill $lower = upper - 1$

We are now capable of comparing the relative performance of the models through

$$\begin{aligned} \text{relative metric} &= 1 - \frac{candidatemodel}{benchmarkmodel} \\ reIQR &= 1 - \frac{eIQR_{candidate}}{eIQR_{benchmark}}. \end{aligned} \tag{7.3}$$

In the following section concerning model development a comparison between models will always mean a comparison of the $reIQR$ as all of the model development has been done using MC simulated data.

7.1.2 Performance metric in real data

In real data we are lacking a truth label, which is required for both the training process and also the calculation of the $eIQR$. Therefore another metric is needed when comparing performance in real data. As mentioned in section 2.2 the distribution of the Z-boson mass follows a Breit-wigner distribution. It is therefore possible to generate an invariant mass estimate of the particle candidate pairs through,

$$
\begin{aligned}
M_{inv}^2 &= 2 \cdot P_{T1} P_{T2} \left(cosh(\eta_1 - \eta_2) - cosh(\phi_1 - \phi_2) \right) \\
k &= cosh\left(\eta_1 - \eta_2\right) - cos\left(\phi_1 - \phi_2\right) \\
M_{Inv} &= \sqrt{2 \cdot \frac{E_1 \cdot E_2}{cosh(\eta_1) \cdot cosh(\eta_2)} \cdot k},
\end{aligned}
\tag{7.4}
$$

using the energy estimates generated by the models we can generate different invariant mass distributions. With a perfect noise-free detector and a perfect model we would be able to fit the resulting distribution with the Breit-Wigner mentioned earlier. However due to noise and imperfections in both detector systems and models the Breit-Wigner is convoluted[3] with a *Crystal-Ball* (CB) function developed by the Crystal Ball collaboration [54], Described by

[3] a convolution is the integral of the product over the two functions, with one reversed and shifted.

$$
\begin{aligned}
f\left(x; \alpha, n, \bar{x}, \sigma\right) &= N \cdot \begin{cases} exp\left(-\frac{(x-\bar{x})^2}{2\sigma^2}\right), & \text{for } \frac{x-\bar{x}}{\sigma} > -\alpha \\ A \cdot \left(B - \frac{x-\bar{x}}{\sigma}\right)^{-n}, & \text{for } \frac{x-\bar{x}}{\sigma} \leq -\alpha \end{cases} \\
&\text{where} \\
A &= \left(\frac{n}{|\alpha|}\right)^n \cdot exp\left(-\frac{|\alpha}{2}|^2\right) \\
B &= \frac{1}{\sigma\left(C + D\right)} \\
C &= \frac{n}{|\alpha|} \cdot \frac{1}{n-1} \cdot exp\left(-\frac{|\alpha|^2}{2}\right) \\
D &= \sqrt{\frac{\pi}{2}\left(1 + erf\left(\frac{|\alpha|}{\sqrt{2}}\right)\right)}
\end{aligned}
\tag{7.5}
$$

[4]

[4] Equation written as seen at [21]

It is now possible to create a fit where the parameters of the Breit-Wigner are fixed to the Z-boson mass values while the CB function parameters are fitted. The resulting width σ_{CB} of the CB function will then, in the same spirit as the reIQR, be a measure of the performance of the model, where a smaller width is the result of a better fit. Sanity checks can be made by looking at the MC truth values, the fit of the MC truth values are shown in Figure 7.1.

The fit shows a low σ_{CB}, which indicate that the Breit-Wigner mostly explains the full fit. It is important to mention here that this fit seems to have a larger value of σ_{CB} compared to dataset used by previous students. The cause of this is not known but could be the result of poorer selection criteria, however this has not been investigated further. The larger σ_{CB} should not have any effect on the final measure due to errors being added in quadrature and with final σ_{CB} values of ~ 2.5 the contribution from 0.073 becomes neglible.

7.2 *Model development*

A model inherited from Rasums Ørsøe's thesis project[78]. Was used as a basis for the development as such the choice of which python

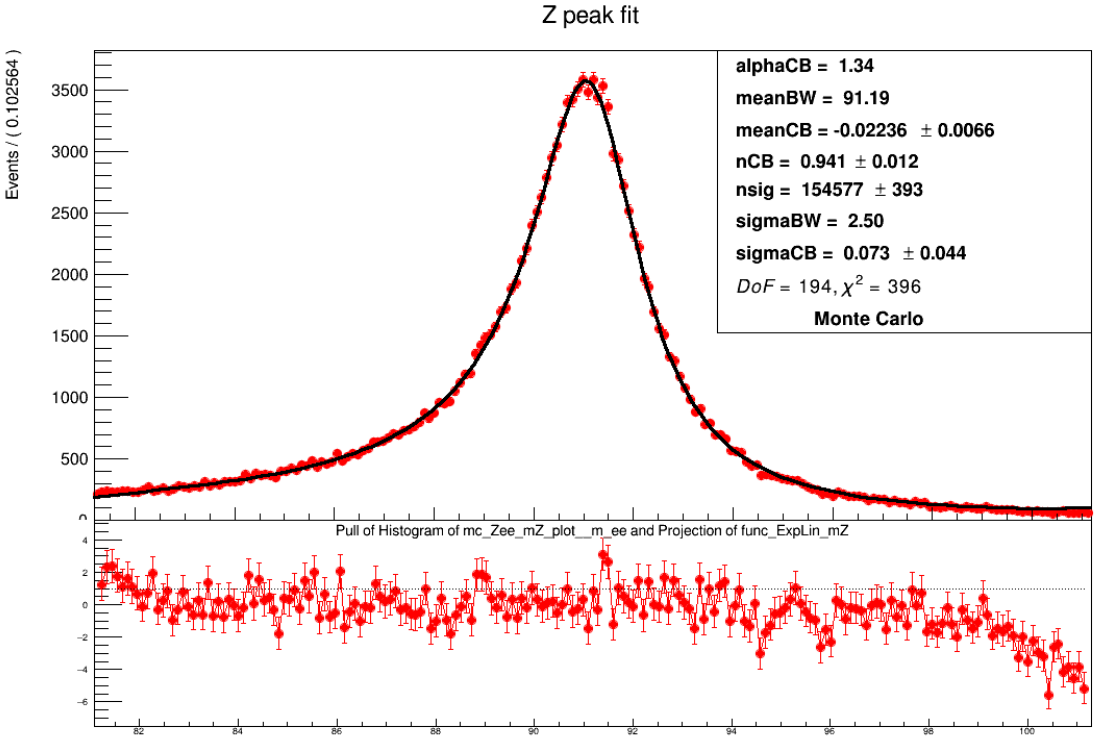


Figure 7.1: Bret-Wigner convolved with a crystal-ball function fitted to the truth MC values.

ML package library to use for the NN had already been made. The two big contenders that are usually considered for large NN projects are the PyTorch [56] or Keras [16]. The decision landed on PyTorch and the PyTorchGeometric library which is specific to graph neural networks. No other libraries have been tested in this thesis.

The first question to consider is whether we want to separate the track and calorimeter cell data into two separate graphs whose outputs are then concatenated towards the end of the model.

7.2.1 *Combined model*

Initially efforts were focused on constructing a model that would use one graph to treat both the calorimeter data and the derived track data. The choice of pursuing this model is from the *belief* that there exists information in the *Edges* between track data and the calorimeter data.

Representing ATLAS data as one graph

Do we want to make the different types of data as similar as possible[5], or do we want to ensure that the model understands the existence of two different data-types and not risk confusion that might occur from the previous method, the second goal can be accomplished by simply having the inputs skewed and zero-pad the features[6]. This is a question that arises from the combination of different

[5] That is if a feature represents the deposited energy in the calorimeter then try and find the counterpart in the track data, which would be the momentum

[6] This again quickly leads to a discussion of whether the values should be zero-padded or given an entirely different value entirely since after rescaling the input features they are all centered at o

data-types into the same graph and as such there is not much help to be found in previous studies.

Two different versions were tested one with zero-padding and one with similar input features. Since no model seemed more performant than the other, the one with similar features was chosen as not having the zero-padding would at least mean smaller sizes of the graph files. [7]

[7] It should be noted that this was quite early on in the project, where neither of the models had any noteworthy performance

Jumping straight to the conclusion of this model, After much testing the model seemed somewhat performant and competing with the ATLAS BDT model, at one time even outperforming the ATLAS model with about 10%. Unfortunately this result was obtained rather early in the process before applying reasonable cuts to the input data, which would ultimately be a disadvantage to the BDT, and the result was not reproducible after implementing the correct selection criteria.

At this point it became clear that a somewhat significant disadvantage with the combined model was the inseparability of the model. That is since the graphs and the cells are combined into one large graph it is hard to gain any insights into what contributed to the models predicting power and also extremely difficult to diagnose why the model was struggling and thereby which measures to take to improve the model. The model also seemed to be quite volatile often running into exploding or vanishing gradients. These issues led to a painful process of researching cutting edge methods for GNNs and blindly implementing and subsequently removing one after another in the hopes of better results.

7.2.2 *Dual-graph model*

This ultimately led to a shift to the dual-graph model since this would at least allow for inspection of each of the sub-models individually by including or excluding each of them when training.

The benefit of this model is that we can inspect the different sub-models individually and then with the assumption that a *stand-alone* model tuned for better energy regression on its own will have better contributions to a combined model. We also have the added benefit of being able to check whether the inclusion of cell and track data lead to an increased performance in the model.

The full final model can be seen in Figure 7.2. The figure highlight the different sub-modules by a colored dashed line.

7.2.3 *Tools for model development.*

People often describe NN models as *black box* models, as a means to express the difficulty in gaining any kind of insights into the inner workings of the model. There do exist some tools and methods, which are mentioned in [51] [14] [66]. However, none of these papers make any mention of graph neural networks and most of these methods are not focusing on model development but rather result interpretation.

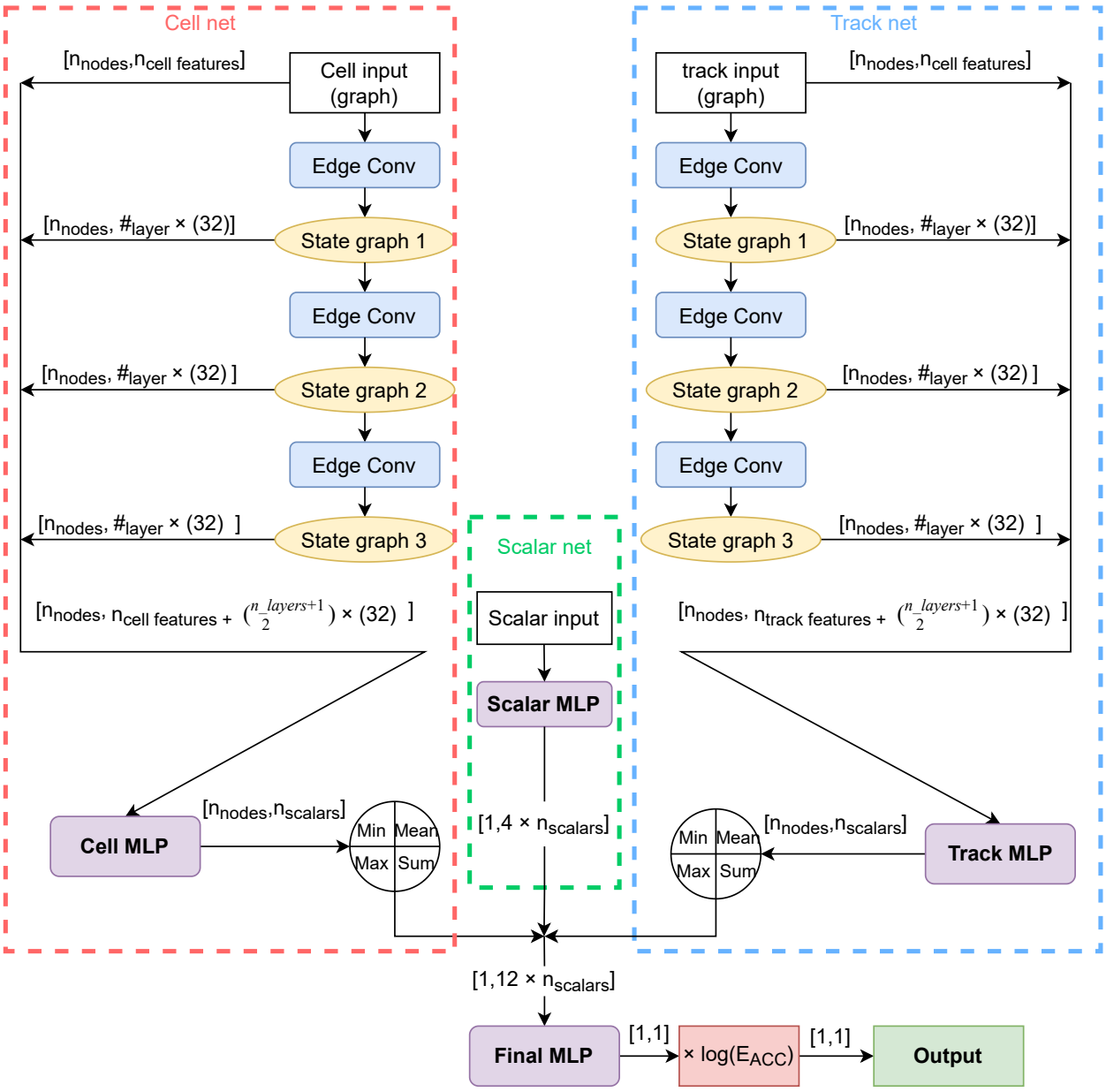


Figure 7.2: A schematic showing the different components of the full model. The different submodules can easily be added and removed from the total network. the output dimensions of the state graphs can be scaled by a full model wide factor, the final model is scaled up by 4.

