Universiteit Utrecht

Master Thesis

The description of high-energy electromagnetic cascades in a highly granular digital calorimeter

Author:
Paul Renes
4012402
Experimental Physics

Supervisor:
prof. dr. Thomas Peitzmann
Utrecht University

Daily supervisor:
dr. ing. Naomi Van der Kolk
Utrecht University

ADS supervisor:
dr. Marco Spruit
Utrecht University

Abstract

We use data obtained with the Forward Calorimeter (FoCal) prototype to take the first steps in performing a three-dimensional fit on shower profiles. Within these shower profiles we observe saturation effects for the innermost rings around the central shower axis. To study the effect of saturation more thoroughly, we construct a toy model that simulates a single pixel sensor in the FoCal. Using a uniform distribution of incoming particles and a fixed cluster size of 3 on a grid of size 100x100, we observe a 2% deviation from linear behaviour at 100 simulated particles. The effect increases gradually towards a higher number of simulated particles. Finally, we use the toy model to generate no-photon and 1-photon data classes that we use for the Applied Data Science project. Using a Convolutional Neural Network we obtain a binary classification accuracy of 0.912 ± 0.063 after 10 epochs.

27 November 2018

Contents

1 Introduction

1.1 ALICE

ALICE (A Large Ion Collider Experiment) is one of the main experiments at the Large Hadron Collider (LHC) at CERN. Its main purpose is the study of the Quark-Gluon Plasma (QGP) that is believed to arise in the collision of two nuclei. The experiment consists of a combination of subdetectors shown in Figure 1.

ALICE excels at charged particle tracking and charged particle identification using the Inner Tracking System (ITS), Time Projection Chamber (TPC) and the Transition Radiation Detector (TRD). Furthermore, the electromagnetic calorimeters PHOton Spetrometer (PHOS) and Electromagnetic Calorimeter (EMCal) are used in combination with the PMD to measure high-energy photons.

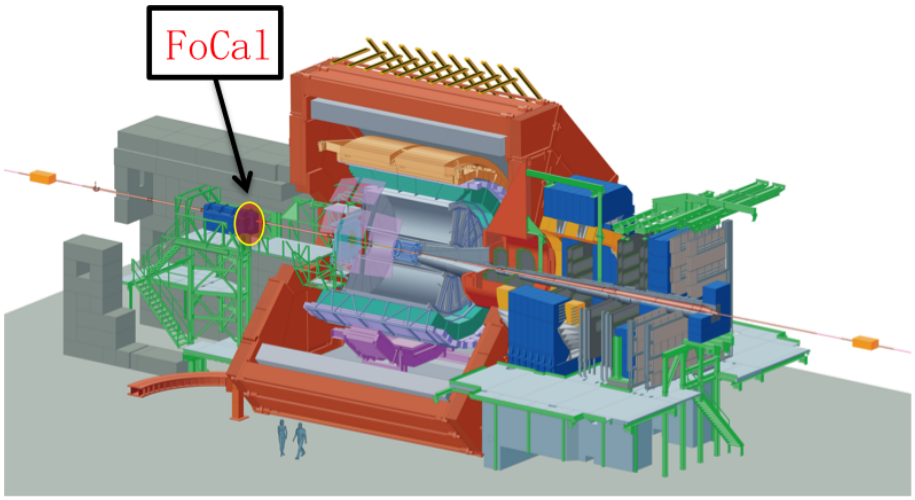


Figure 1 – [15] The ALICE experiment layout, showing all detectors used during measurements. The proposed position of the FoCal is indicated just outside of the magnet.

1.2 Physics motivation FoCal

The measurement of these high-energy photons is of importance to our understanding of the inner structure of protons and nuclei. The structure of protons and nuclei is described by parton distribution functions (PDF) [5]. These parton distribution functions are crucial for many observables within particle physics. For example, the number density of partons determines the cross section for a certain specific process. Furthermore, they determine the initial conditions for a Quark-Gluon Plasma. There are various models for the description of PDFs. A standard description uses perturbative QCD and uses on linear evolution equations. Alternative models implement phenomena which are due to non-linear behaviour. The most important effect of non-linear evolution would be gluon saturation.

Currently, these PDFs at low-momentum fraction x are not well constrained. Therefore, we want to do measurements that allow us to constrain the PDF at low x. The momentum fraction x of incoming partons can be approximated by [15]:

$$x \approx \frac{2p_T}{\sqrt{s}} e^{-y}, \tag{1}$$

where p_T is the transverse momentum and y the rapidity of the measured particle, and $\sqrt{s}$ is the center of mass energy of the collision. Furthermore, at high particle energies the rapidity y is approximately equal to the pseudorapidity

$$y \approx \eta = -\ln\left(\tan\left(\frac{\theta}{2}\right)\right) \tag{2}$$

where θ is the polar angle with respect to the beam axis. Therefore, small x-values can be obtained by measuring at low-p_T, high center of mass energy and high values of η, which correspond to small polar angles. Specifically, we want to measure in the forward region.

However, the current detectors that can measure high-energy photons are not capable of performing these specific measurements. The PHOS is positioned at the bottom of the ALICE experiment and covers $|\eta| \leq 0.12$ in pseudorapidity. The EMCal, another electromagnetic calorimeter in ALICE, is also part of the cylindrical detector system around the interaction point. It covers $|\eta| \leq 0.7$ in pseudorapidity and is mainly used for jet quenching measurements. As we want to perform our measurements at high pseudorapidity $\eta \geq 3$, we need a new calorimeter that allows us to do this. The proposed calorimeter is called the Forward Calorimeter (FoCal). Figure 2 shows the proposed position of both the electromagnetic and the hadronic part of the FoCal with respect to the rest of the ALICE experiment.

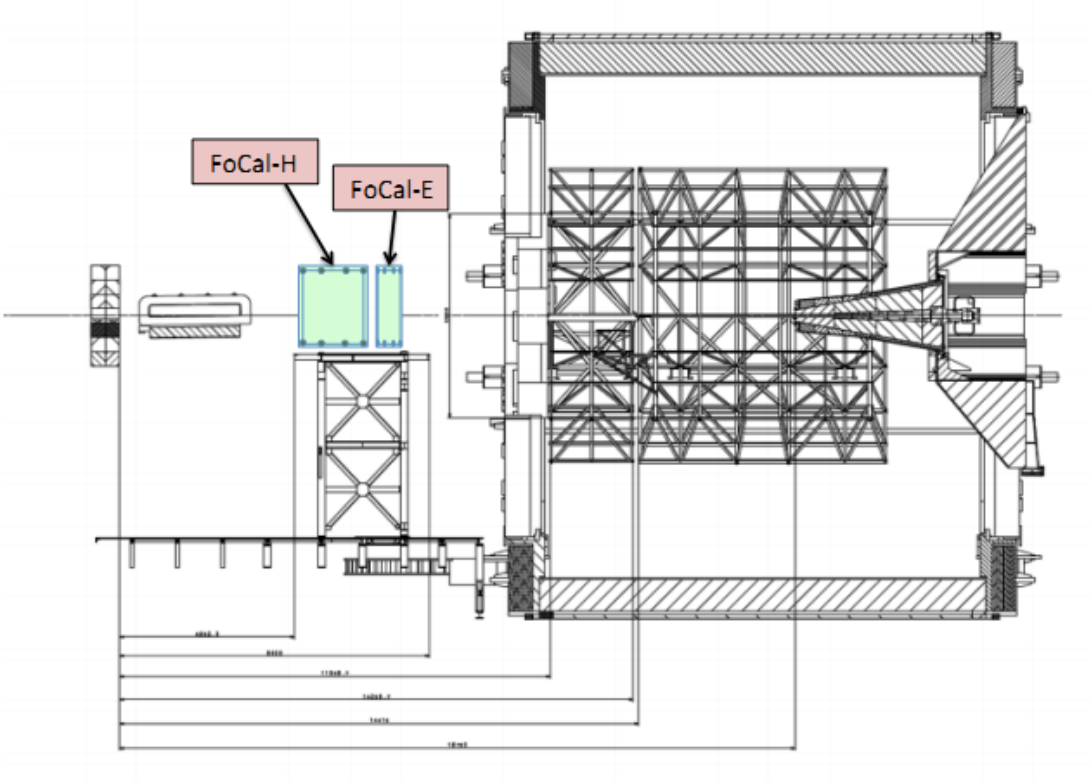


Figure 2 – [13] The ALICE experiment layout schematically. The proposed position of the FoCal is indicated just outside of the magnet. As shown it consists of an electromagnetic (E) part and an hadronic (H) part.

Moreover, in the mentioned measurement region many pions are produced. Neutral pions can decay via $\pi^0 \rightarrow \gamma\gamma$, thus creating two photons. Unfortunately, we are not interested in these decay photons but only in the direct photons that come from the interaction point. Our proposed new detector should therefore be able to distinguish between direct photons and two photons coming from a pion decay. To achieve this at small polar angles, the detector has to be highly granular. One additional advantage of high granularity is that it allows analyses of the 3D shower shape within the calorimeter.

1.3 Research goal

A lot of research has already been performed on the FoCal prototype [10, 15, 13]. Therefore, we can build on what has already been done to extend our knowledge. One of the advantages of the highly granular prototype is that it allows us to track the particles within the detector. Thus, in principle, we can obtain a three dimensional shower profile of all particles created during the

cascade. If we were able to construct a model to fit to such a shower profile, it would help us with for instance particle identification (PID) or the determination of the energy of the incoming particle.

Various models have already been tested for this very purpose. However, these models are fits to two dimensional profiles. Therefore, not all information on the shower profile is contained within those fits. The original goal of this research was to extend these models to three dimensions or construct a new three dimensional model, allowing us to capture as much information as possible in the parameters of that model. However, due to a crash of the quark-cluster in Utrecht we were forced to abandon that goal prematurely. This cluster contained the processed data that we had obtained with the prototype along with the code that was used for this processing. At that point we had two choices: either we could rewrite the processing code in order to obtain the processed data once more, or we could try to do something else.

From the research that we had already done we had seen several interesting properties of the data. First of all, the data we obtain from the FoCal is not perfect. We see effects of saturation in several layers of the prototype at high energies. To fix this, we wanted to use Monte Carlo simulations to find an alternative observable to the number of hits in order to generate a correction factor. Ideally, this correction factor would then solve our problem of saturation. Unfortunately, we were unable to achieve that goal. However, as the problem of saturation was something that we had to deal with, we decided not to rewrite all the code. As rewriting that would probably have taken up the remaining time of my research, we decided to build a toy model that would allow us to study the phenomenon of saturation.

Moreover, part of the research is dedicated to the Applied Data Science (ADS) profile. Therefore, we implement ADS techniques in several steps of our project. More specifically, we search for an algorithm that helps us determine the number of cascades in the detector. This could then be used as a classification technique to distinguish direct photons from decay photons.