A permutation of input feature method contributed to [30] allows for a feature importance estimate in deep neural networks however this too is not of much help doing the early stages of development but certainly are interesting ones you have a working model, as it allows you to cut away meaningless features reducing computation time and freeing up vital RAM. During the initial phases of model development one is generally left with two tools, the training/validation loss plot,[8] which is shown in figure 7.3 and the result of the tested model. Which can basically be summed up to trial and error.

[8] It is useful to plot the learning rate along with the loss, assuming the learning rate to be one of the most important hyperparameters.

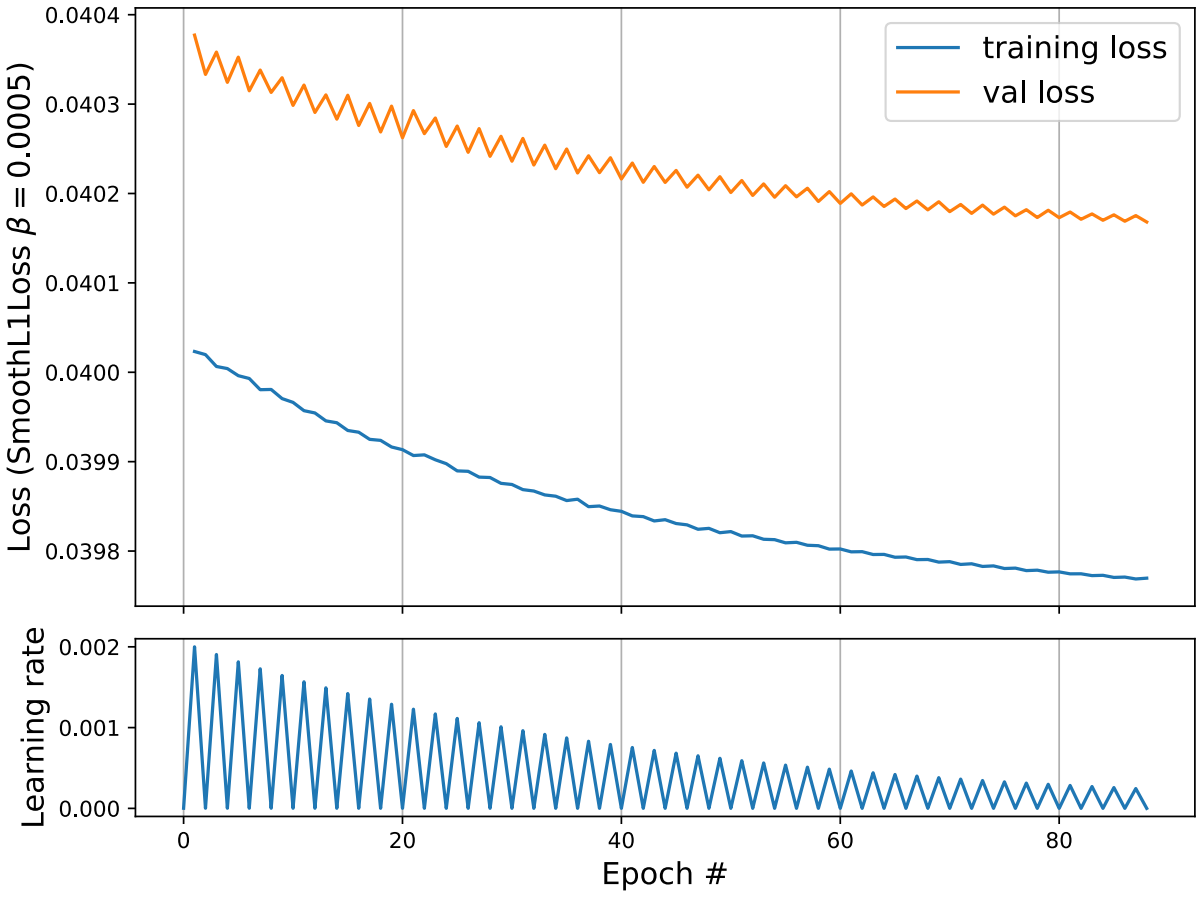


Figure 7.3: The training and validation loss of the final model, it should be noted that this model was initialized from a well predicting initial state, otherwise a larger error in the very early epochs would be expected

Once a functioning model has been obtained we enter into what is called the *hyperparameter optimization* or *hyperparameter search* phase, and salvation from the cumbersome trial and error task is at hand, since one can implement an automated *bayesian optimization scheme.* In this thesis optuna [5] was initially implemented. The optuna framework allows the developer to make use of a bayesian optimization scheme with added benefits such as early stopping and quick production of resulting graphs.

7.2.4 *Bayesian optimization*

The goal is finding a global optimization of a combination of hyperparameters [9] which results in the best performing model[10]. The Bayesian optimization model achieves this task by iteratively building a probabilistic model of the hyperparameter space based on an assumption of *smoothness* between results. The specific *sampling* algorithm employed by Optuna is the TPE Sampler [55]. A Bayesian optimization algorithm seeks to determine an acquisition function that balances exploitation, which can be described as where the algorithm thinks a good resulting model is likely to be and exploration, which are areas of the hyperparameter space which have yet to be explored figure 7.4, shows an illustrative model of the scenario.

[9] User tuneable parameters

[10] Based on the user defined objective function

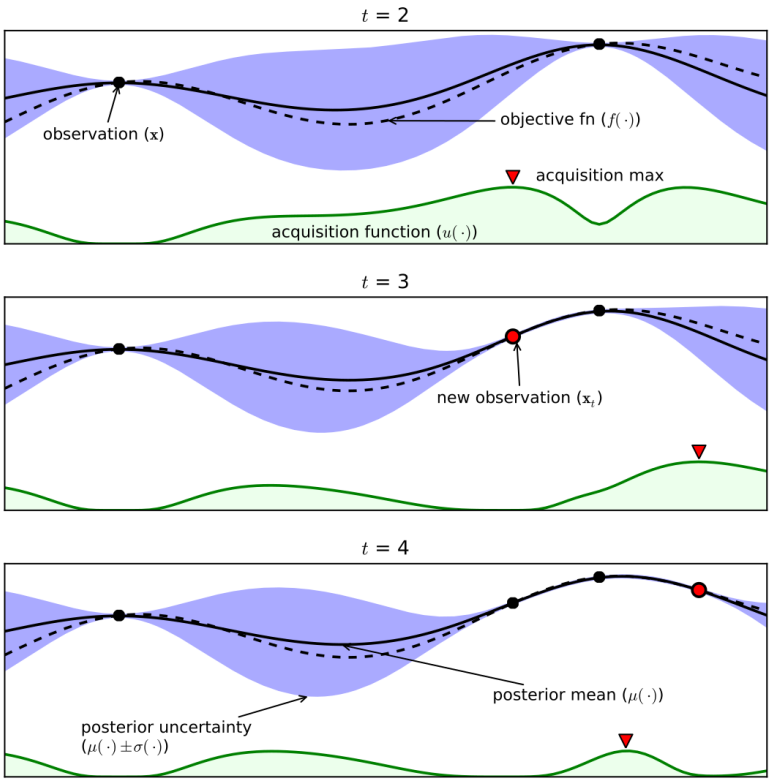
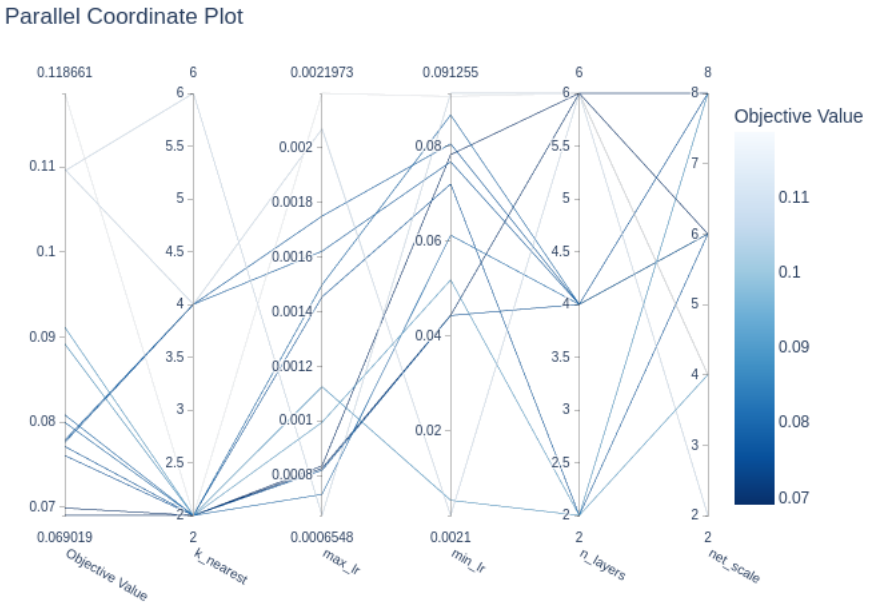


Figure 7.4: An illstrative figure showing the principles of a bayesian optimization algorithm, which visualizes the acquisition functions trade-off between exploration and exploitation. Figure from [10]

Unfortunately the implementation of the optuna optimization model was set up and ran earlier in the process while the model still used the unified graph format, and since implementing the new model structure there have not been enough time to implement and run optimization on the new model, or any of the sub-models, meaning the resulting model is using user tuned hyperparameters. An example of the graphs resulting from using optuna can be seen in figure 7.5

(a) Shows the intermediate objective values at different epochs called steps in optuna



(b) The figure shows the different hyperparameter combinations tested and the resulting objective values with a color gradient

Figure 7.5: Examples of using Optuna, which unfortunately has not been applied to the final model.

7.3 *Additional architecture*

The following section seeks to highlight some different methods, which have been tested during the development phase.

7.3.1 *Methods related to learning*

A lot of different solutions to learning rate schedulers, optimizers, methods for finding optimal upper or lower bounds on the learning

rate as well as different loss functions were tested.

optimizers

For optimizers the two main optimizers which were tested where the NAdam optimizer and the SGD with nesterov momentum. In the final model the SGD with nesterov momentum is implemented, both optimizers are available from the pytorch package [56].

learning rate schedulers

Several learning rate schedulers were tested amongst others; One-cycle cyclical with an exponential decay tail, cyclical with no decay, cyclical with linear decay of max learning rate, cyclical learning rate with exponential decay of max learning rate, and finally the one chosen a cyclical learning rate with exponential decay of both the max and min learning rate, along with a cyclical schedule for the Nesterov momentum which is non decaying but cycles opposite the learning rate cycle. The final scheduler is created for this model.

This schedule/optimizer combo was preferred as it seemed more stable than previously tested schedules had better results and was not dependent on max number of epochs like some other schedules with decay were. An example of the learning rate over a training run can be seen in figure 7.3.

learning rate finder

For this model it was not managed to produce a working learning rate finder. The idea of a learning rate finder is to, after a short training period, increase the learning rate exponentially while recording the training loss the see how the loss initially does not move much before then descending, hitting a plateau and then exploding, as seen in the toy model in figure 7.6 b. However, this was never observed when attempted on this model and the results would instead look like what is shown in figure 7.6 a. Testing afterwards also found that the learning rate limits found by ignoring the unexpected structure would not necessarily be effective. The method was therefore ultimately abandoned.

Loss function

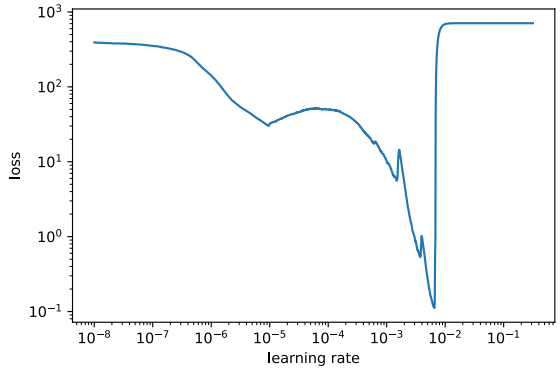
Several different loss functions were tested in this thesis, the logcosh loss

$$L(y, \hat{y}) = \sum |log(cosh(y^2 - \hat{y}^2))| \tag{7.6}$$

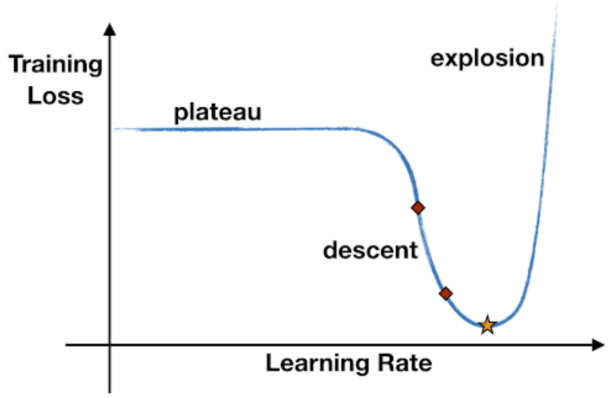
[52]

was considered a top contender as it was both used in the IceCube model developed by Rasmus Ørsøe[78] and in the CNN developed by Malte Algren[6].

Instead the final architecture uses the SmoothL1Loss[32] with mean as the optional reduction parameter



(a) An attempt at employing a learning rate finder, which would after a brief warmup at a low learning rate, record the training loss and increase the learning rate exponentiallyto determine lower and upper bounds for the learning rate.



(b) This figure shows the expected result after employing a learning rate finder, figure from [49] which also questions the utility of the method

Figure 7.6: Figures showing training loss as a function of learning rate

$$L(y, \hat{y}) = \frac{1}{N}\sum\{l_1, ..., l_N\} \tag{7.7}$$

$$l_n = \begin{cases} 0.5(\hat{y} - y)^2/\beta, & \text{if } |\hat{y} - y| < \beta \\ |\hat{y} - y| - 0.5 \cdot \beta, & \text{otherwise} \end{cases} \tag{7.8}$$

from the pytorch library [56] with a very low beta of $\beta = 0.0005$, it should be noted that with this β value the SmoothL1Loss is almost equivalent to the MAE loss, however during development it was found to be more stable. Examples of the different loss functions tested in this thesis can be seen in figure 7.7

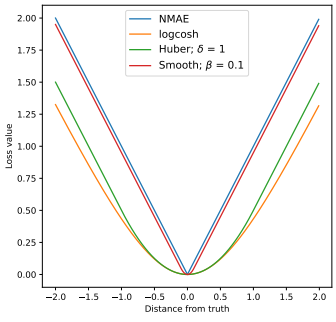


Figure 7.7: An illustrative figure of some different loss fucntions tested in this thesis.

7.3.2 *Graph normalization layers*

One could consider to include graph normalization layers as they are described in [13]. The graph normalization layer would be included twice in each of the MLPs in the EdgeConv layers. However upon testing it was found to be costly in terms of both computation time and RAM to the point where an implementation with the currently

available resources would only be barely feasible with the model architecture used in this thesis. Moreover, the implementation did not find any considerable performance increases immediately upon implementation and the method was therefore left out of the final model.

8 Results

The following section will contain all the results of the final model. Results of earlier model versions, have not been included since only rough plots, if at times any, were generated during the development phase.

8.1 Performance of GNN vs ATLAS BDT in $Z \rightarrow ee$

This section will cover all the performance plots of the full GNN model and how it compares the boosted decision tree model currently implemented in ATLAS

8.1.1 ReIQR.

The main result is the relative effective interquartile range measure described in section 7.1.1.

The uncertainty of the medians are calculated as

$$\sigma_{Med} = \sigma_{\bar{x}} \cdot \sqrt{2/\pi} \tag{8.1}$$

where $\sigma_{\bar{x}}$ is the uncertainty of the mean.

And the uncertainty of the ReIQR is then calculated through error propagation assuming the eIQR for the GNN and ATLAS BDT models to be independent.

Figure 8.1, shows the resulting peaks around 1 when plotting the predicted energies over the truth label energies. The figure shows no weird artifacts or skewing in neither the GNN model or the ATLAS BDT model.

The final results of ReIQR is

$$ReIQR = \mathbf{12.5} \pm \mathbf{3} \cdot \mathbf{10}^{-3} \tag{8.2}$$

In section 7.1.1 it was mentioned that the choice of 25th and 75th percentile were arbitrary choices and in order to show that the selection is not a nitpicking of the best result a plot of the ReIQR as result of the chosen percentile has been plotted in figure 8.2

Looking at Figure 8.2 We see that the GNN performances is better across all ranges and that the mean of the full range is better than the value posted at the 25th/75th percentile. Thus showing that there has been no accidental *cherry picking*. It should be noted that when getting extremely close to the 100th/0th percentile the GNN_{eIQR} converges at 7.4 while the $ATLAS_{eIQR}$ converges at 1.25, however at

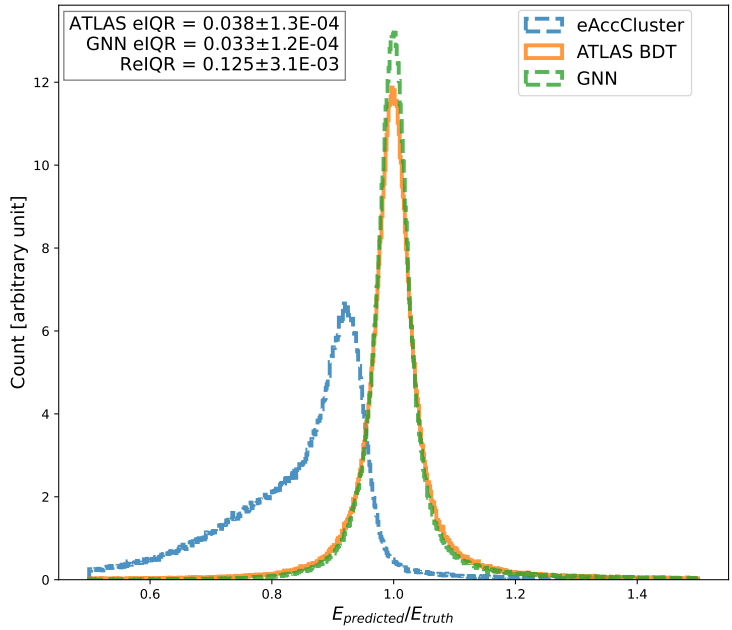


Figure 8.1: Histograms of the GNN and ATLAS BDT predictions over the MC truth value, as well as the cluster energy in the Accordion

that "percentile" we cannot really be considered on the peak yet. A more reasonable range from 5-45 and 95-55 for the lower and upper percentiles have been plotted. Excluding the low statistics area at the very peak as well as the outlier dominated area at the base.

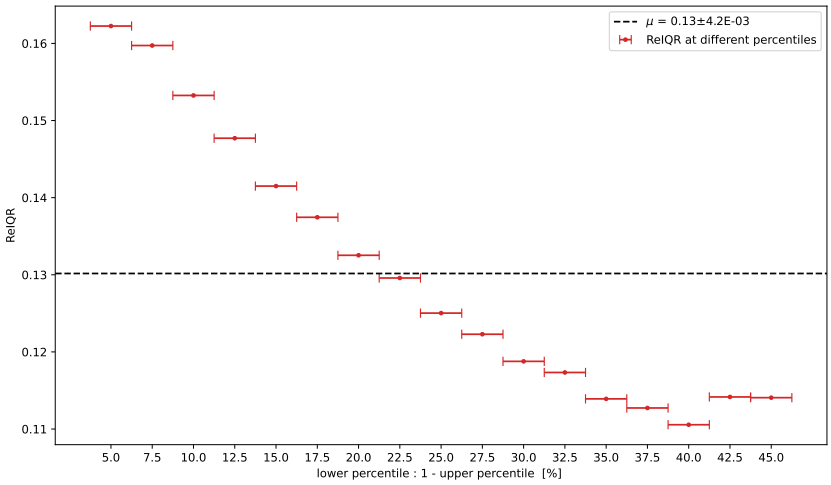


Figure 8.2: Plot showing the full range of RelQR using different percentiles.

8.1.2 *Further investigative plots*

The plots in the following section tries to investigate whether there are specific regions of the data set where the GNN model performs better or worse. Comparing it to the ATLAS BDT model in order to see if there are regions where one model has a relative advantage over the other. The plot shown in figure 8.3 is a common ATLAS

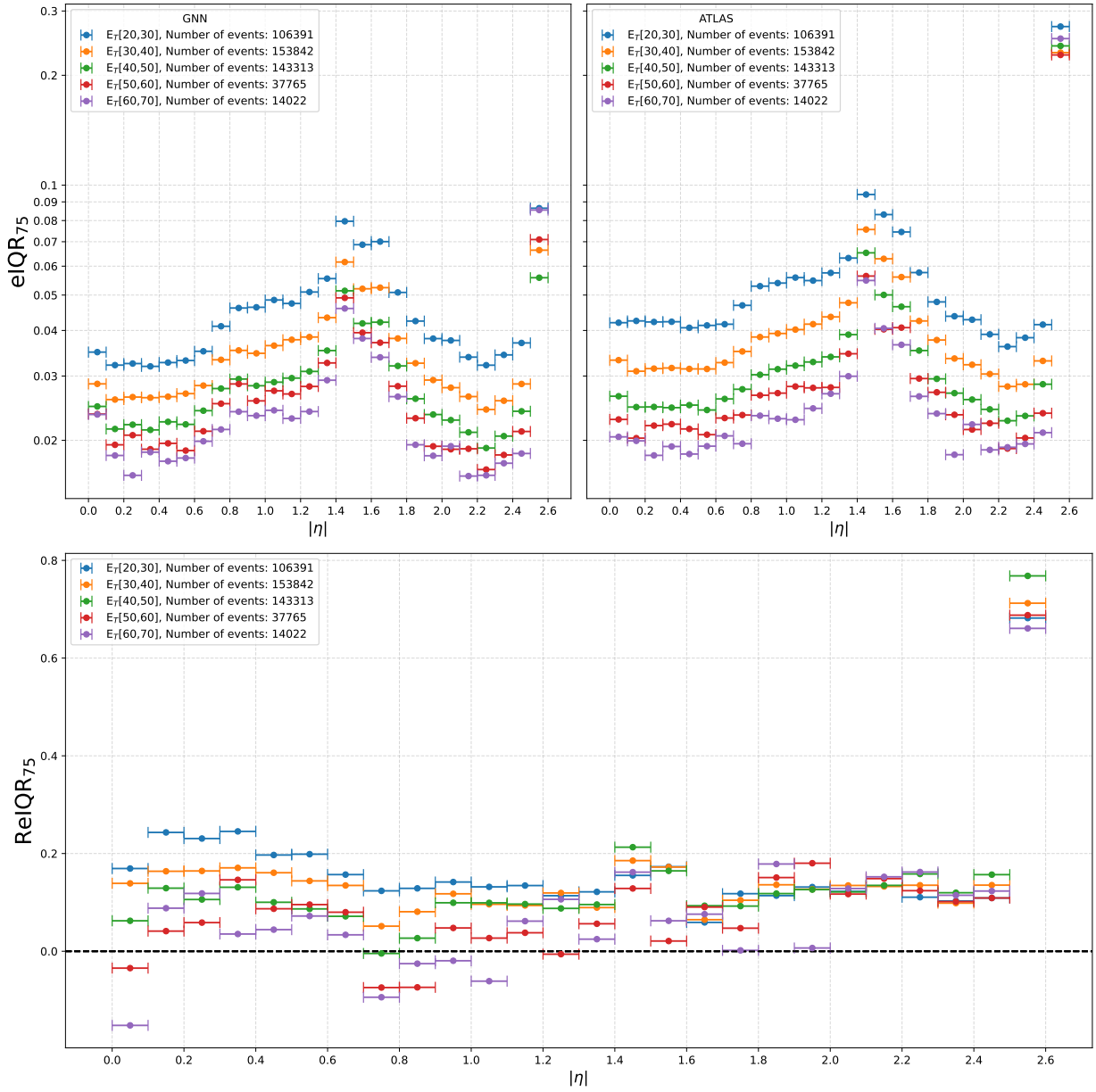


Figure 8.3: $|\eta|$ and energy (color) binning of the eIQR and the ReIQR.

performance plot, where not only the energy ranges but also the $|\eta|$ ranges are evaluated. The plot shows eIQR for both the GNN model and the ATLAS model as well as the ReIQR binned in $|\eta|$ for different energy ranges. The first thing to note is the performance drop between $|\eta| \sim [1.35 - 1.75]$ this is as expected and is due to the lower resolution in the crack region as mentioned in section 3.3.1.

The main thing to note about $|\eta|$ are that the eIQR are very similar for both models, which lead to constant increase in performance over the $|\eta|$ range. Another interesting result is that relative performance increase is highest for low energy events and lowest for high energy events. In the high energy there are some $|\eta|$ bins, where the GNN performs worse than the ATLAS BDT counterparts, however this is also the area, with the lowest statistics. The ReIQR seems to be slightly higher with the low energy electrons. It should be noted that the BDT uses a cut-off at $|\eta| < 2.5$ and therefore the large ReIQR above the cut-off is not a fair evaluation point.
The plot shown in figure 8.4 does not rely on the ReIQR as a performance measure, but instead a 2d heatmap or histogram of the distance from the predictions to the truth label as a function of the truth label energies. This plot is used to gain insights into the shape of the prediction errors. The figure shows no obvious bias towards over or underestimating at any energy regions. The rigid cone shape in the ATLAS BDT are a result of the cut in $|1 - \frac{E^{BDT}_{ATLAS}}{E_{truth}}|$ and therefore expected.

Figure 8.5 shows the eIQR and the ReIQR as a function of average interactions per crossing which is strongly correlated with pileup. The figure clearly shows an increase from $\sim$ 9% to $\sim$ 16% ReIQR over the pileup range. This is expected since the GNN includes track information which helps resolution in high pileup. As mentioned earlier pileup increases with an increase in luminosity and it is therefore very beneficial for any new implementation to perform better in these high pileup areas in preparation for run 3 and high luminosity run 4.

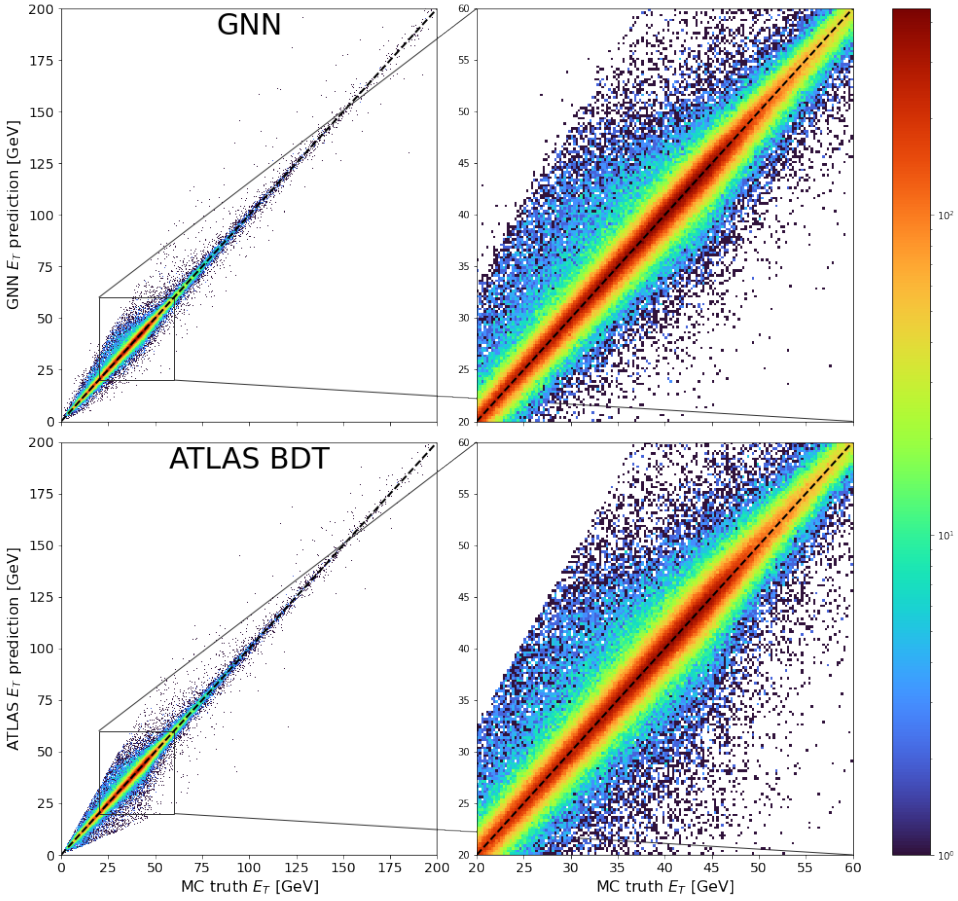


Figure 8.4: A two dimensional histogram (heatmap), of the GNN or ATLAS BDT predicted energy against the truth label energy.