In Section 2 we explain the concept of digital calorimetry which distinguishes the FoCal from contemporary calorimeters. Section 3 then cover the methods used for our analysis. This includes a description of the data as well as the toy model. In Section 4 we discuss the results we obtained with our initial analysis followed by the results obtained with our toy model. We discuss these results in Section 5. Subsequently, we explain our ADS project in Section 6 followed by our conclusion of the thesis.

2 Digital calorimetry

A large part of this research is based on data obtained with the FoCal prototype, which is a digital calorimeter. Therefore, we devote this section to a description of the main properties and parameters used in such a detector.

2.1 Calorimetry

The energy of a particle can be measured using a calorimeter [14]. When particles traverse the calorimeter, they lose energy and as a result they produce secondary particles. Subsequently, these secondary particles undergo the same process, giving rise to a cascade of particles in the detector. By measuring the deposited energy of all particle in the cascade, or shower, we obtain the energy of the initial incoming particle. For the purpose of FoCal, which is to measure photons, we focus on electromagnetic calorimeters as photons interact electromagnetically. However, particles that interact via the strong nuclear force can be measured by a hadronic calorimeter.

Two types of calorimeters exist in modern experiments: homogeneous and sampling calorimeters. To clarify the difference, we introduce the concepts of active and absorber material. Active material in a calorimeter is the material where the deposited energy can be measured. Absorber material is dense material that contributes to the development of the shower, but does not allow measurements. A homogeneous calorimeter consists solely of active material, which allows for measuring the total deposited energy. However, active material is not as dense as absorber material, so these detectors often take up a lot of space as they still need to contain the entire shower. The solution to this issue is to build a sampling calorimeter where layers of absorber material are followed by layers of active material. The shower then develops faster due to the presence of

absorber material and thus less space is required. However, the total deposited energy can not be measured but has to be derived from the energy deposited in the active layers of the detector.

A calorimeter has several properties that indicate the quality and performance. We briefly explain three of these. First of all, materials have a specific radiation length X_0 (g/cm^2) where a unit in X_0 corresponds to a particle being left with a fraction 1/e of its initial energy. It is given by [12]

$$X_0 = \frac{716 \cdot A}{Z(Z+1)\ln(287/\sqrt{Z})} \text{g cm}^{-2}, \tag{3}$$

where A is the mass number of the nucleus and Z the atomic number. The two most prominent effects resulting in the loss of energy are bremsstrahlung and ionization. Bremsstrahlung is produced when a charged particle is deflected by another charged particle and loses kinetic energy in the process. This energy is converted into a photon that then gets radiated. Ionization of atoms results in loss of energy as charged particles kick out electrons, thereby losing energy in the collision. Moreover, the critical energy E_c (eV) is the energy at which bremsstrahlung and ionization play an equally important role in the energy loss of a particle. Finally, the Molière radius R_M describes the width of an electromagnetic cascade. It is defined as the radius in which 90% of the total energy of the shower is deposited. In terms of X_0 and E_c, the Molière radius is given by [7, 1]

$$R_M = \frac{21.2 \text{ MeV}}{E_c} X_0. \tag{4}$$

2.2 Principle of FoCal

Unlike existing sampling calorimeters where each individual active layer is homogeneous, the FoCal implements digital pixel sensors as the active material in several layers. The design concept of the FoCal is shown in Figure 3. The digital pixel sensors serve as the high granularity cells. This high granularity is the key ingredient, as it should allow us to make the distinction between one high-energy photon or two decay photons. Furthermore, the high granularity layers provide information on the shower shape which could in turn be used for PID. The deposited energy in such a high granularity layer is then proportional to the number of fired pixels.

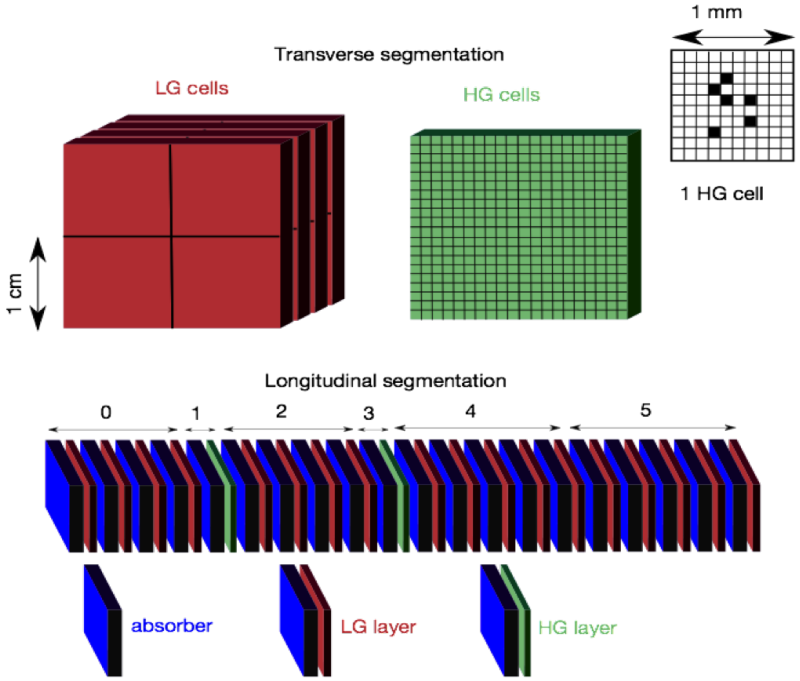


Figure 3 – [15] The proposed FoCal design concept. The detector consists of three different components: low granularity layers (red), high granularity layers (green) and absorbers (blue).

The rest of the detector is constructed as a sampling calorimeter with alternating absorber layers and low granularity layers that give accurate energy measurements. The proposed FoCal

uses tungsten (W) as absorber material and silicon (Si) as active material. Figure 4 shows in more detail how the different layers should be stacked on top of each other. An important note is that we should not forget the read-out boards and their contribution to the total radiation length. Each granularity pad, both low and high, requires a read-out board to process and store the obtained pixel data. Acquiring a reasonable speed for the read-out process is a subject on its own, and we do not go into further detail here.

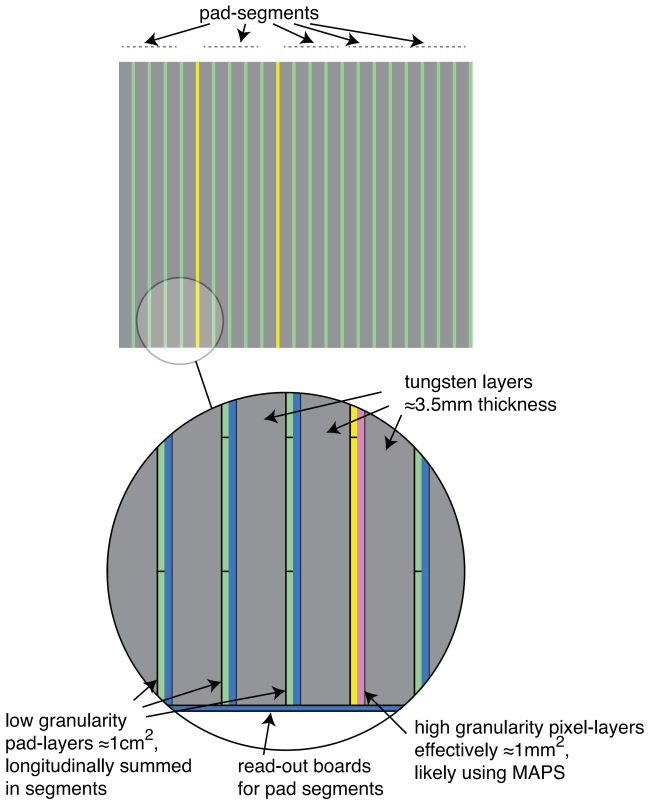


Figure 4 – [13] A schematic overview of the proposed FoCal showing the different layers. More information on this figure!

To test the principle of Si-W highly granular digital calorimeter layers used in the FoCal detector, a prototype has been built. As the purpose is to prove the functionality of such a digital calorimeter, the layers of the prototype do not correspond to those in the proposed FoCal detector. Alternatively, 24 layers consisting of a tungsten absorber volume followed by a silicon pixel sensor have been put together. A schematic overview of the prototype is shown in Figure 5. We see that a block of 20 mm tungsten has been placed between layers 21 and 22. When interpreting the data of these layers, we should keep that in mind. In addition, the first layer is covered by aluminum for support and thermal conductivity.

The sensors used for the active layer are CMOS sensors, a type of Monolithic Active Pixel Sensors (MAPS). They consist of 640 x 640 pixels with a pixel pitch of 30μm. Four of these sensors combined make up an active layer in the prototype. Repeating this 24 times, we end up with roughly 39 million pixels that we must read out for each event. It should be noted that not all sensors used in the prototype are of the same type. There are differences in resistivity, thickness and quality. This should not be a problem, but we should keep these facts in mind.

Furthermore, in order to keep the shower size as small as possible tungsten (W) has been chosen for the absorber layer. This high Z material ensures a small Molière radius, which allows our sampling calorimeter to remain relatively small. Per layer, we have 3.3 mm of tungsten followed by a MAPS which adds up to an effective thickness of 0.97 X_0.

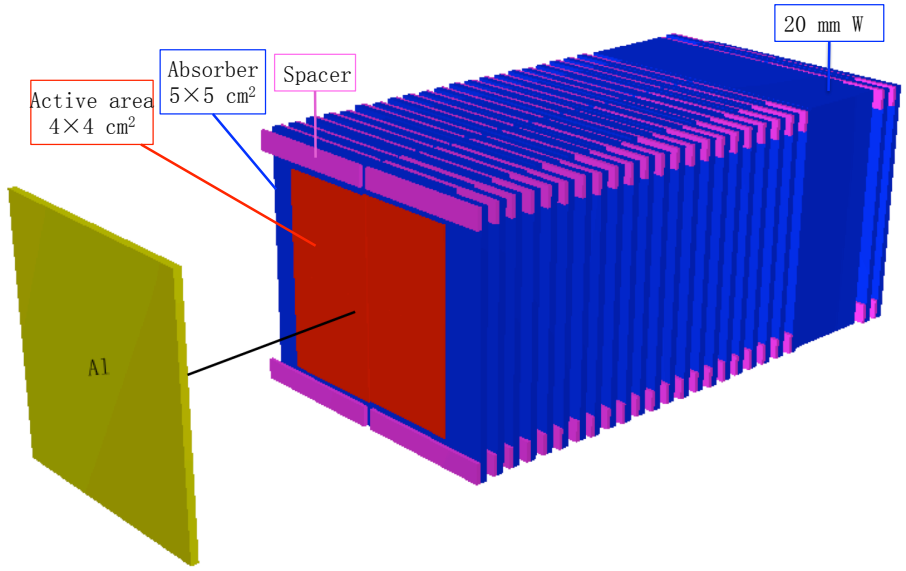


Figure 5 – [15] A schematic overview of the FoCal prototype. The aluminum in front is attached to the first layer for the actual prototype.