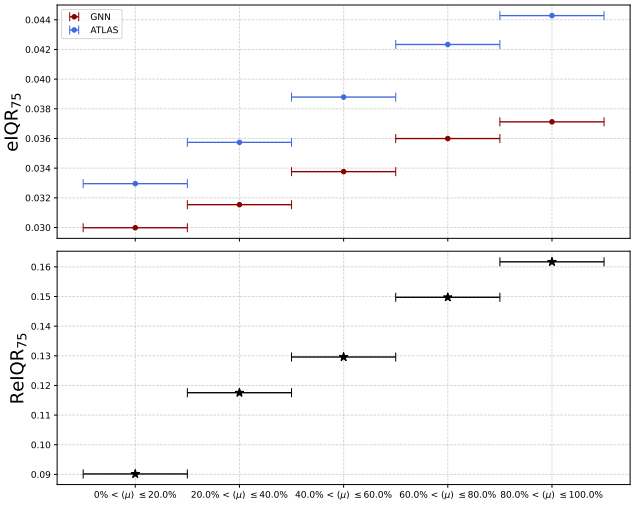


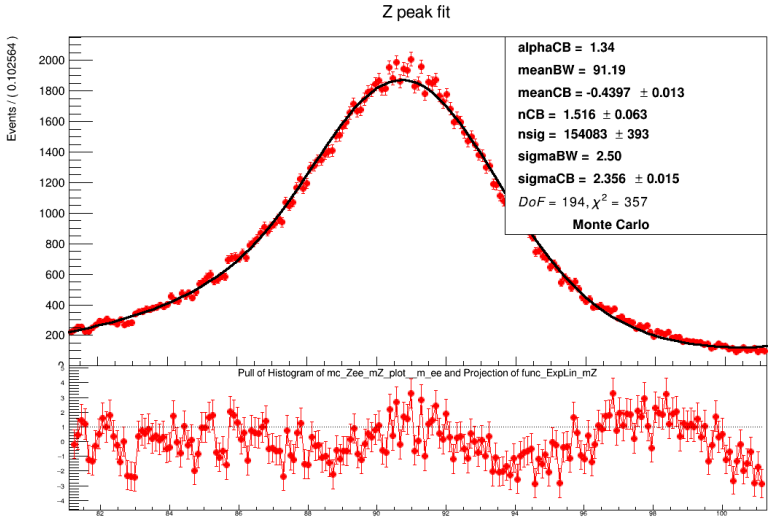
Figure 8.5: a binned plot of the eIQR for the GNN and ATLAS BDT models along with the ReIQR as a function of MinMax scaled $\langle\mu\rangle$ binned in 5 bins.

8.1.3 *Z-mass peak fitting.*

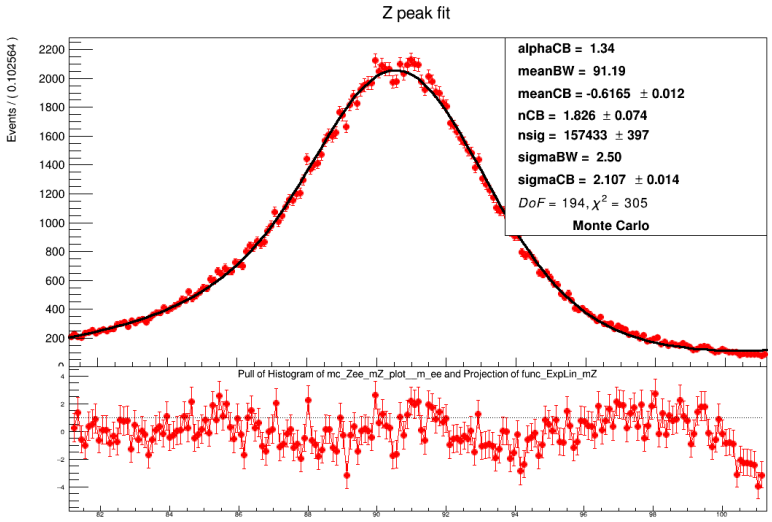
Figure 8.6 shows the z mass peak fits. As explained in 7.1.2 we can create another relative performance metric using the σ_{CB} in place of the eIQR. From the fit values we get

$$\langle 1 - \frac{\sigma_{CB}^{GNN}}{\sigma_{CB}^{ATLAS}} \rangle = 1 - \frac{2.107 \pm 0.014}{2.356 \pm 0.015} = 10.56 \pm 0.008 \tag{8.3}$$

The increase in performance seems consistent with the ReIQR metric although be it slightly lower. It should be mentioned that the other fitting parameter nCB also have small influence on the width and it is therefore required that they are similar in order for the method to be representative. In our case there is some difference in nCB between the two, but the performance measure is still considered valid.



(a) The figure shows the Breit-Wigner convoluted with a crystal-ball function fit of the ATLAS BDT model



(b) The figure shows the Breit-Wigner convoluted with a crystal-ball function fit of the full GNN model

Figure 8.6: The two figures shows $BW \otimes CB$ fits using the energy estimations from the two models. Only truth labeled electrons have been used in the fit.

8.2 Comparison to CNN

Finally it should be mentioned that the CNN implementation in [6] developed by previous students which have been mentioned throughout the thesis managed a

$$ReIQR_{CNN} = 22.4 \pm 0.7\% \tag{8.4}$$

at the 75th/25th percentile in $Z \rightarrow ee$ and is therefore outperforming the GNN model developed in this thesis. It should be noted that there are slight differences in the data selection steps between the two models and the feature distributions have not been checked against each other, the CNN model has also worked with more data than the GNN model.

8.3 Performance of the GNN sub-models

This section will contain a brief overview of the performance of the different sub-models of the GNN model by excluding parts of the model shown in Figure 7.2.

The results can be seen in table 8.1

	ReIQR
cell net	-0.03
track net	-0.79
scalar net	-0.03
cell + track net	Not tested
cell + scalar net	+0.04
track + scalar	-0.032
Full	+0.125

Table 8.1: small table that shows the performance of the different sub-models the figures and uncertainties of the results can be found in 10.2.

It can be seen that not a single one of the networks on their own give better performance than the ATLAS BDT, and gain in performance stems from inclusion of both cell, track and scalar data. This proves that the increase in performance does not stem from including additional scalar features but rather from the inclusion of the different data types. Unexpectedly the combination of track net with scalar net did not see a significant improvement when compared to using just the scalar-net.

9 Conclusion & Outlook

9.1 $Z \rightarrow ee$ performance and comparisons

The goal of this thesis was to construct a graph neural network implementation for energy regression using track and calorimetry data from the ATLAS experiment. Comparisons were made to the boosted decision tree model currently implemented at ATLAS, and a performance increase of

$$ReIQR = 12.5 \pm 3 \times 10^{-3}, \tag{9.1}$$

was found using the Relative effective InterQuartile Range. A performance increase of

$$\langle 1 - \frac{\sigma_{CB}^{GNN}}{\sigma_{CB}^{ATLAS}} \rangle = 10.56 \pm 0.008 \tag{9.2}$$

was found using the z-peak fit method.

The weaknesses and strengths of the GNN model across different energy and $|\eta|$ ranges was found to be similar to that of the ATLAS BDT model. Where the model seemed to have a slightly larger increase in performance for the low energy electrons.

The GNN developed in this thesis ended with a poorer performance gain compared to a CNN method developed by previous students. It should be mentioned that there are differences in the data used, and also the GNN has not gone through any kind of effective hyperparameter optimization process.

9.2 Analysis of the sub models

The analysis of the submodels shows that the combination of different data types (cells, tracks and scalars) is what drives the increased performance and not the inclusion of additional scalar features.

9.3 Final architecture

The dual graph architecture shown in figuer 7.2 was used in the final model, as it proved to be easier to develop. It is *not* argued that a functional single graph model cannot be developed, this thesis just did not successfully do so.

The following are a description of some of the key components of the model.

- EdgeConv operater

  The EdgeConv operater uses an MLP as the non-linear function and the average as the aggregation function.

  The edges are defined as the n nearest neighbors in euclidean distance.

- Loss function

  The SmoothL1Loss from the pytorch library [56] was used as the loss function.

- Optimizer
  - The model implements a SGD optimizer from the pytorch library [56] with Nesterov momentum.

- Schedulers

  The learning rate scheduler was developed for this model and is a cyclical learning rate with exponentially decaying upper and lower limits.

  The momentum scheduler for the Nesterov momentum uses a cyclical momentum with fixed limits shifted a half cycle compared to the learning rate scheduler.

- Activation functions

  The model uses LeakyReLU as the activation function throughout the network.

- Target

  The target is the logged MC truth label energy.

- Multiplication layer

  The final MLP output is multiplied by the logged cluster energy in the accordion before the final output.

9.4 Next steps

Owing to the promising results of the model the following further work exploring the performance of GNN for energy regression in calorimetry and track data is encouraged. However, it should be noted that the current leading model is the CNN, why it might be reasonable to try and start with improving the model.

9.4.1 Further, model development

The work done in this thesis provided a working model with relative improvement over the ATLAS BDT model, however it is suspected that further improvement might be found through the application of the following additions

- A hyperparameter optimization scheme like Optuna, see section7.2.4.
- Input feature analysis and subsequent removal of irrelevant features.
- Switching from x,y,z to η, ϕ, R, along with switch in calculation of distance for nearest neighbor calculation.
- Reimplementation of a combined model once a better idea of the model structure has been found.
- Usage of more data[1].

[1] This thesis did not use all the available data due to time constraints, and it has not been checked if the model still benefits from more.

9.4.2 *Additional implementation in ATLAS*

Implementation of the model in both MC and real data for $z \to ee$ $z \to \mu\mu\gamma$ and $H \to \gamma\gamma$ (only MC). This could be done for a more full comparison with other contending models.

As well as creating a universal model which can be trained for both $e\&\gamma$ regression in both MC and DATA and hopefully providing an overall improvement over models trained only on the specific data which it tries to predict, a so-called ensemble model.

As well as implementations for classification tasks rather than regression tasks[2]

[2] requires another activation function at the output layer, as well as another data selection.

9.4.3 *Detector output data*

In this thesis we used derived features for the track instead of more *raw* detector output data, the cell data is also picked after the clustering algorithm has picked the RoI. One could imagine applying more *raw* detector output level data. It should be noted that the amount of data increases drastically as we go to more raw output data. The choice of using the derived track data was deliberate. The tracking algorithm is quite precise and data output of the inner detector is massive. But using the cell data before the clusters have been gathered could be of some interest.

9.4.4 *Expanding to other calorimetry and tracking experiments.*

The model can handle both varying input sizes and also varying numbers of features through either splitting up the graphs into submodels or by zero-padding. The method should therefore be able to be quickly adopted to other experiments that use both tracking and calorimetry data with totally different detector structures without much changing of the base model.

10 Appendix

10.1 Distribution figures

The data distributions without the logged y-axis.

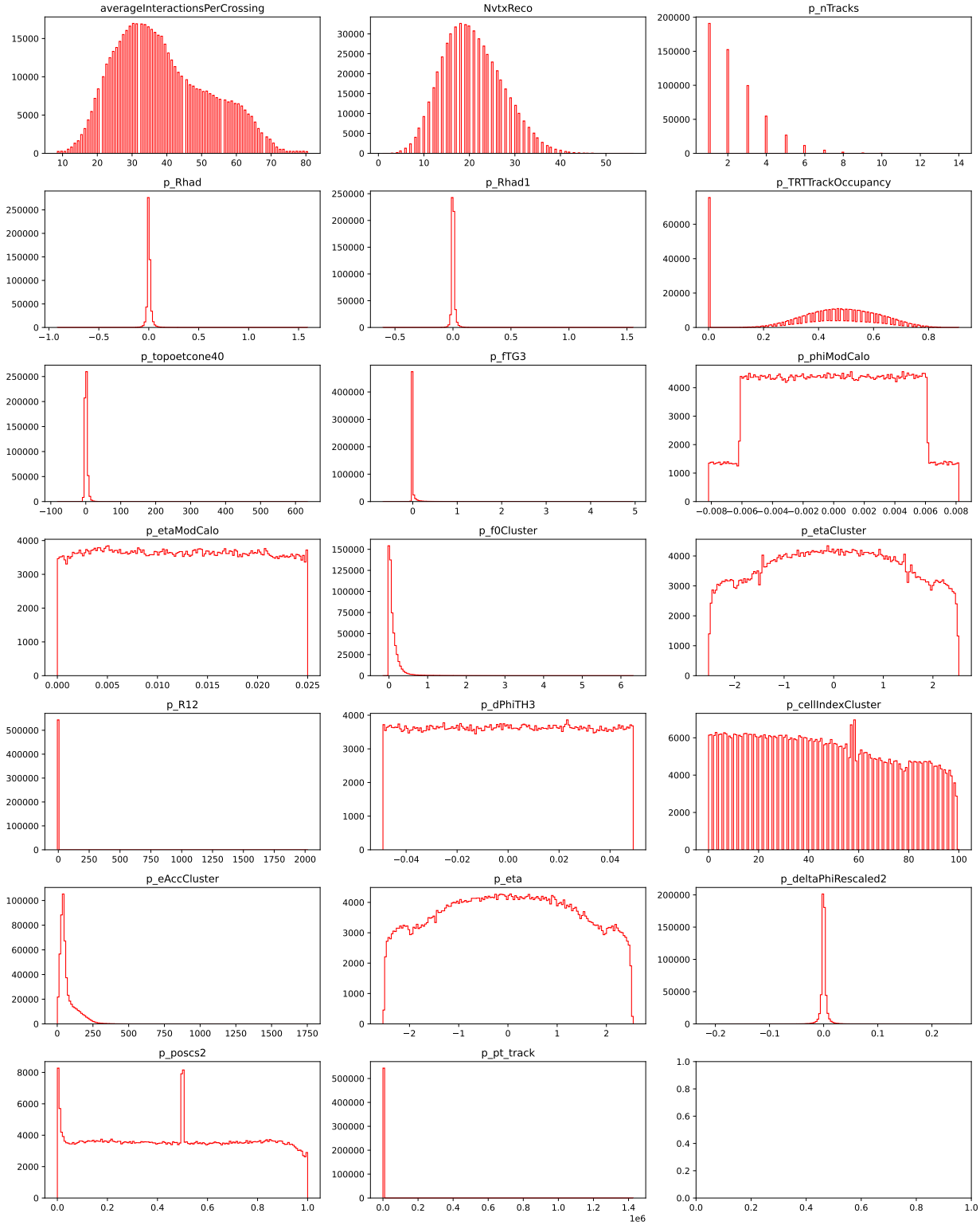


Figure 10.1: Histograms of all the different scalar variable distributions which are included in the final model.

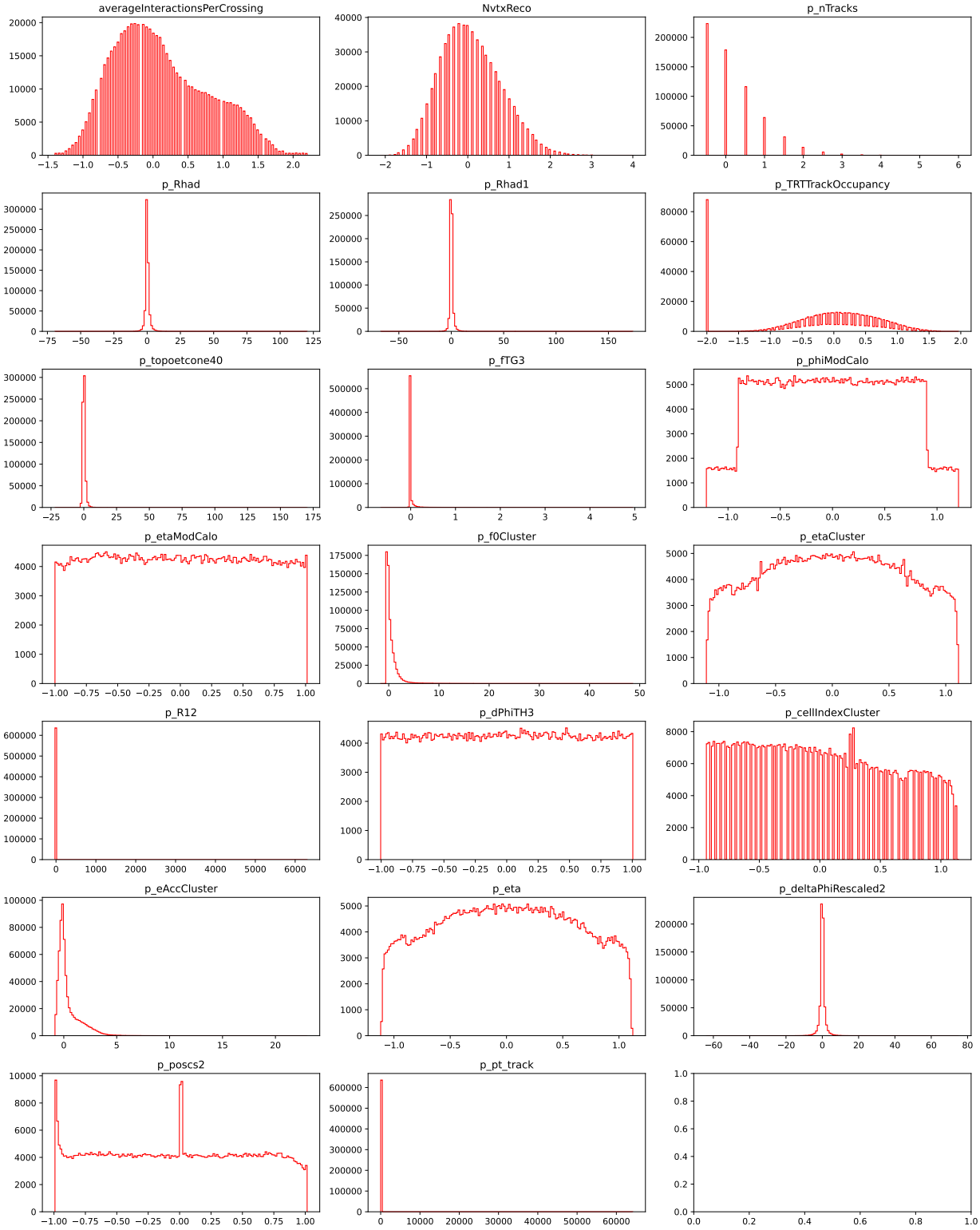


Figure 10.2: Histograms of all the different scalar variable distributions which are included in the final model, as they appear after transformations

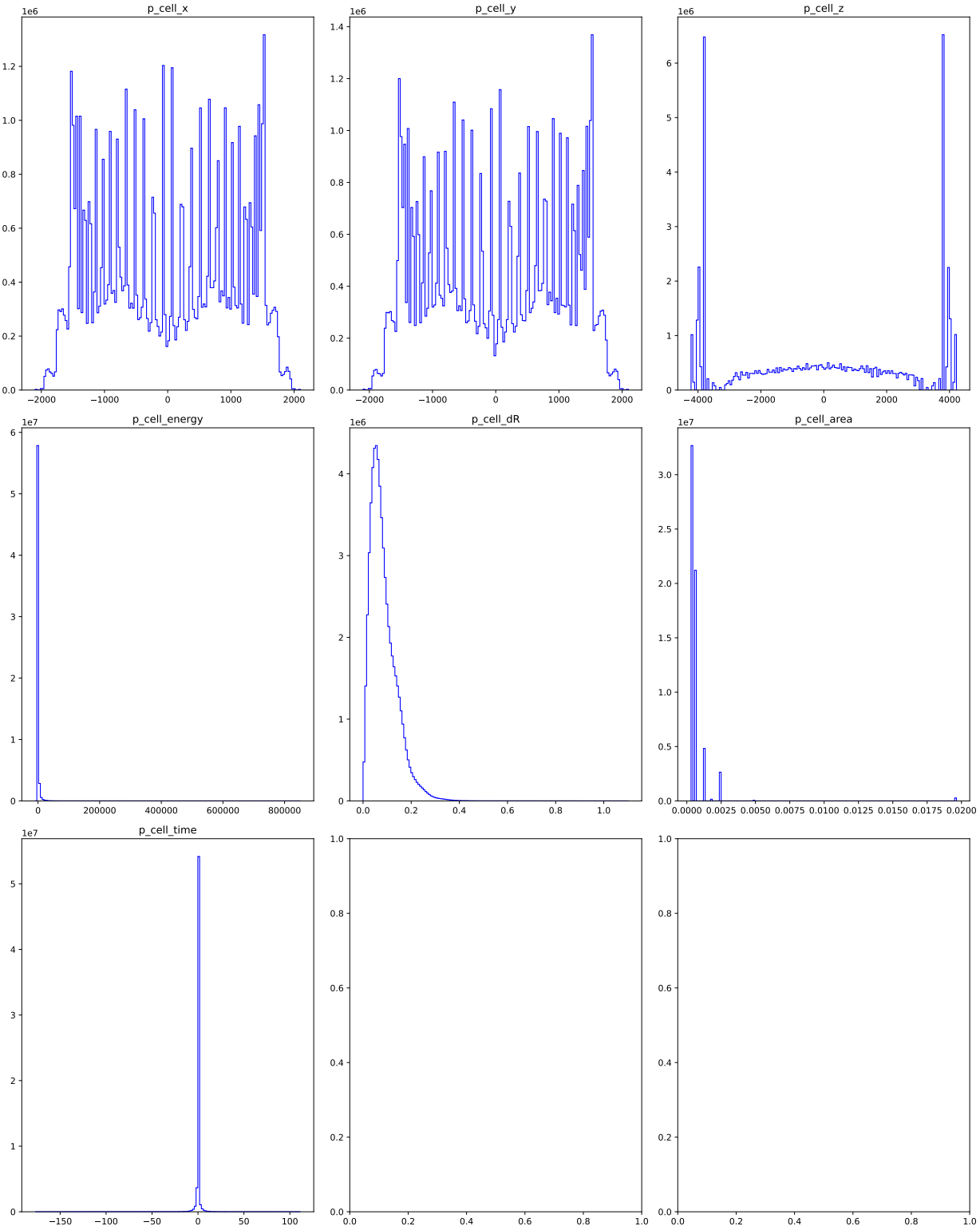


Figure 10.3: Histograms of all the different cell variable distributions which are included in the final model

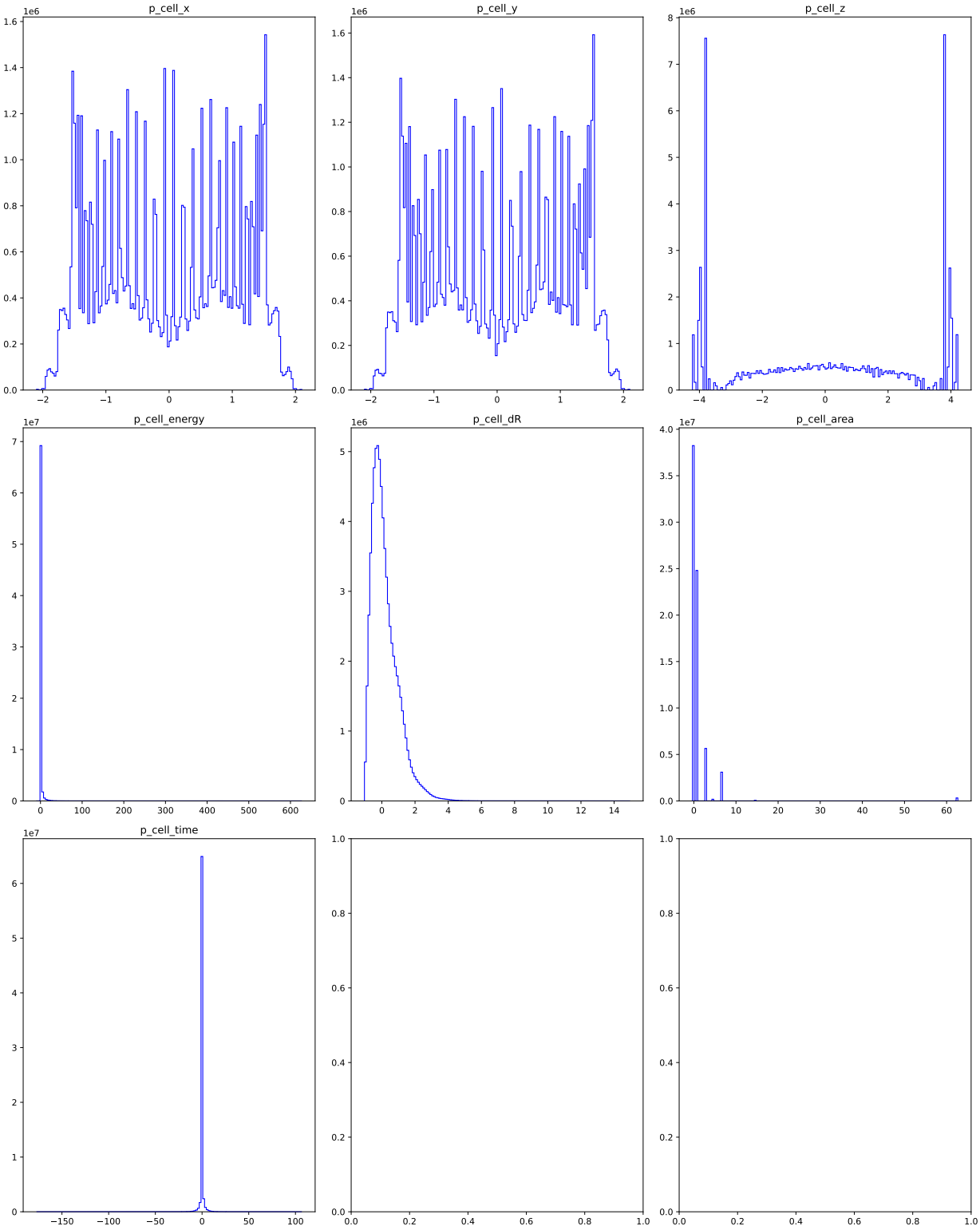


Figure 10.4: Histograms of all the different cell variable distributions which are included in the final model, as they appear after transformations