2.3 Saturation

One important aspect is that of saturation. In digital calorimetry we use pixel sensors to measure the energy of an incoming particle and to track the resulting cascade of particles. However, we are forced to use a binary pixel read-out to keep the amount of data within reasonable limits. This brings a problem when a large part of the available pixels is indeed fired. For example, when two particles in a cascade deposit their energy in a single pixel, this pixel is fired and returns a 1 in the read-out. However, the information that there we in fact two particles contributing to the deposited energy is lost.

This loss of information is not that important at low energies, because at those energies there are not that many particles and therefore not as much overlap in energy deposition. When we increase the energy however, the number of particles in the shower increases and thus the chance of overlap increases. Therefore, we should check at what point the saturation starts to play a significant role and try to find a way to correct or it. In Section 3.3.2 we describe our toy model which focuses on saturation effects in a single pixel sensor.

3 Methods

This section focuses on the data we use, including the data obtained by the FoCal prototype as well as the data obtained with our toy model. Furthermore, we explain the analysis macros used in our initial research and end with a thorough description of all steps in the toy model.

3.1 Description of FoCal data

Data have been obtained with the FoCal prototype in test beams at multiple energies. Measurements in a test beam have been performed at DESY, PS and SPS. The full range of measured energies can be found in Table 1. The focus of these test beams was on measuring electromagnetic showers, i.e. beams containing either electrons or positrons. However, due to the way the test beam is produced at CERN, a small fraction of pions is present. This presents us with the convenient possibility to also study those.

In principle, the data that we obtain from the FoCal has a relatively simple structure. Due to the fact that it is a digital calorimeter, we receive information on all working pixels present in the prototype. These pixels can either be fired (1) or not (0), which is precisely the information we obtain. For each of the 24 layers we read out the four sensors, providing us with a 2D grid of pixels. Combining the layers then allows us to visualize the full data as a 3D pixel map containing 1's

Table 1 – [15] An overview of data samples obtained from test beams at SPS, PS, DESY and Utrecht.

Time	Site	Particle type	Energy (GeV)
Feb 2014	DESY T22	e^+	2, 3, 4, 5.4
Sep 2014	CERN PS T9	e^-, $\pi^\pm$, p	2, 3, 4, 5, 6, 8, 10
Nov 2014	CERN SPS T8	e^+, $\pi^\pm$	30, 50, 100
Nov 2014	CERN SPS T8	e^-, $\pi^\pm$	244
2013-2016	Utrecht lab	cosmic	-

and 0's. The fired pixels then correspond to hits due to a particle passing through an active part of the prototype. Figure 6 shows fully processed and cleaned data in all 24 layers of the prototype for multiple events at 244 GeV measured at the SPS.

A striking feature of Figure 6 is the white parts that appear in many of the layers. These parts correspond to the pixels that do not work, the dead pixels. We see several lines of dead pixels and numerous dead rectangle areas. A few layers even have complete sensors that are damaged and thus unable to return pixel information. In some layers, such as layer 21, this plays a significant role as three-quarters of the layer is dead. Other layers have only small parts that are dead and require less correction work. How we perform the correction for these dead areas is described later on in Section 3.3.1

On top of gathering data with the FoCal prototype, we also use a Monte Carlo simulation where we extract equivalent information. In the simulation we know the exact location and energy deposition of particles in the shower. However, a pixel grid is manually added to be able to mimic the data obtained by the FoCal. Thus, the energy deposition of a particle is assigned to a specific pixel in the grid. Afterwards, we can set a threshold value for this energy deposition to distinguish between fired pixels (hits) and non-fired pixels. Combining the FoCal data with the simulation data then allows us to further develop our analysis methods.

3.2 Data processing steps

The FoCal prototype obviously does not hand us processed data. We collect raw data and through a series of assignments we obtain data similar to that in Figure 6. We describe the main steps in this data processing process below.

3.2.1 Demultiplexing

The raw data collected by the prototype is recorded by FPGA boxes and consists of two different data types: the pixel stream and the trigger stream. The pixel stream includes data obtained from the pixel sensors in the prototype and the trigger stream includes data from all available trigger channels, both internal and external. By combining these two data types, we are able to generate frames that show all pixel values for a given event.

The first step in this process is called the demultiplexing step. The raw data is not stored directly onto a hard disk, but is temporarily saved in the buffers of our FPGA boxes. It arrives at these FPGA's via several independent serial lines. The FPGA's introduce a level of multiplexing, meaning that data from several serial inputs are combined to a single output. Throughout this data gathering stage several layers of multiplexing are applied to the data to allow for an efficient read-out. As a result, this has to be reversed afterwards.

We do not go into the full details of the demultiplexing steps here, as it is not relevant for the rest of the thesis. However, previous research has resulted in lookup tables including all levels of multiplexing [10]. These tables contain the pattern in which the data is written to memory and by reversing that pattern we can reconstruct the structure of data from a single event.

3.2.2 Trigger stream alignment

Subsequently, we start the next step called trigger stream alignment. Each of the four FPGA boxes has a certain clock value used to synchronize data streams. However, these clock values are not identical and have to matched for all FPGA boxes. Matching these clock values allows us to have an overview of the pixel values in the prototype at a specific point in time.

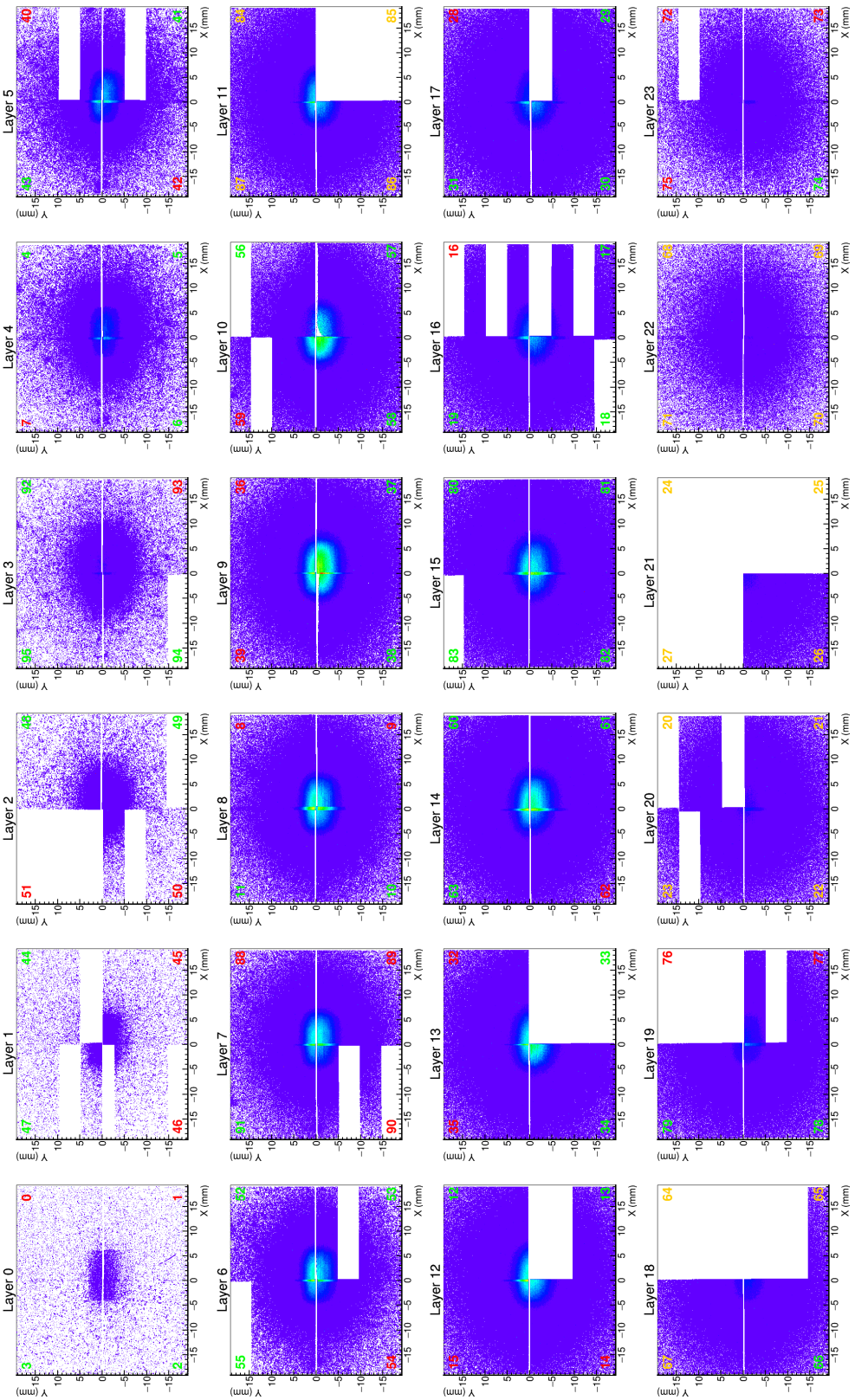


Figure 6 – [13] Uncalibrated distribution of hits in 2D in all 24 layers of the FoCal prototype for cleaned-up SPS data at an energy of 244 GeV.

The issue is that the clocks of different FPGA boxes do not necessarily start at the same time and run at different speeds. Therefore, it is slightly more complicated to match corresponding trigger streams. In order to match those, we must use the trigger streams that are also recorded. To prevent mismatching of trigger streams, we use only triggers that are generated by the PMTs connected to the trigger system [10]. Using only these triggers, we minimize the effect of differences in clock speeds and clock values.

3.2.3 Frame selection

Finally, we create a format in which we store the entire state of the detector at a certain point in time. This format is called a frame and contains all information belonging to an event. Adding up all 24 layers containing 4 pixel sensors, we end up with information on 96 sensors. The pixel sensors that we use require some integration time in order to collect all charge generated by an event. A frame thus comprises a period in time, which for our prototype has a maximum length of 639 μs [10]. Such a frame includes pixel information, trigger information and diagnostic information.

To keep the data within limits and to ensure that we only store interesting events, we only store signals when the external trigger system has fired. When that happens, that trigger information is stored along with the positions of fired pixels. Diagnostic information is added to store information on for instance which sensors performed correctly or whether the demultiplexing phase was successfully completed. Then, the frames can be used for further offline analysis

3.3 Analysis

As we mentioned before, the frames obtained after processing the data are ready to be analyzed. Here, we explain the methods we used to obtain results. This includes the code we wrote and the programs used to process that. First, we give a brief description of the macros used to read in the frames and analyze them. Afterwards, we discuss our toy model in which we attempt to mimic a single layer in the detector to further study saturation effects.