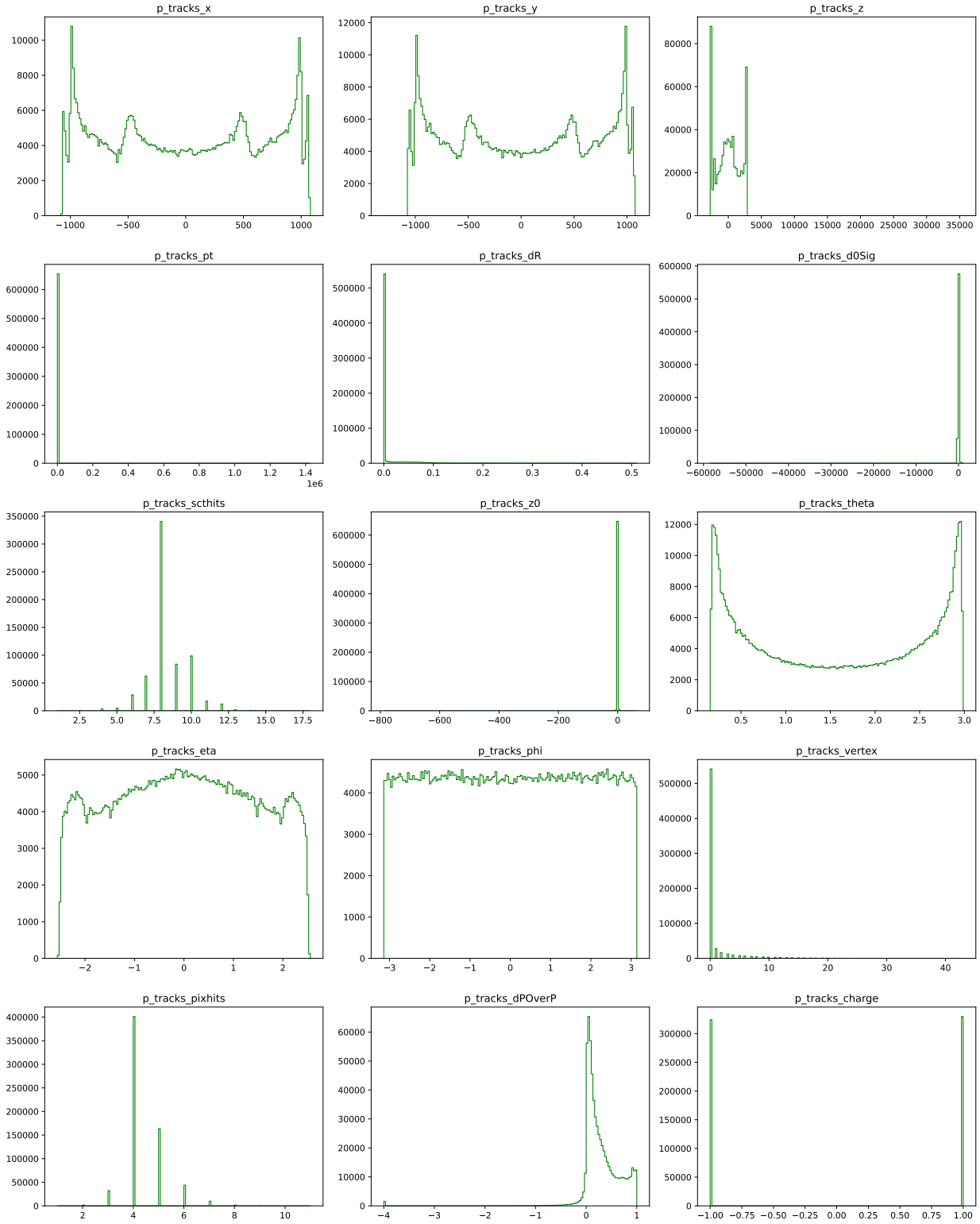


Figure 10.5: Histograms of all the different track variable distributions which are included in the final model.

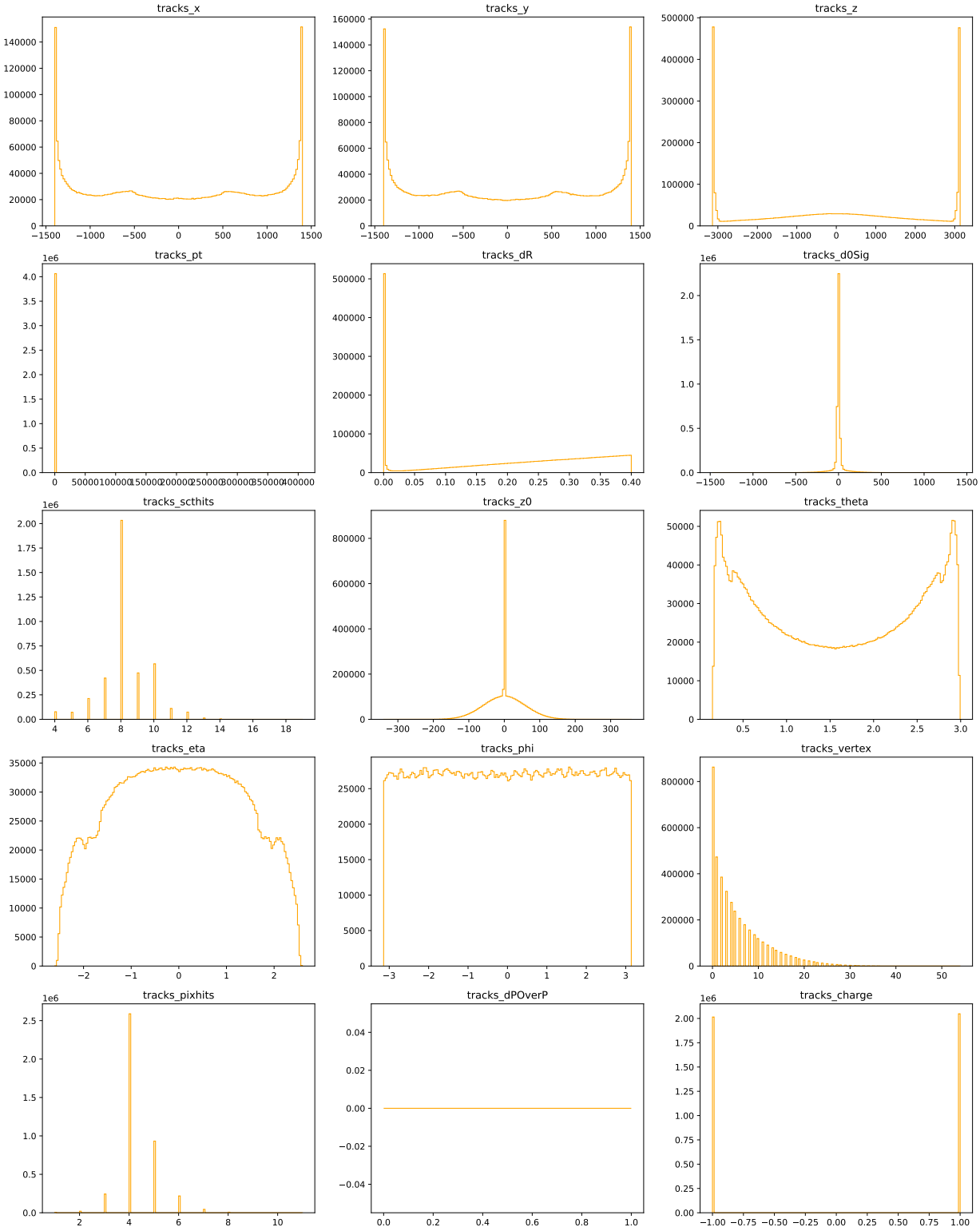


Figure 10.6: Histograms of all the different track variable distributions which are included in the final model.

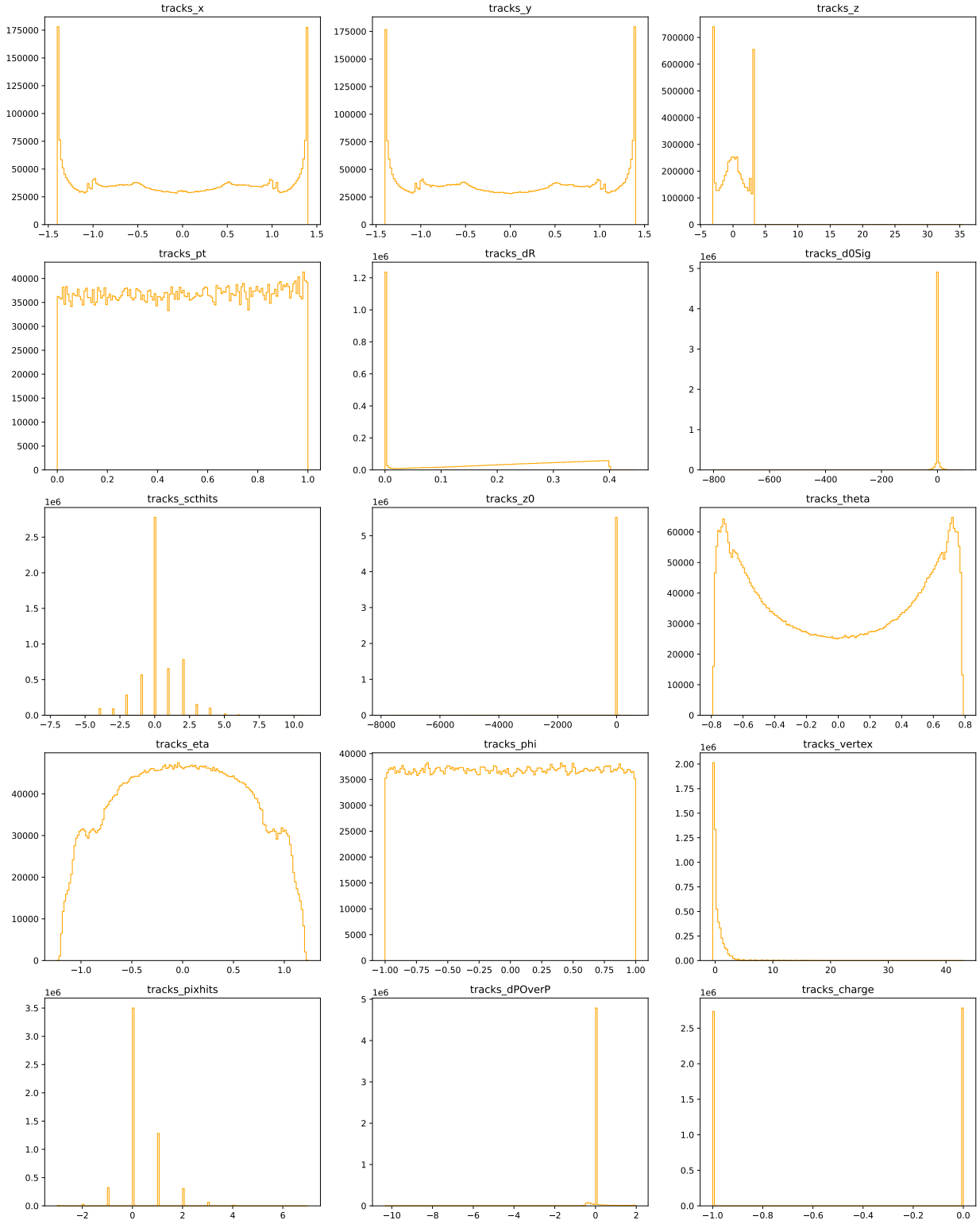


Figure 10.7: Histograms of the combined track distributions for both tracks and probe tracks as they appear in the graphs, that is after transformations.

10.2 *Sub-model figures*

The results from the testing of the sub-models.

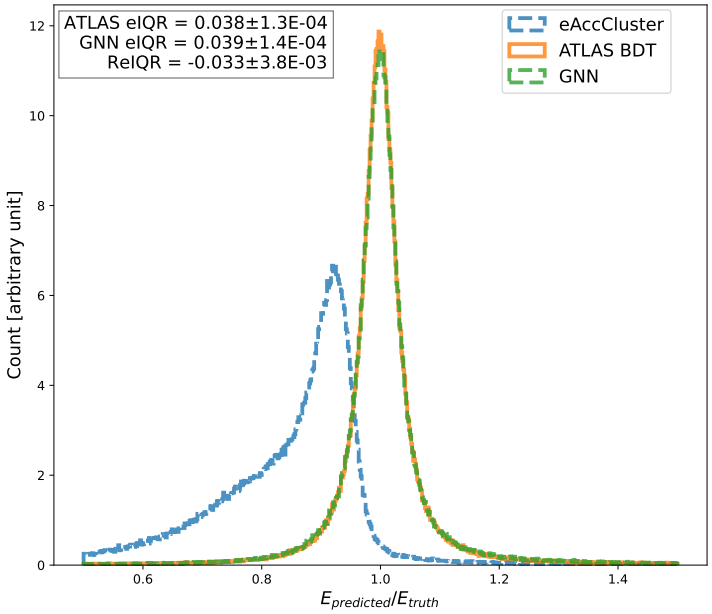


Figure 10.8: Histogram of the GNN and ATLAS BDT predictions over the MC truth value for the cell net sub-model.

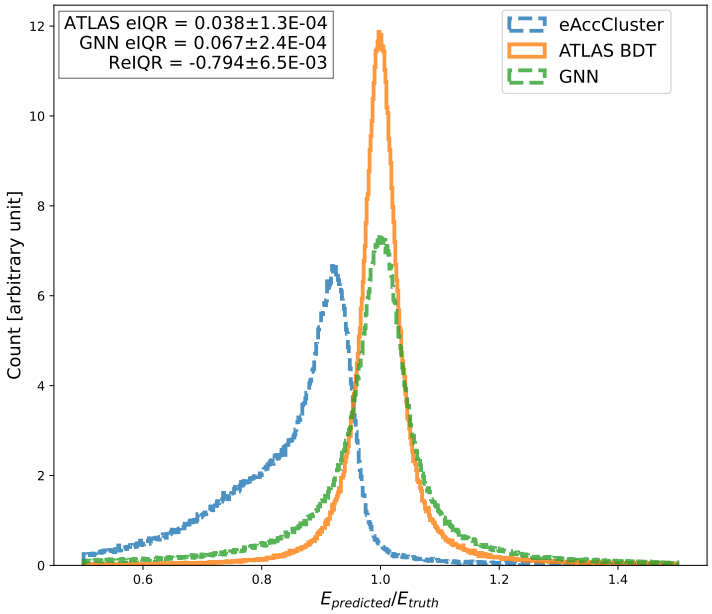


Figure 10.9: Histogram of the GNN and ATLAS BDT predictions over the MC truth value for the track net sub-model.