3.3.1 Initial data analysis

In the following part we discuss the macros that we used to analyze the data frames collected by the FoCal prototype. The macros consist of five parts:

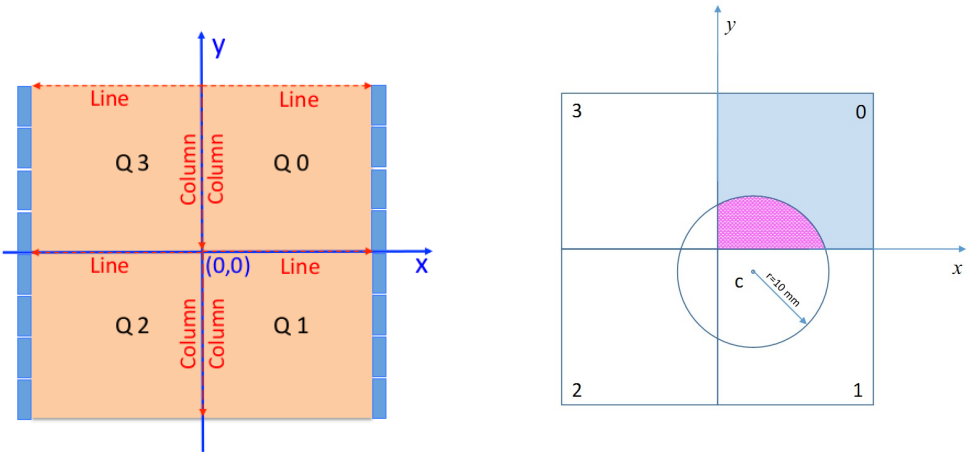
1. Reading in data frames
2. Defining histograms to fill in the body of the macro
3. Constructing observables from the pixel data
4. Filling appropriate histograms with newly acquired information
5. Saving histograms in a different data format

Naturally, the first step we take in our data analysis is reading in the actual data. To be clear, that is the data obtained with test beams shown in Table 1. These data frames were stored in multiple root-files that we could easily loop over. An extensive collection of classes was already written to aid us in extracting the required information from these frames. For example, creating an instance of the *LayerDensity* class with a data frame as input would return an object containing information on all layers of the prototype for that frame. We do not discuss these classes here, but they did help us a great deal in our analysis.

So, looping over the data frames and creating instances of the analysis classes for each frame allowed us to relatively easily access the pixel information in these frames. However, the interesting part was not getting the pixel information. Using the pixels, we had to come up with alternative observables that would give us new insights into the data. Examples are the amount of hits in a single layer, or combining all layers to obtain the total number of hits in the entire detector at a certain energy. Finding the right observable is crucial in determining the energy and type of the particle as precise as possible. Therefore, before constructing these additional observables, we defined numerous histograms in which we could display them. Alternating the axes, ranges and energies then allows us to inspect trends in the different variables.

As described in Section 2, in a digital calorimeter the energy of an incoming particle is proportional to the total number of pixels in the detector. Ideally, we could read out the number of fired pixels and use that as the total number of hits for each event. However, the FoCal prototype is far from ideal. As shown in Figure 6, the detector contains many dead sensors and dead pixels. In our analysis we must correct for this missing data in order to obtain a reasonable estimate for the total number of hits, and thus the energy of the incoming particle.

To achieve that, we introduce the concept of hit density. As mentioned, a single layer in the prototype consists of four quadrants. A schematic of that is shown in Figure 7a. Unfortunately, due to the presence of dead sensors and dead pixels, not all of the layers in the FoCal prototype perform correctly. As an example, we take the case that quadrant 0 (Q0) is a dead sensor and thus does not give us any output. A particle comes in at a random position and creates a cascade of particles around a central position. A visualization of that scenario is shown in Figure 7b, where the central position of the shower is given by c.



(a) **(b)**

Figure 7 – [13] Schematic overview of the composition of sensors in a single layer of the FoCal prototype.

In order to solve this issue we use the pixel information available in Q1, Q2 and Q3 and extrapolate that to Q0. We do that as follows. First, we divide the layer into rings around the central shower position. In this example, the inner rings would then be solely in Q1, but the more outer rings will have parts in Q0 and eventually Q2 and Q3. Subsequently, we calculate for each of the rings how many pixels actually work, and how many hits are present in that ring. These two properties combined with the known area of a single pixel gives us a hit density in hits/mm^2. If we define the distance from the central shower position $c = (\mathrm{x}_c,\mathrm{y}_c)$ as

$$r = \sqrt{(x - x_c)^2 + (y - y_c)^2}, \tag{5}$$

then we calculate the hit density in quadrant q as [15]:

$$\nu_q(r) = \frac{N^q_{hit}(r) - \Sigma p_i}{N^q_{pixel}(r) \cdot A_{pixel}} \tag{6}$$

where $N^q_{hit}(r)$ is the number of hits in a ring at distance r, N^q_{pixel} is the number of working pixels in that ring and Σp_i is total measured noise contribution obtained by summing the noise probability p_i over all working pixels. The area corresponding to a single pixel in a sensor is in our case fixed at $A_{pixel} = (30\mu\mathrm{m})^2$.

Afterwards, we can simply multiply the hit density in a certain ring with the area of that ring to obtain the number of hits in that ring, including quadrant 0. In this way, we circumvent the issue of dead pixels by extrapolating the information in the other three quadrants. By adding up all rings around the central position we then obtain the number of hits in the entire layer.

These and other properties extracted from the pixel data are subsequently filled in the appropriate histograms and can be analyzed in more detail. Finally, we write all histograms created in the macro to a different root-file for later reference.

Sadly, these macros were lost in the quark-cluster crash, so we were forced to conduct our research differently. In Section 4.1 we describe the results obtained up until that point, where we ended with the issue of saturation.

3.3.2 Toy model

In order to study the concept of saturation more thoroughly we decided to develop a toy model. Unlike the simulation performed with Geant where we simulate all 24 layers of the detector, we can constrain this model to a single pixel layer. Naturally, different layers of the FoCal deal with a variable number of particles. However, for our toy model we decided to take the number of particles as an input parameter to be able to switch between these layers. We go into the input parameters of our model in more detail later on. We wrote our model within the ROOT framework, combining C++-style code with ROOT libraries for efficient and clear code.

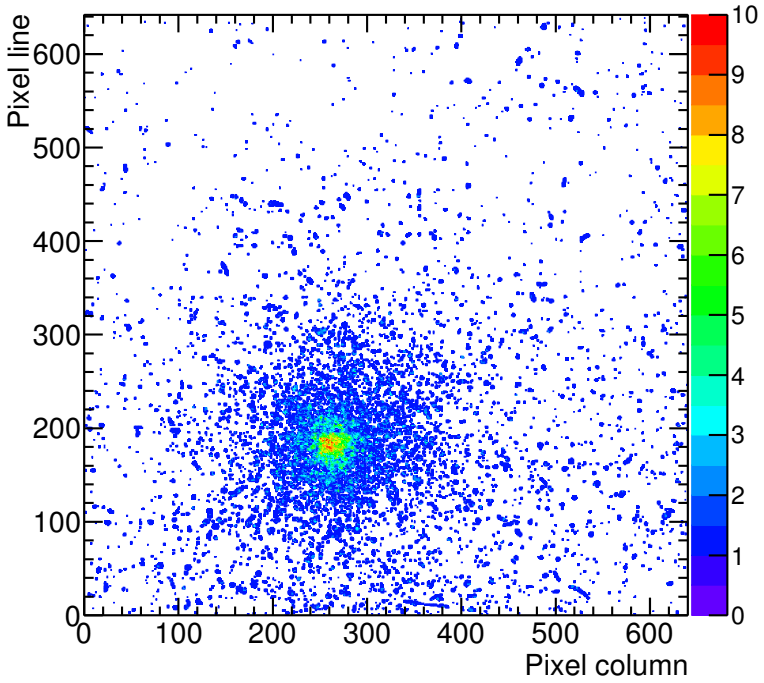


Figure 8 – [13] Plots showing the pixel map of a single chip in the prototype for 244 GeV SPS data.

For inspiration on how the output of the model should look like we used Figures 6 and 8. Especially the latter, which shows pixel data from a single chip for SPS data at 244 GeV, lead us towards a clear view of the output. If we could recreate a similar map, then we would be able to continue our study of saturation in the detector.

Eventually, we were able to achieve that. A visualization of an event in the model is shown in Figure 9. It shows an event where we simulate 10000 particles on a square grid with a size of 100 pixels. In this initial simulation phase, the cluster size belonging to a single particle is uniformly distributed between zero and four. The figure shows the number of particles that contributed to a hit in each of the 10000 pixels from which our pixel map is constructed.

In the remaining part of this section we cover all steps in the toy model. The complete model *ToyModel.C* is added with the thesis. Here, we point out the important parts that possibly require explanation. In principle, our model consists of the following four steps:

1. Set initial parameters

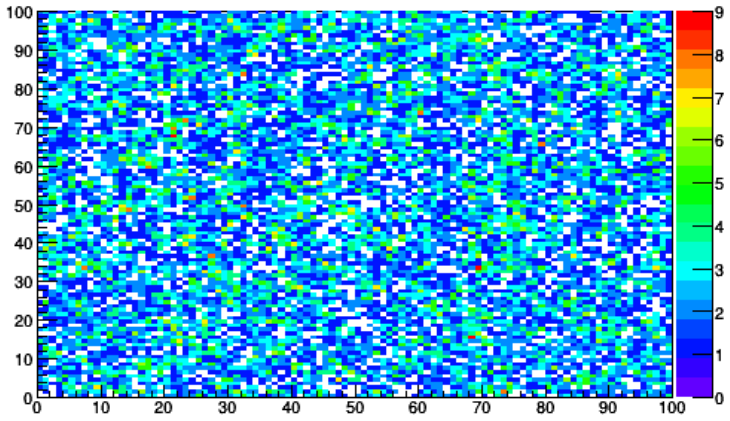


Figure 9 – Pixel map of size 100x100 created by the toy model. The 10000 incoming particles have a uniformly distributed position. The cluster sizes are uniformly distributed between 0 and 5.

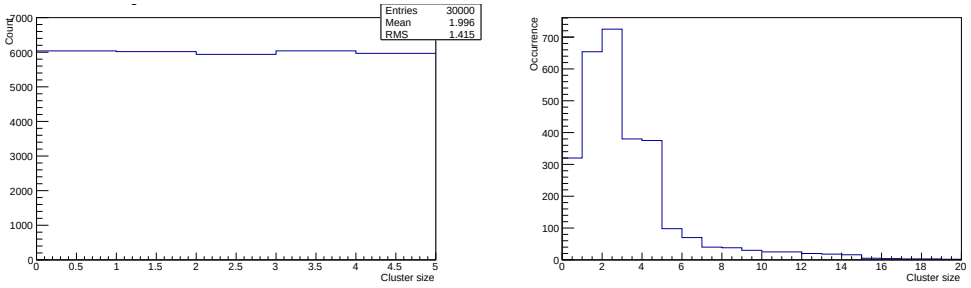
2. Define the grid
3. Random particle loop
4. Grid evaluation

As with any interesting model, we want to be able to specify certain input parameters. To that end, we chose four parameters that are mandatory when running the model. First of all, we specify *NParticles* which defines the total number of particles that we simulate on our grid. This parameter basically allows us to simulate events of different energies, as the number of particles is proportional to the energy. Moreover, we require a value for *gridSize* which determines the size of our rectangular pixel map. For the main part of our research we put the value of this parameter at 100, signifying a size of 100x100 pixels. Then we have two somewhat less important input parameters, namely *maxClusterSize* and *pixelSize* that, respectively, determine the maximum cluster size belonging to a single particle and the actual size of a single pixel in our pixel map. The latter was built in to allow for a comparison with other models, but due to time considerations was not used much. For simplicity, we put the value of *pixelSize* at 1. On top of that, we initialize instances of several objects that allow us to write efficient code. This includes a number of variables with clear concise names as well as a random generator that we use, for example, to generate random positions on the grid.

Subsequently, we define that grid as a two-dimensional array *Grid* with indices (i, j), where $0 \leq i \leq gridSize - 1$ and $0 \leq j \leq gridSize - 1$. A combination (i, j) then refers to a specific position of a pixel on the grid. Before we start generating particles, we initialize all values of *Grid* as 0. A value of 0 signifies that at a certain position on the grid there are no hits generated by particles.

Then, we start generating particles that we 'shoot' at our pixel map. That process includes two steps. Firstly, we generate a random position on the grid by drawing a random i and j from a certain distribution. These distributions are not fixed and one can choose between different function. Examples include uniform and Gaussian distributions, but also distributions with a related physical background such as Equation 7 which we describe in Section 4.1.2. Second, we know that a single particle does not fire a single pixel, but raises a cluster of pixels above the threshold. Therefore, we implement multiple cluster size distributions to aid us in our research. Simple cluster size distributions encompass uniform and fixed distributions. However, a distribution based on actual data from the FoCal prototype is shown in Figure 10b.

Finally, after generating an amount of particles equal to $NParticles$, we end up with a filled pixel map such as the one shown in Figure 9 when the incoming positions of the particles are uniformly distributed. We then evaluate the results by looping over the two-dimensional grid and extracting certain properties. We can count the number of hits, i.e. counting the number of pixels with a value ≥ 1. In combination with the number of generated particles that gives us an idea of



(a) **(b)**

Figure 10 – Two cluster size distributions. In (a) we see a uniform distribution of clusters whereas (b) [15] shows a distribution based on FoCal prototype data.

the amount of information lost due to saturation. Additional observables can be extracted from the pixel map, which we discuss in further detail in Section 4.2.

Our toy model is a statistical model, using many random numbers and distributions. Therefore, in order to reduce the errors in our results, we often repeat this process several times to increase our statistics.

4 Results

So far, the analysis tasks were in preparation of achieving our desired research goals. Therefore, we chose to display the obtained plots in the previous section rather than here. However, perhaps we could count the plots from Section 4.1.4 and onwards as actual results that aid in achieving those goals. This decision is surely open for debate. Should this be under the Analysis section or should this be a separate section?

4.1 Initial data analysis

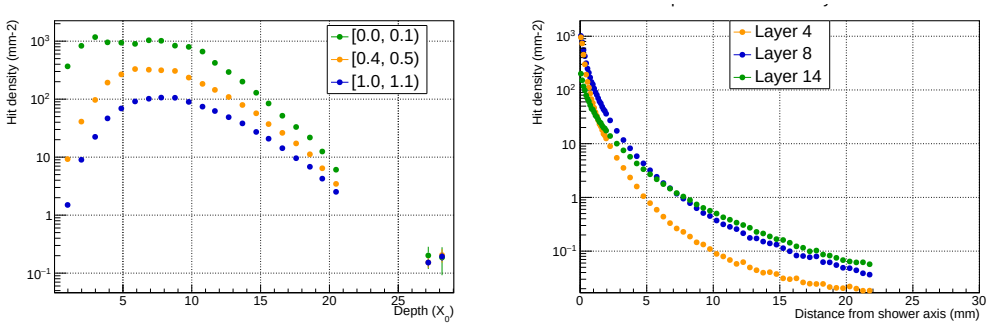
In this section we cover the different analysis tasks performed on the FoCal data and the corresponding Monte Carlo simulation. In the simulation we can set the exact energy and type of the incoming particles, allowing us to compare with the actual data. Naturally, we choose energy values corresponding to those in Table 1 for that purpose.

4.1.1 Longitudinal and lateral profiles

From the data we can construct both longitudinal and lateral profiles. These profiles both display the hit density at specific points in the detector. For longitudinal profiles we obtain the hit density obtained in each layer of the FoCal. This allows us to examine for instance the layer in which the shower deposits most of its energy, i.e. the shower maximum. Moreover, we can examine the longitudinal profile in a set of rings around the entry point of the incoming particle. An example of such a longitudinal profile for the 100 GeV data from SPS is shown in Figure 11a. We see that shape of the hit density profile varies for different rings. Interestingly, we see the apparent effect of saturation for the innermost ring. For that ring the hit density appears to saturate at a value of 1000 mm^{-2}, whereas the ideal value equals 1111 mm^{-2}. When we examine rings further outward, we see the hit density decrease. The obvious gap in the profile at high X_0 is due to the large block of tungsten in the prototype.

Lateral profiles display the hit density in rings around the central axis of the shower. In principle, these are used to examine the shower in a layer individually. The lateral profiles for three layers in the prototype are displayed in Figure 11b. We see that in the inner most rings the hit density is highest. In order to obtain the total number of hits in a single layer, one can simply calculate the integral of the lateral profile corresponding to that layer.

Essentially, all analysis tasks we perform start from either a longitudinal or lateral profile for a specific case. Subsequently, we extract the required information from these profiles for the task at hand. For example, we study the total number of hits in the FoCal for an event by taking the



(a) (b)

Figure 11 – Longitudinal and lateral hit density profiles for SPS data at 100 GeV. We show the hit density for (a) the longitudinal profile of three rings around the central shower position and (b) the lateral profile of three layers in the detector.

integral of the longitudinal profile for that specific event. Furthermore, we examine the differences between simulation results and the obtained data.

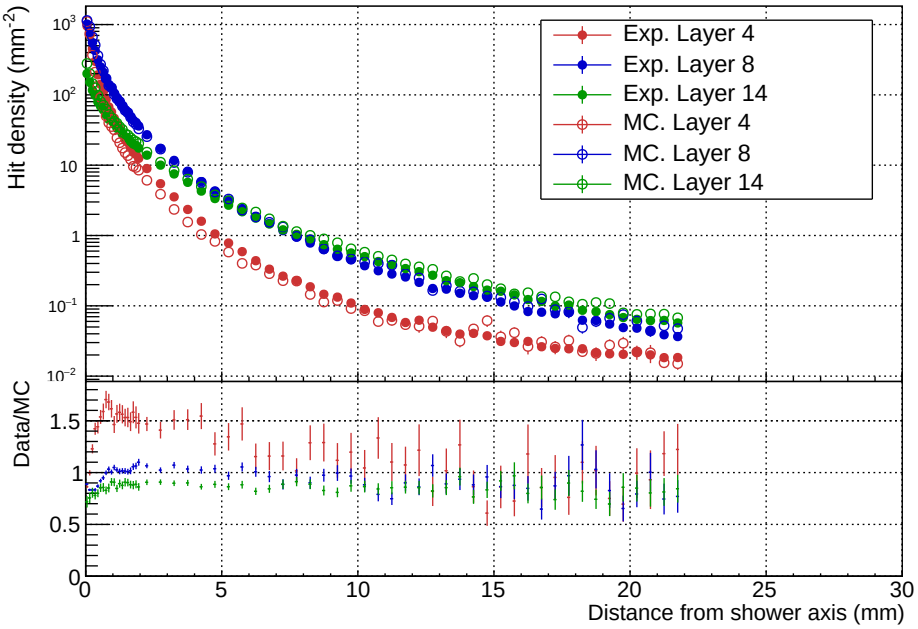


Figure 12 – Comparison of lateral hit density profiles between FoCal data and Monte Carlo simulation for three layers in the detector at 100 GeV. The open points represent the MC simulation of positrons with 200 simulated events. The filled points show the FoCal data from SPS. In the panel we show the ratio between data and MC for the corresponding layers.

Figure 12 shows a comparison between lateral profiles of three layers for both simulation as well as data. The panel shows the ratio between values obtained by data and simulation. We observe the most prominent difference in layer 4, where the ratio reaches values of around 1.5 within the inner rings. Towards higher distances from the shower this difference slowly decreases until it settles around 1 with fairly large uncertainties. For layers 8 and 14 the MC yields higher hit densities than the data for the innermost rings. In layer 8 the data and MC are fairly equal for the rest of the rings. For layer 14 on the other hand, the Monte Carlo yields higher hit densities throughout all rings in the layer.

4.1.2 Profile fits

In addition to manually extracting information from the profiles described above, we want to be able to fit a model in which the parameters of that model capture the crucial aspects of the profile. We know that for instance a pion and a positron induce a different shower in a calorimeter. More specifically, the shape and the starting point differ due to differences in hadronic and electromagnetic behavior. Ideally, these properties then show up in the profiles we discussed, allowing us to distinguish between the two by fitting a specific model.

Several fits have already been performed on lateral hit density profiles for electromagnetic showers, shown in Figure 13. The most suitable fit is given by

$$g(r) = p_0 \left(1 + \frac{r}{p_1 \cdot p_2}\right)^{-p_1} \tag{7}$$

where p_0, p_1 and p_2 are fit parameters and r the distance from the shower axis. We omit the rest of the fit functions as they are not of interest to the purpose of this project.

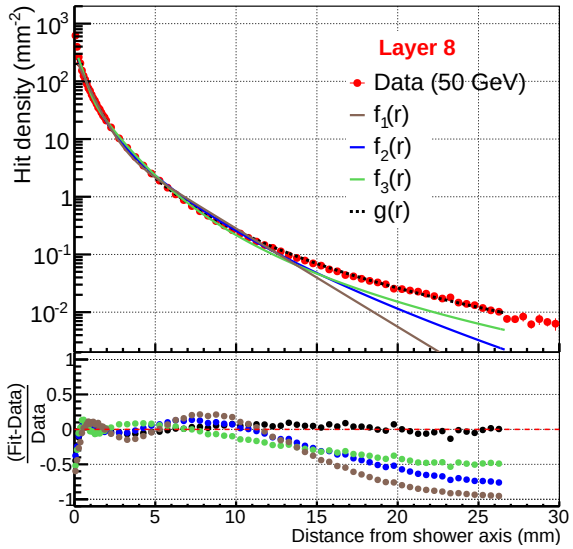


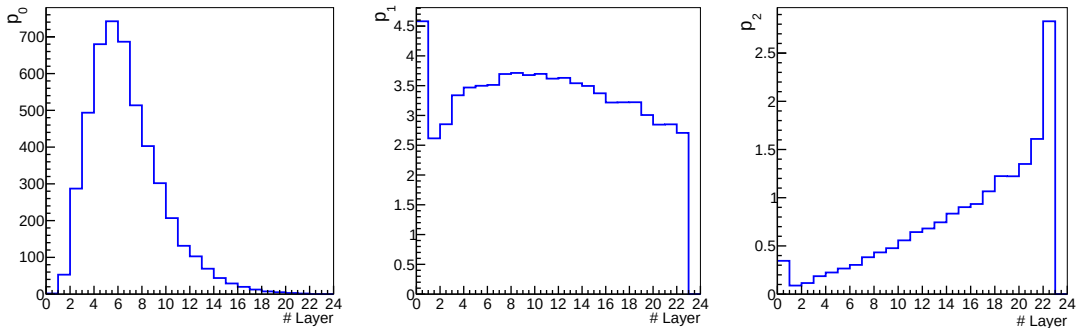
Figure 13 – [15] A comparison of lateral hit density profiles for 50 GeV positrons in layer 8 and four fit functions. The panel shows the deviation of the fit with respect to the lateral profile data points.