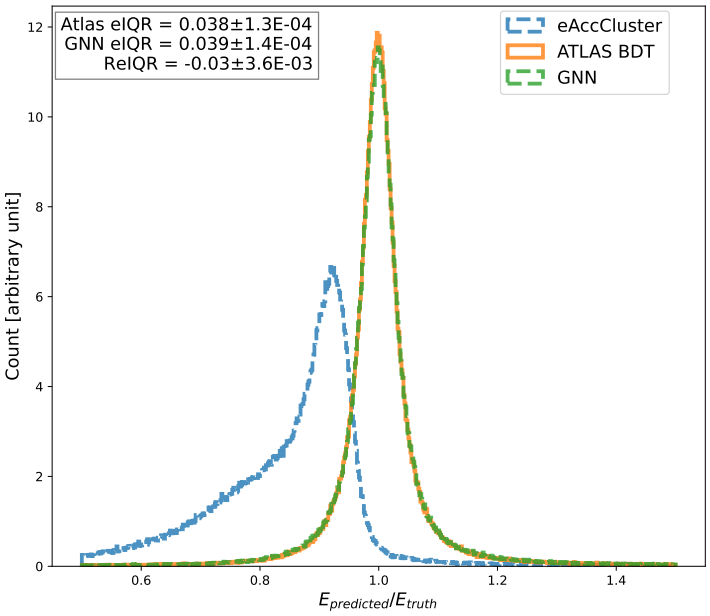


Figure 10.10: Histogram of the GNN and ATLAS BDT predictions over the MC truth value for the scalar net sub-model.

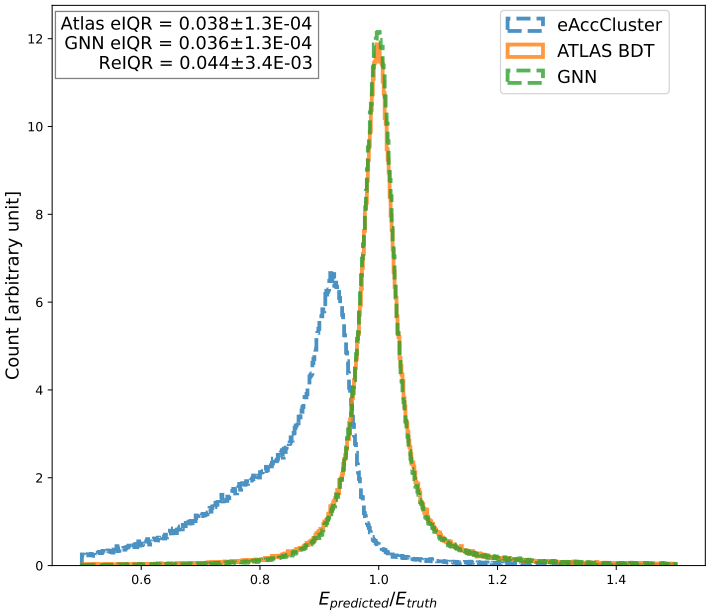


Figure 10.11: Histogram of the GNN and ATLAS BDT predictions over the MC truth value for the cell+scalar net sub-model.

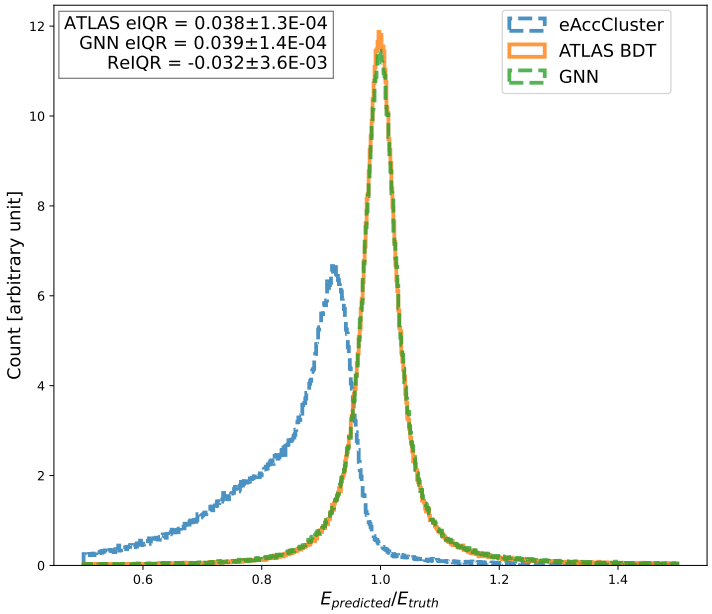


Figure 10.12: Histogram of the GNN and ATLAS BDT predictions over the MC truth value for the track+scalar net sub-model.

Bibliography

[1] M. Aaboud et al. "Electron and photon energy calibration with the ATLAS detector using 2015–2016 LHC proton-proton collision data." en. In: *Journal of Instrumentation* 14.03 (Mar. 2019). Publisher: IOP Publishing, P03017–P03017. ISSN: 1748-0221. DOI: 10.1088/1748-0221/14/03/P03017. URL: https://doi.org/10.1088/1748-0221/14/03/p03017 (visited on 04/21/2022).

[2] M. Aaboud et al. "Performance of the ATLAS trigger system in 2015." en. In: *The European Physical Journal C* 77.5 (May 2017), p. 317. ISSN: 1434-6052. DOI: 10.1140/epjc/s10052-017-4852-3. URL: https://doi.org/10.1140/epjc/s10052-017-4852-3 (visited on 05/18/2022).

[3] G. Aad et al. "Electron and photon performance measurements with the ATLAS detector using the 2015–2017 LHC proton-proton collision data." en. In: *Journal of Instrumentation* 14.12 (Dec. 2019). Publisher: IOP Publishing, P12006–P12006. ISSN: 1748-0221. DOI: 10.1088/1748-0221/14/12/P12006. URL: https://doi.org/10.1088/1748-0221/14/12/p12006 (visited on 03/31/2022).

[4] G. Aad et al. "The ATLAS Experiment at the CERN Large Hadron Collider." en. In: *JINST* 3 (2008). ISBN: 9789290833482, S08003. DOI: 10.1088/1748-0221/3/08/S08003. URL: https://cds.cern.ch/record/1129811 (visited on 04/08/2022).

[5] Takuya Akiba et al. *Optuna: A Next-generation Hyperparameter Optimization Framework*. Number: arXiv:1907.10902 arXiv:1907.10902 [cs, stat]. July 2019. DOI: 10.48550/arXiv.1907.10902. URL: http://arxiv.org/abs/1907.10902 (visited on 06/26/2022).

[6] Malte Algren. *Improving e & g reconstruction at ATLAS using a convolutional neural network*. 2021.

[7] Paolo Francavilla and. "The ATLAS Tile Hadronic Calorimeter performance at the LHC." en. In: *Journal of Physics: Conference Series* 404 (Dec. 2012). Publisher: IOP Publishing, p. 012007. ISSN: 1742-6596. DOI: 10.1088/1742-6596/404/1/012007. URL: https://doi.org/10.1088/1742-6596/404/1/012007 (visited on 04/12/2022).

[8] B. Andersson et al. "Parton fragmentation and string dynamics." en. In: *Physics Reports* 97.2 (July 1983), pp. 31–145. ISSN: 0370-1573. DOI: 10.1016/0370-1573(83)90080-7. URL: https://www.sciencedirect.com/science/article/pii/0370157383900807 (visited on 02/18/2022).

[9] Mikhail Belkin et al. "Reconciling modern machine learning practice and the bias-variance trade-off." In: *arXiv:1812.11118 [cs, stat]* (Sept. 2019). arXiv: 1812.11118. URL: http://arxiv.org/abs/1812.11118 (visited on 03/29/2022).

[10] Eric Brochu, Vlad M. Cora, and Nando de Freitas. *A Tutorial on Bayesian Optimization of Expensive Cost Functions, with Application to Active User Modeling and Hierarchical Reinforcement Learning*. Number: arXiv:1012.2599 arXiv:1012.2599 [cs]. Dec. 2010. DOI: 10.48550/arXiv.1012.2599. URL: http://arxiv.org/abs/1012.2599 (visited on 06/26/2022).

[11] P. Busch. "The Time-Energy Uncertainty Relation." In: *arXiv:quant-ph/0105049* 734 (2007). arXiv: quant-ph/0105049, pp. 73–105. DOI: 10.1007/978-3-540-73473-4_3. URL: http://arxiv.org/abs/quant-ph/0105049 (visited on 02/17/2022).

[12] Katy Börner, Soma Sanyal, and Alessandro Vespignani. "Network Science." In: *In Annual Review of Information Science & Technology*. 2007, pp. 537–607.

[13] Tianle Cai et al. *GraphNorm: A Principled Approach to Accelerating Graph Neural Network Training*. arXiv:2009.03294 [cs, math, stat]. June 2021. DOI: 10.48550/arXiv.2009.03294. URL: http://arxiv.org/abs/2009.03294 (visited on 06/30/2022).

[14] Junghoon Chae et al. *Visualization for Classification in Deep Neural Networks*. en. 2017. URL: https://www.semanticscholar.org/paper/Visualization-for-Classification-in-Deep-Neural-Chae-Gao/5028de2dddef306fc92e907922e464426c7e6789 (visited on 06/26/2022).

[15] S. Chatrchyan et al. "Observation of a new boson at a mass of 125 GeV with the CMS experiment at the LHC." en. In: *Physics Letters B* 716.1 (Sept. 2012), pp. 30–61. ISSN: 0370-2693. DOI: 10.1016/j.physletb.2012.08.021. URL: https://www.sciencedirect.com/science/article/pii/S0370269312008581 (visited on 03/31/2022).

[16] François Chollet and others. *Keras*. 2015. URL: https://keras.io.

[17] ATLAS Collaboration. *Athena*. Language: eng. May 2021. DOI: 10.5281/zenodo.4772550. URL: https://zenodo.org/record/4772550 (visited on 04/20/2022).

[18] ATLAS Collaboration. "Operation and performance of the ATLAS semiconductor tracker." In: *Journal of Instrumentation* 9.08 (Aug. 2014). arXiv: 1404.7473, P08009–P08009. ISSN: 1748-0221. DOI: 10.1088/1748-0221/9/08/P08009. URL: http://arxiv.org/abs/1404.7473 (visited on 04/06/2022).

[19] ATLAS Collaboration. "Performance of the ATLAS Track Reconstruction Algorithms in Dense Environments in LHC Run 2." In: *The European Physical Journal C* 77.10 (Oct. 2017). arXiv:1704.07983 [hep-ex], p. 673. ISSN: 1434-6044, 1434-6052. DOI: 10.1140/epjc/s10052-017-5225-7. URL: http://arxiv.org/abs/1704.07983 (visited on 05/19/2022).

[20] The ATLAS Collaboration. "Observation of a new particle in the search for the Standard Model Higgs boson with the ATLAS detector at the LHC." In: *Physics Letters B* 716.1 (Sept. 2012). arXiv: 1207.7214, pp. 1–29. ISSN: 03702693. DOI: 10.1016/j.physletb.2012.08.020. URL: http://arxiv.org/abs/1207.7214 (visited on 03/31/2022).

[21] *Crystal Ball function*. en. Page Version ID: 990932753. Nov. 2020. URL: https://en.wikipedia.org/w/index.php?title=Crystal_Ball_function&oldid=990932753 (visited on 05/28/2022).

[22] *CS231n Convolutional Neural Networks for Visual Recognition*. URL: https://cs231n.github.io/neural-networks-3/ (visited on 04/13/2022).

[23] *DerivationFramework < AtlasProtected < TWiki*. URL: https://twiki.cern.ch/twiki/bin/viewauth/AtlasProtected/DerivationFramework (visited on 04/20/2022).

[24] Matthew J. Dolan, Christoph Englert, and Michael Spannowsky. "Higgs self-coupling measurements at the LHC." en. In: (June 2012). DOI: 10.1007/JHEP10(2012)112. URL: https://arxiv.org/abs/1206.5001v2 (visited on 02/22/2022).

[25] Joel Duarte et al. *Improving Di-Higgs Sensitivity at Future Colliders in Hadronic Final States with Machine Learning*. Mar. 2022.

[26] *EGammaxAODDerivations < AtlasProtected < TWiki*. URL: https://twiki.cern.ch/twiki/bin/view/AtlasProtected/EGammaxAODDerivations#Derivations_defined_for_egamma (visited on 04/29/2022).

[27] "Electron and photon reconstruction and performance in ATLAS using a dynamical, topological cell clustering-based approach." In: (Dec. 2017).

[28] John Ellis. "Higgs Physics." In: *arXiv:1312.5672 [hep-ex, physics:hep-ph]* (Dec. 2013). arXiv: 1312.5672. URL: http://arxiv.org/abs/1312.5672 (visited on 02/21/2022).

[29] *Fifty years of quarks*. en. URL: https://home.cern/news/news/physics/fifty-years-quarks (visited on 02/23/2022).

[30] Aaron Fisher, Cynthia Rudin, and Francesca Dominici. *All Models are Wrong, but Many are Useful: Learning a Variable's Importance by Studying an Entire Class of Prediction Models Simultaneously*. Number: arXiv:1801.01489 arXiv:1801.01489 [stat]. Dec. 2019. DOI: 10.48550/arXiv.1801.01489. URL: http://arxiv.org/abs/1801.01489 (visited on 06/26/2022).