However, these are fits of two-dimensional profiles. Our goal is to construct a 3D model, as that allows us to study the full shape of the cascade. In order to achieve that, we should combine the longitudinal and lateral profiles. Thus, these profiles should have a generic shape and vary only depending on the parameters of a model. The values of the fit parameters used in $g(r)$ depend on the layer in which we fit the lateral profile. This dependence is shown in Figure 14. We see that these plots have a specific shape which might prove useful when we start constructing our model.

As we see in Figure 11a, there are still some peculiarities in the profiles that we can improve upon before we extend our study to 3 dimensions. The issue of saturation, visible in the longitudinal profiles at higher energies, is something that would certainly influence the quality of our fits. The rest of this section is devoted to describing our efforts in resolving this issue.

4.1.3 Calibration

The sensors used in the FoCal prototype are all of different quality. As described in Section 2, there are three different types of sensors present of varying thickness and specific resistivity. Additionally, the production process is not perfect and can lead to sensors that have a high number of dead or unstable pixels. To adjust for this difference in performance and quality of the sensors, we apply a calibration. Furthermore, we use a correction method to solve the issue of dead areas or dead sensors in the prototype.



(a) (b) (c)

Figure 14 – [15] Fit parameter values of (a) p_0, (b) p_1 and (c) p_2 used in Equation 7 obtained for 50 GeV positrons in all layers of the detector.

Fortunately, these calibration factors have already been calculated and the correction method has been developed [15, 13]. The calibration factors are obtained using 50 GeV positron data. Figure 15 displays the plot that shows the average longitudinal profile before and after calibration. We see that after calibration the jumps in the profile vanish and we are left with a smooth curve.

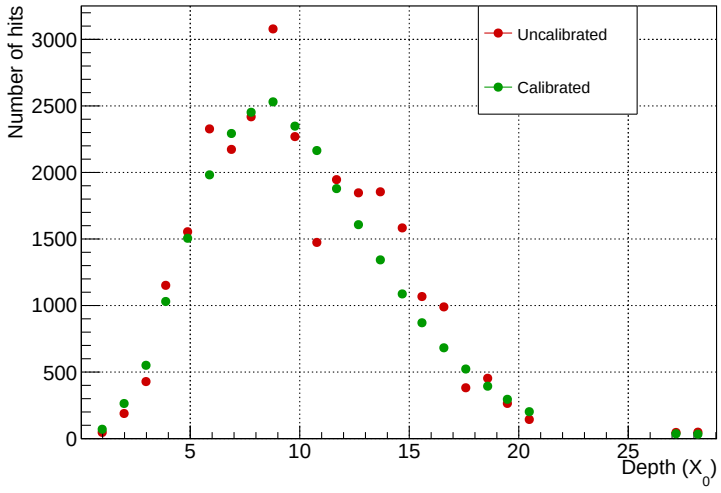


Figure 15 – Average calibrated (green) and uncalibrated (red) longitudinal profiles for 100 GeV SPS data.

4.1.4 Saturation

However, implementing the calibration does not solve the issue of saturation that we see in our longitudinal profiles in Figure 11a. We want to fix this issue so that we can construct a model that works at different energies. To this end, we perform Monte Carlo simulations to understand the saturation process. In the MC, we have access to other properties than in FoCal data such as the deposited energy E_{dep} in each pixel. Unlike the number of hits, this property should not suffer from saturation effects because we do not introduce a cut-off value. The idea is to use the ratio between the deposited energy and the number of hits to generate a correction factor that solves our saturation problem.

Likewise, we plan on extracting alternative observables from the simulation that might be related to the energy of the incoming particle. Then, we compare which of the observables is most directly related to the energy and use that one for the correction factor. However, for now we stick to the deposited energy.

4.1.5 Charge diffusion

Charge diffusion comprises the spatial smearing of electron-hole pairs that are created in the silicon chips. These pairs are collected by diodes due to their natural diffusion, but this inherently leads to charges being collected by neighboring pixels. This effect is shown schematically in Figure 16.

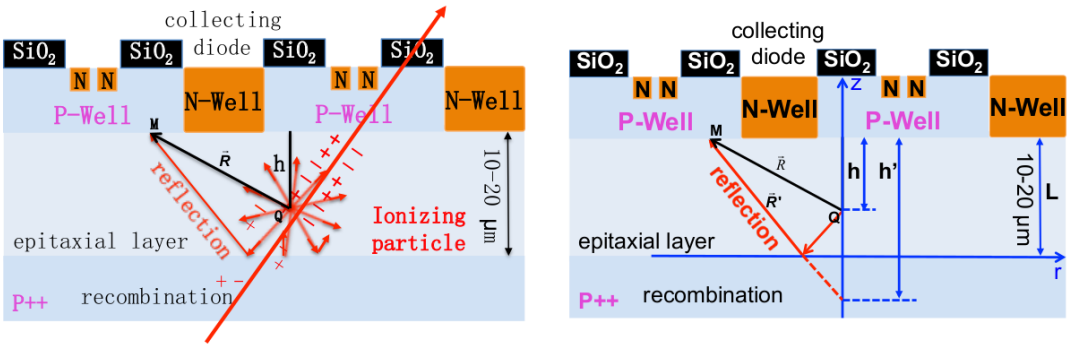


Figure 16 – [13] Schematic of the working principle behind charge diffusion.

We study the effect of charge diffusion in the Monte Carlo simulation by looking at the total number of hits obtained per event. We examine the differences by performing the calculation both with and without charge diffusion. It should be noted that within the code framework that we use, applying charge diffusion introduces noise pixels whereas the case without charge diffusion does not include noise. Figure 17 shows that we obtain a significantly larger number of hits per event due to charge diffusion. A rough estimate would be that charge diffusion yields around four times as many hits as no charge diffusion.

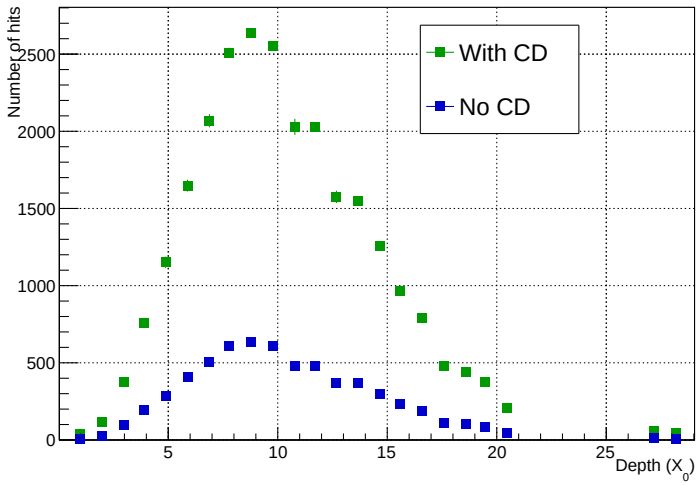


Figure 17 – Average longitudinal profiles of 100 GeV MC simulations with 200 simulated events. We show both the case with (green) and without (blue) charge diffusion applied.

Intuitively, this does not lead to saturation effects when there are few hits in the detector, i.e. when the energy of the incoming particle is relatively small. However, when the energy increases, the number of hits increases and then the effect of charge diffusion could cause loss of information due to saturation. To check our intuition we constructed a toy model that we use to study saturation, see Section 3.3.2.

4.1.6 Linearity

To see where saturation starts to play an important role we examine the total number of hits and total deposited charge Q_{dep} for multiple energies. In the ideal case without saturation, these properties should both depend linearly on the energy. However, this is not trivial and thus we check the energy dependence of the two properties. Figure 18 shows the total number of hits, specifically the integrated longitudinal profile, per event as a function of energy.

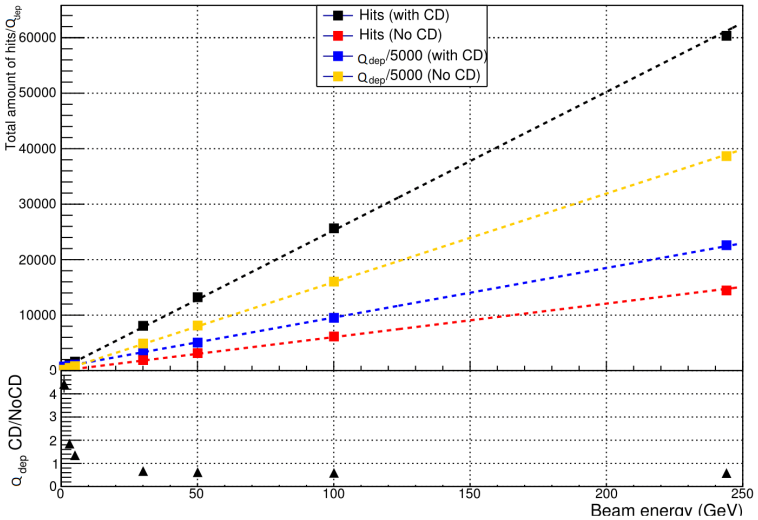


Figure 18 – Linearity of the total number of hits and the total deposited charge Q_{dep} in all layers of the detector as a function of energy. The data is obtained using MC simulations with 200 simulated events for all energies. The dashed lines represent first order polynomial fit functions *pol1*, i.e. linear fit functions. The panel shows the ratio of Q_{dep} with and without charge diffusion applied.

We see that the deposited charge fits tend to describe the data points better than the fits for the number of hits. More specifically, the 244 GeV data points for the total number of hits with and without charge diffusion are slightly below the fit, indicating some form of saturation at that energy. However, this does not appear to be the case for the total deposited charge.

Interestingly, we observe that the total amount of deposited charge Q_{dep} is higher without charge diffusion compared to the case with charge diffusion applied. One would assume that the total amount of charge would not vary between these two cases, but apparently it does. Presumably, this is caused by the inherent isotropic nature of diffusion. The charges are diffused isotropically and therefore do not all reach diodes that collect them. We do not see this in the case of the total number of hits. Namely, in that case we apply charge diffusion which could be enough to push neighbouring diodes over the threshold of firing. As we do not use a threshold for the deposited charge, we thus observe the amount of charge that is lost in the process.

The current ratio panel shows us that the difference between the deposited charge with and without charge diffusion remains constant at high energies. The difference at low energy can be explained by the fact that in the case of applied charge diffusion we inherently add noise as well, in contrast to the case without charge diffusion.

4.2 Toy model results

In the following section we show all results acquired with the toy model.

4.2.1 Pixel map displays

First of all, we display a number of pixel maps that we examined. The simplest and most intuitive situation is where we use a uniform distribution of incoming particles. If we shoot 500 particles at a pixel map with two sides of 100 pixels, this gives us pixel maps similar to the one in Figure 19. We

start with a uniform cluster size distribution such as the one shown in Figure 10a. Subsequently, we increase the amount of particles while keeping the rest of the settings the same. That way we obtain maps shown in Figure 20. One can already imagine more and more clusters starting to overlap if we keep increasing the number of particles.

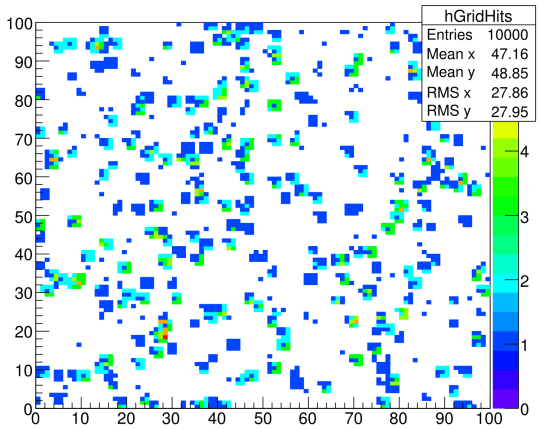
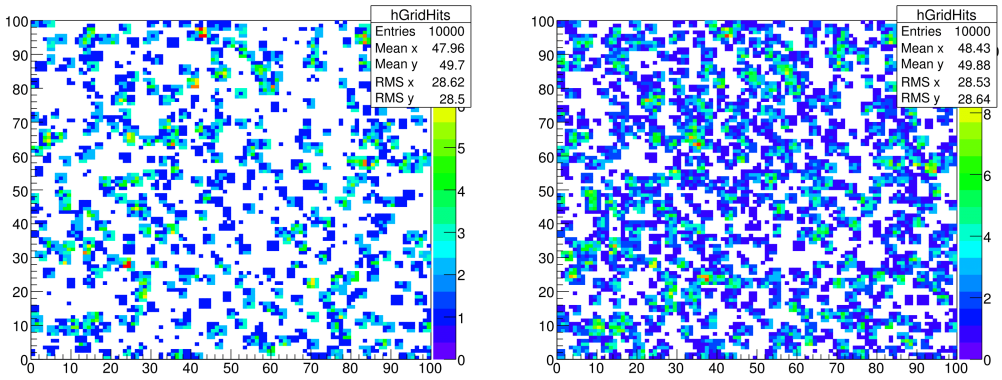


Figure 19 – Pixel map with size 100x100 obtained with the toy model by running a simulation with 500 particles that have a uniform cluster size between 1 and 4. The incoming positions of the particles are randomly, uniformly distributed.

We are able to vary the distribution of positions of the incoming particles. Differently, we use a Gaussian distribution where we set the center and width of the distribution. The pixel map shown in Figure 21 has the center of the distribution at 50 and has a width of 10 pixels. Obviously, we obtain a great deal more overlapping pixels when we focus the modeled shower particles more at the center of the pixel map.



(a) **(b)**

Figure 20 – Pixel maps with size 100x100 obtained with the toy model by running a simulation with (a) 1000 particles and (b) 2000 particles that have a uniform cluster size between 1 and 4. The incoming positions of the particles are randomly, uniformly distributed.

These are a few examples of the pixel maps we create with the model. Of course, many more distributions can be implemented and studied. However, due to time considerations we stick to these two types of input distributions for this research. We do play with the cluster size distribution, which will be examined more closely at the end of this section.

4.2.2 Hits vs particles plots

As described in Section 3.3.2, the main purpose of the toy model was to study the effect of saturation in a single layer in a digital calorimeter. Therefore, it is interesting to examine the number of hits we collect for a specific number of particles that we shoot in our simulation.

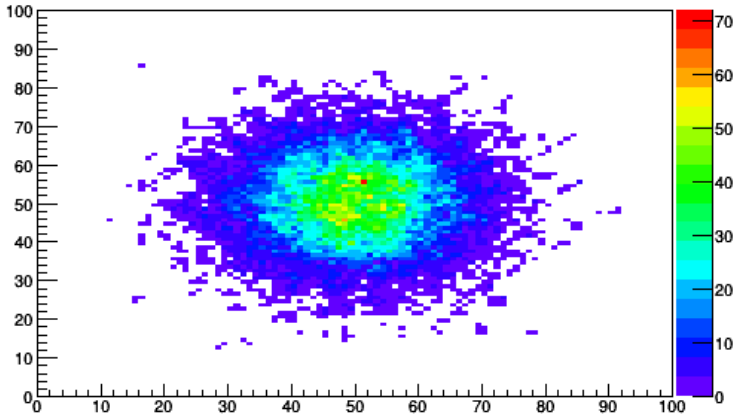


Figure 21 – Pixel map with size 100x100 obtained with the toy model by running a simulation with 10000 particles that have a uniform cluster size between 1 and 4. The incoming positions of the particles are distributed Gaussian-like with its center at 50 and a width of 10.

We start with a plot that shows the full range of our model, almost reaching the maximum asymptotic value of 10000 (100*100) hits in the pixel map. This plot is shown in Figure 22. We see that the relation between these two observables is far from linear at this scale. This is an indication that we are dealing with some form of saturation in our model.

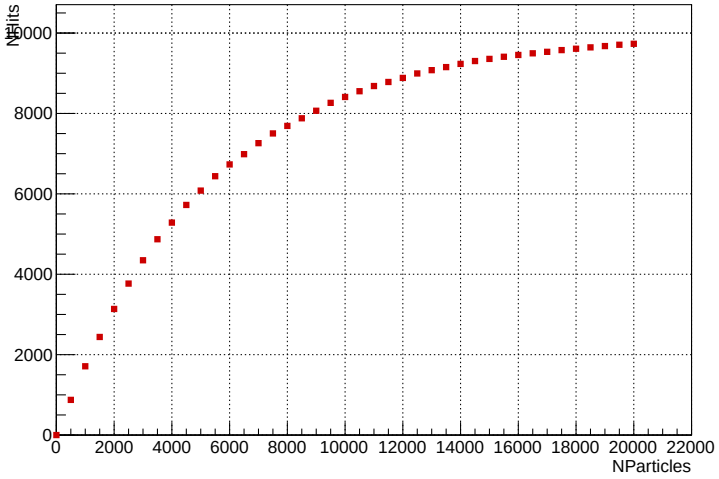


Figure 22 – Hits vs particles obtained by the toy model for a grid of size 100x100 using the cluster size distribution shown in Figure 10b and a uniform distribution of particle positions.

A similar plot of the number of hits versus the number of particles, only now using the Gaussian input distribution, is shown in Figure 23. As expected due to the central position of the Gaussian mean, the asymptotic value is not obtained quite as fast as with the uniform distribution. As shown in Figure 21, we have many overlapping pixels in the center of the grid whereas on the outside there are many pixels that have not been fired.

Both of the previous plots are acquired by running the model once. We observe a greater spread in data points with the Gaussian input distribution. Moreover, the uniform input distribution allows for a more fundamental study of saturation due to the non-localized distribution of particles. If we were to use the Gaussian distribution, we would see saturation right at the start. However, we would like to see at what occupation of the pixel map saturation actually has a significant influence on the result. To that extent, we choose to mainly use uniform distributions as that will benefit our research.

On the other hand, we should keep in mind that by using a uniform distribution we stray

away from acquiring results that can be applied to a realistic situation. We thus focus on the fundamental properties of saturation rather than on a more realistic situation that we obtain with the Gaussian input distribution. In the end, we should also examine these realistic situations to observe the effect of saturation more thoroughly.

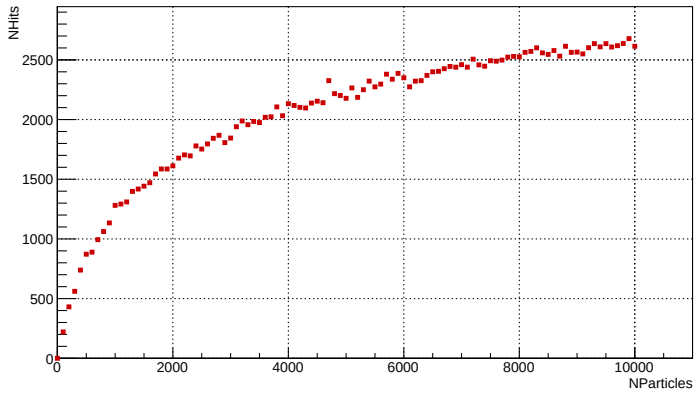


Figure 23 – Hits vs particles obtained by the toy model for a pixel map of size 100x100 using the cluster size distribution shown in Figure 10b and a Gaussian distribution of particle positions.

Along with the distribution of incoming particles there is another distribution that we can vary, namely the cluster size distribution. To determine which distribution shows the optimal properties for conducting our specific study, we examined several plots for each of the distributions. Again, our study is simplified and the choices made here are not aimed at creating a realistic shower profile, but at examining the effect of saturation. Here, we show the two most important distributions that we examined: the cluster distribution shown in Figure 10b and a fixed cluster size distribution with a cluster size of 3.

First we display three plots for the number of hits vs number of particles using the former, i.e. the variable cluster size distribution. These plots are shown in Figure 24. For each of these plots, we repeated the model a specific amount of times depending on the range in the amount of particles. We see the full range of the model, going up to 10000 particle and reaching the maximum number of hits. Then, we zoom in twice to examine the range up to 2000 and the range up to 200 particles

Subsequently, we display three similar plots using a fixed cluster size of 3 in Figure 25. The three ranges look quite similar to the previous plots, apart from the uncertainties being significantly smaller with a fixed cluster size. As there can be no variability in the values drawn from the cluster size distribution, the models tend to give a more reliable output. This is clearly visible in the difference between Figures 24c and 25c, which inclines us to use the latter in our study of the saturation effect. The uncertainties are more prominent at low particle values and thus we want to minimize those as much as possible. That allows us to study the starting point of saturation more thoroughly.

4.2.3 Linear fits on the number of hits

To further analyze these results we decided to perform several linear fits on the particle-hit plots. These fits have the form $p_0 + p_1 \cdot x$, where p_0 and p_1 are the two linear fit parameters. We impose p_0 to be equal to zero, which is a reasonable assumption as there can be no hits when there are no particles. Ideally, p_1 would give the average cluster size value in our model.