[31] Justin Gilmer et al. *Neural Message Passing for Quantum Chemistry*. Number: arXiv:1704.01212 arXiv:1704.01212 [cs]. June 2017. DOI: 10.48550/arXiv.1704.01212. URL: http://arxiv.org/abs/1704.01212 (visited on 06/26/2022).

[32] Ross Girshick. *Fast R-CNN*. arXiv:1504.08083 [cs]. Sept. 2015. DOI: 10.48550/arXiv.1504.08083. URL: http://arxiv.org/abs/1504.08083 (visited on 06/30/2022).

[33] Walter D. Goldberger, Benjamin Grinstein, and Witold Skiba. "Light scalar at LHC: the Higgs or the dilaton?" en. In: (Aug. 2007). DOI: 10.1103/PhysRevLett.100.111802. URL: https://arxiv.org/abs/0708.1463v1 (visited on 02/22/2022).

[34] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[35] Heather M Gray. "Track reconstruction with the ATLAS experiment." en. In: (), p. 52.

[36] Aidan Grummer. *Measurement of the effective silicon band gap energy with the ATLAS Pixel detector*. eng. 2021.

[37] Rajat Gupta. *Getting started with Neural Network for regression and Tensorflow*. en. June 2017. URL: https://medium.com/@rajatgupta310198/getting-started-with-neural-network-for-regression-and-tensorflow-58ad3bd75223 (visited on 06/28/2022).

[38] Richard H. R. Hahnloser et al. "Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit." In: *Nature* 405.6789 (June 2000), pp. 947–951. ISSN: 1476-4687. DOI: 10.1038/35016072. URL: https://doi.org/10.1038/35016072.

[39] L. A. Harland-Lang et al. "Parton distributions in the LHC era: MMHT 2014 PDFs." en. In: *The European Physical Journal C* 75.5 (May 2015), p. 204. ISSN: 1434-6052. DOI: 10.1140/epjc/s10052-015-3397-6. URL: https://doi.org/10.1140/epjc/s10052-015-3397-6 (visited on 02/23/2022).

[40] Wassily Hoeffding. "Probability Inequalities for Sums of Bounded Random Variables." In: *Journal of the American Statistical Association* 58.301 (1963). Publisher: [American Statistical Association, Taylor & Francis, Ltd.], pp. 13–30. ISSN: 0162-1459. DOI: 10.2307/2282952. URL: https://www.jstor.org/stable/2282952 (visited on 03/29/2022).

[41] Vipul J. *Cyclical Learning Rates — The ultimate guide for setting learning rates for Neural Networks*. en. Aug. 2019. URL: https://medium.com/swlh/cyclical-learning-rates-the-ultimate-guide-for-setting-learning-rates-for-neural-networks-3104e906f0ae (visited on 06/29/2022).

[42] Shruti Jadon. *Introduction to Different Activation Functions for Deep Learning*. en. Feb. 2022. URL: https://medium.com/@shrutijadon/survey-on-activation-functions-for-deep-learning-9689331ba092 (visited on 03/30/2022).

[43] Xiangyang Ju and Benjamin Nachman. "Supervised Jet Clustering with Graph Neural Networks for Lorentz Boosted Bosons." In: *Physical Review D* 102.7 (Oct. 2020). arXiv: 2008.06064, p. 075014. ISSN: 2470-0010, 2470-0029. DOI: 10.1103/PhysRevD.102.075014. URL: http://arxiv.org/abs/2008.06064 (visited on 04/29/2022).

[44] J. Kiefer and J. Wolfowitz. "Stochastic Estimation of the Maximum of a Regression Function." In: *The Annals of Mathematical Statistics* 23.3 (Sept. 1952). Publisher: Institute of Mathematical Statistics, pp. 462–466. ISSN: 0003-4851, 2168-8990. DOI: 10.1214/aoms/1177729392. URL: https://projecteuclid.org/journals/annals-of-mathematical-statistics/volume-23/issue-3/Stochastic-Estimation-of-the-Maximum-of-a-Regression-Function/10.1214/aoms/1177729392.full (visited on 03/30/2022).

[45] *LARG3-TDR-barrelM_samplings_presamp_new.png (PNG Image, 800 × 650 pixels)*. URL: https://twiki.cern.ch/twiki/pub/AtlasPublic/LArCaloPublicResultsDetStatus/LARG3-TDR-barrelM_samplings_presamp_new.png (visited on 04/08/2022).

[46] Y. LeCun et al. "Backpropagation Applied to Handwritten Zip Code Recognition." In: *Neural Computation* 1.4 (Dec. 1989). Conference Name: Neural Computation, pp. 541–551. ISSN: 0899-7667. DOI: 10.1162/neco.1989.1.4.541.

[47] Zhiyuan Liu and Jie Zhou. "Introduction to Graph Neural Networks." In: *Synthesis Lectures on Artificial Intelligence and Machine Learning* 14.2 (Mar. 2020). Publisher: Morgan & Claypool Publishers, pp. 1–127. ISSN: 1939-4608. DOI: 10.2200/S00980ED1V01Y202001AIM045. URL: https://www.morganclaypool.com/doi/10.2200/S00980ED1V01Y202001AIM045 (visited on 04/12/2022).

[48] *LuminosityPublicResultsRun2 < AtlasPublic < TWiki*. URL: https://twiki.cern.ch/twiki/bin/view/AtlasPublic/LuminosityPublicResultsRun2#2018_pp_Collisions (visited on 03/31/2022).

[49] Simona Maggio. *The Learning Rate Finder Technique: How Reliable Is It?* en-us. URL: https://blog.dataiku.com/the-learning-rate-finder-technique-how-reliable-is-it (visited on 06/29/2022).

[50] Brian R. Martin. *Nuclear and Particle Physics: An Introduction*. English. 2nd edition. Chichester, U.K: Wiley, Mar. 2009. ISBN: 978-0-470-74275-4.

[51] Grégoire Montavon, Wojciech Samek, and Klaus-Robert Müller. "Methods for Interpreting and Understanding Deep Neural Networks." In: *Digital Signal Processing* 73 (Feb. 2018). arXiv:1706.07979 [cs, stat], pp. 1–15. ISSN: 10512004. DOI: 10.1016/j.dsp.2017.10.011. URL: http://arxiv.org/abs/1706.07979 (visited on 06/26/2022).

[52] Thilo Moshagen, Nihal Acharya Adde, and Ajay Navilarekal Rajgopal. *Finding hidden-feature depending laws inside a data set and classifying it using Neural Network*. arXiv:2101.10427 [cs, stat]. Jan. 2021. DOI: 10.48550/arXiv.2101.10427. URL: http://arxiv.org/abs/2101.10427 (visited on 06/30/2022).

[53] R Nowak. "Lecture 7: PAC bounds and Concentration of Measure." en. In: (), p. 6.

[54] M. Oreglia. "A Study of the Reactions $\psi^\prime \to \gamma \gamma \psi$." Other thesis. Dec. 1980.

[55] Yoshihiko Ozaki et al. "Multiobjective tree-structured parzen estimator for computationally expensive optimization problems." In: *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*. GECCO '20. New York, NY, USA: Association for Computing Machinery, June 2020, pp. 533–541. ISBN: 978-1-4503-7128-5. DOI: 10.1145/3377930.3389817. URL: https://doi.org/10.1145/3377930.3389817 (visited on 06/26/2022).

[56] Adam Paszke et al. "Automatic differentiation in PyTorch." en. In: (Oct. 2017). URL: https://openreview.net/forum?id=BJJsrmfCZ (visited on 05/27/2022).

[57] Fabian Pedregosa et al. "Scikit-learn: Machine Learning in Python." In: *Journal of Machine Learning Research* 12.85 (2011), pp. 2825–2830. ISSN: 1533-7928. URL: http://jmlr.org/papers/v12/pedregosa11a.html (visited on 06/28/2022).

[58] H. Pernegger. "The Pixel Detector of the ATLAS experiment for LHC Run-2." en. In: *Journal of Instrumentation* 10.06 (June 2015). Publisher: IOP Publishing, pp. C06012–C06012. ISSN: 1748-0221. DOI: 10.1088/1748-0221/10/06/C06012. URL: https://doi.org/10.1088/1748-0221/10/06/c06012 (visited on 04/03/2022).

[59] Andrew Purcell. "Go on a particle quest at the first CERN webfest." en. In: *CERN Document Server*. Number: BUL-NA-2012-269. Aug. 2012, p. 10. URL: https://cds.cern.ch/record/1473657 (visited on 02/18/2022).

[60] Huilin Qu and Loukas Gouskos. "Jet tagging via particle clouds." In: *Physical Review D* 101.5 (Mar. 2020). Publisher: American Physical Society, p. 056019. DOI: 10.1103/PhysRevD.101.056019. URL: https://link.aps.org/doi/10.1103/PhysRevD.101.056019 (visited on 04/29/2022).

[61] Herbert Robbins and Sutton Monro. "A Stochastic Approximation Method." In: *The Annals of Mathematical Statistics* 22.3 (Sept. 1951). Publisher: Institute of Mathematical Statistics, pp. 400–407. ISSN: 0003-4851, 2168-8990. DOI: 10.1214/aoms/1177729586. URL: https://projecteuclid.org/journals/annals-of-mathematical-statistics/volume-22/issue-3/A-Stochastic-Approximation-Method/10.1214/aoms/1177729586.full (visited on 03/30/2022).

[62] "Mean Absolute Error." en. In: *Encyclopedia of Machine Learning*. Ed. by Claude Sammut and Geoffrey I. Webb. Boston, MA: Springer US, 2010, pp. 652–652. ISBN: 978-0-387-30164-8. DOI: 10.1007/978-0-387-30164-8_525. URL: https://doi.org/10.1007/978-0-387-30164-8_525 (visited on 06/28/2022).

[63] "Mean Squared Error." en. In: *Encyclopedia of Machine Learning*. Ed. by Claude Sammut and Geoffrey I. Webb. Boston, MA: Springer US, 2010, pp. 653–653. ISBN: 978-0-387-30164-8. DOI: 10.1007/978-0-387-30164-8_528. URL: https://doi.org/10.1007/978-0-387-30164-8_528 (visited on 06/28/2022).

[64] Matthias Schott and Monica Dunford. "Review of single vector boson production in pp collisions at $\sqrts = 7$ TeV." In: *Eur. Phys. J. C* 74 (2014). _eprint: 1405.1160, p. 2916. DOI: 10.1140/epjc/s10052-014-2916-1.

[65] Anna Sfyrla et al. "The Detector Control System for the ATLAS Semiconductor Tracker Assembly Phase." In: *Nuclear Science, IEEE Transactions on* 52 (Sept. 2005), pp. 938–943. DOI: 10.1109/TNS.2005.852737.

[66] Atefeh Shahroudnejad. *A Survey on Understanding, Visualizations, and Explanation of Deep Neural Networks*. Number: arXiv:2102.01792 arXiv:2102.01792 [cs]. Feb. 2021. DOI: 10.48550/arXiv.2102.01792. URL: http://arxiv.org/abs/2102.01792 (visited on 06/26/2022).

[67] Torbjörn Sjöstrand et al. "An Introduction to PYTHIA 8.2." In: *Computer Physics Communications* 191 (June 2015). arXiv: 1410.3012, pp. 159–177. ISSN: 00104655. DOI: 10.1016/j.cpc.2015.01.024. URL: http://arxiv.org/abs/1410.3012 (visited on 02/18/2022).

[68] Leslie N. Smith. *Cyclical Learning Rates for Training Neural Networks*. arXiv:1506.01186 [cs]. Apr. 2017. DOI: 10.48550/arXiv.1506.01186. URL: http://arxiv.org/abs/1506.01186 (visited on 06/29/2022).

[69] A. Sperduti and A. Starita. "Supervised neural networks for the classification of structures." In: *IEEE Transactions on Neural Networks* 8.3 (May 1997). Conference Name: IEEE Transactions on Neural Networks, pp. 714–735. ISSN: 1941-0093. DOI: 10.1109/72.572108.

[70] *SQLite Home Page*. URL: https://www.sqlite.org/index.html (visited on 04/29/2022).

[71] *The Inner Detector*. en. URL: https://atlas.cern/Discover/Detector/Inner-Detector (visited on 04/07/2022).

[72] *UX: Standard Model of the Standard Model*. en-US. Nov. 2013. URL: http://davidgalbraith.org/portfolio/ux-standard-model-of-the-standard-model/ (visited on 02/23/2022).

[73] V. Vapnik. "Principles of Risk Minimization for Learning Theory." In: *Advances in Neural Information Processing Systems*. Vol. 4. Morgan-Kaufmann, 1991. URL: https://proceedings.neurips.cc/paper/1991/hash/ff4d5fbbafdf976cfdc032e3bde78de5-Abstract.html (visited on 03/28/2022).

[74] Yue Wang et al. *Dynamic Graph CNN for Learning on Point Clouds*. Number: arXiv:1801.07829 arXiv:1801.07829 [cs]. June 2019. DOI: 10.48550/arXiv.1801.07829. URL: http://arxiv.org/abs/1801.07829 (visited on 06/26/2022).

[75] Rikiya Yamashita et al. "Convolutional neural networks: an overview and application in radiology." eng. In: *Insights into Imaging* 9.4 (Aug. 2018), pp. 611–629. ISSN: 1869-4101. DOI: 10.1007/s13244-018-0639-9.

[76] Daniele Zanzi. "The ATLAS Trigger in 2017 and 2018: Developments and performance." In: *PoS* ICHEP2018 (2019), p. 197. DOI: 10.22323/1.340.0197.

[77] P.A. Zyla and others. "Review of Particle Physics." In: *PTEP* 2020.8 (2020), p. 083C01. DOI: 10.1093/ptep/ptaa104.

[78] Rasmus F. Ørsøe. *A Graph Neural Network Approach to Low Energy Event Reconstruction in IceCube Neutrino Observatory*. 2021.

List of Figures

List of Tables

Index