One such fit is shown in Figure 26a, where we use the cluster size distribution from Figure 10b and fit our parameters on the range between 0 and 1000. A large part of the data points obviously does not follow the linear fit. Hence, we zoom in on the part that does. In Figure 26b we narrow down the range to particles between 0 and 2000 while at the same time we increase the number of repetitions to 30. Furthermore, we perform the fit on data points between 0 and 200, showing us that the deviation from linear behaviour starts sooner than we could observe in the full-range plot.

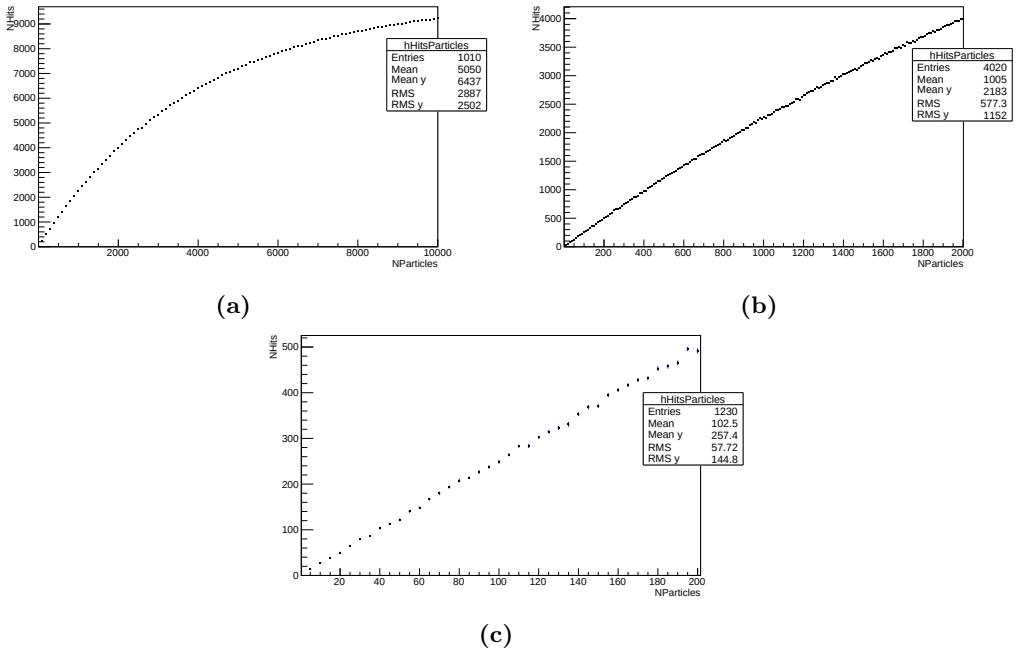


Figure 24 – Hits vs particles obtained by the toy model for a pixel map of size 100x100 using the cluster size distribution shown in Figure 10b and a uniform distribution of particle positions. Each data point represents the average over (a) 10 (b) 20 and (c) 30 repetitions of the model.

Thus, we compare the two distribution cases at even lower particle values. For that we use Figures 27a and 27b which both have a range between 0 and 100. At those values, the linear behaviour starts to become eminent throughout the entire plot. For the two plots we fit the linear parameters only for the values up until a particle number of 20. The variable cluster distribution, shown in Figure 27a, returns a value of $p_1 = 2.32$ that corresponds to the average cluster size. Even with 30 repetitions the number of hits still vary quite a bit. This is due to the fact that the number we acquire for p_1 comes from a cluster size distribution. That distribution introduces some spread in the cluster sizes that we in turn observe in this plot.

Figure 27b shows a similar plot using a fixed cluster size of 3 in our model. This fit looks good at first sight with low uncertainties and relatively linear data points. However, we obtain a fit value of $p_1 = 2.79$. As we mentioned, we imposed a fixed cluster value of 3 on our model for this fit. We can therefore conclude that something is off in the implementation of this cluster size. On the other hand, the fit does appear stable and could still prove valuable.

Thus, we decided to fix this bug rather than disposing the fixed cluster size option all together. After a period of debugging, we found the error to be in our neighbour selection method. For a fixed cluster size of 3 we randomly picked two neighbouring pixels to the pixel of the incoming particle. However, at that point we did not check whether these two neighbours were the same and therefore that was the case for a fraction of all particles. By changing the method to search for neighbours until the cluster size is 3, we actually impose the cluster size on our model. Examples of those results are shown in the next section.

4.2.4 Starting point saturation

Now that we have decided on which distributions to use, namely a uniform distribution of incoming particles and a fixed cluster size of 3, we go into even more detail. We zoom in towards extremely small particle values to check when we observe the first signs of saturation effects. In order to check that more thoroughly, we added a ratio panel to our plots that display the difference between the value of our data points and the value of the linear fit. Specifically, we show

$$ratio = \frac{data - fit}{fit}, \tag{8}$$

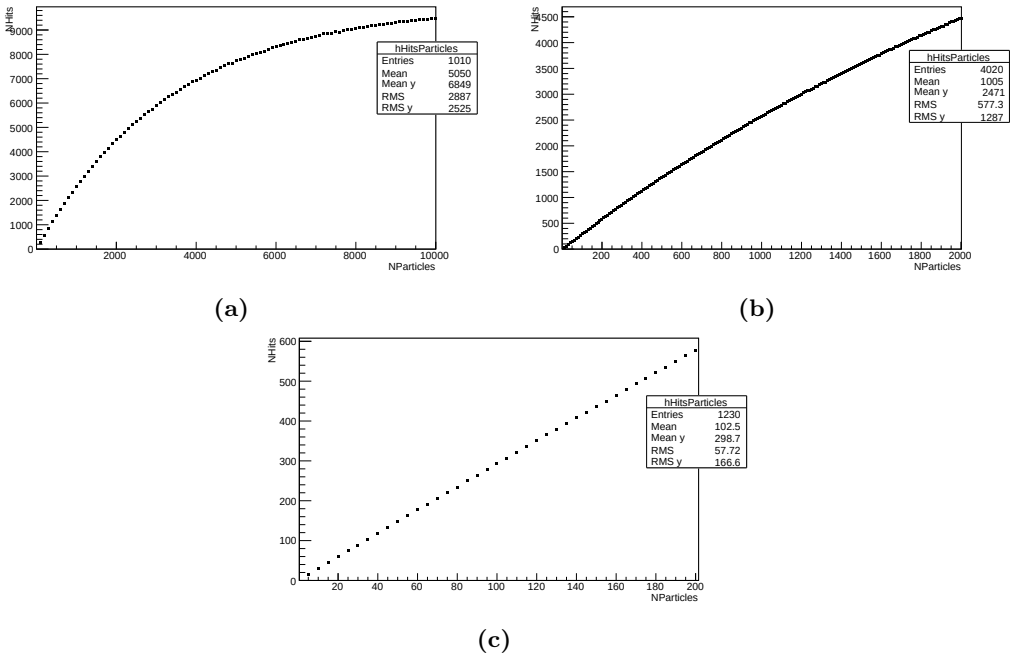


Figure 25 – Hits vs particles obtained by the toy model for a pixel map of size 100x100 using a fixed cluster size of 3 and a uniform distribution of particle positions. Each data point represents the average over (a) 10 (b) 20 and (c) 30 repetitions of the model.

where the variable names speak for themselves. One of these panels is shown in the bottom part of Figure 28a.

To demonstrate that we indeed fixed the neighbour selection problem discussed in the previous section, we observe that the top part of Figure 28a contains a linear fit with a slope $p_1 = 2.97$ and $\chi^2 = 22.95$. Keeping in mind that the border pixels in our grid do not have as many neighbours as the central pixels, this number is very close to the imposed cluster size of 3.

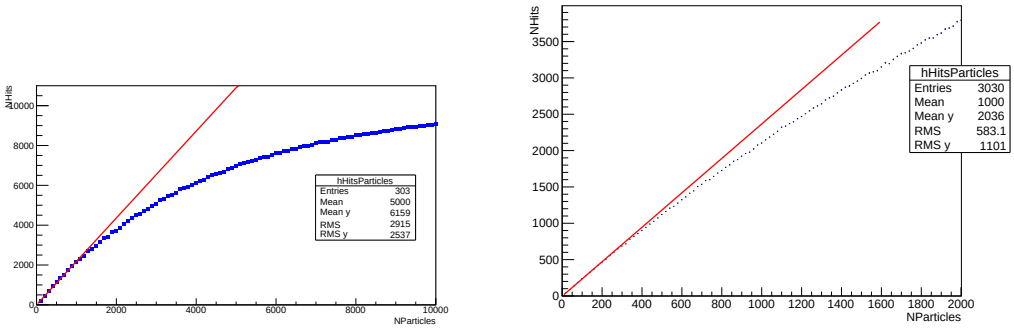
As mentioned, we try to find the point where saturation comes into play by examining uniformly distributed layers which should be randomly filled. By setting the cluster size to a fixed value we can then get a feeling of the impact saturation has at small particle numbers. To achieve that, we show four plots with relatively small particle ranges including a ratio panel that shows the differences between data and fit in more detail. It should be noted that we keep the number of repetitions of the model equal in the following plots. Specifically, we use 30 repetitions for each particle range.

Firstly, Figure 28a shows a plot displaying the hits for a number of particles in the range between 0 and 20. We already pointed out that the slope corresponds to the cluster size value. Moreover, the panel shows ratio values that spread around 0, indicating tiny differences between the fit and the data points.

As we do not yet observe differences in the ratio panel, we increase our particle range to values up to 50. That is shown in Figure 28b. Again, no significant deviations from the fit can be spotted in this particle range.

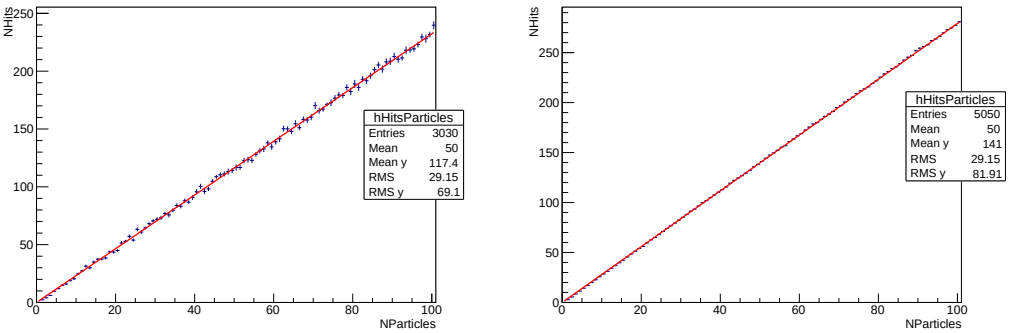
We rinse and repeat our previous action of increasing the particle range that we examine. Figure 29a shows the outcome when we increase the range up to 100 particles. The first half of the plot is similar to the previous one, showing no significant deviations in the ratio panel. However, when we look closely at the right part of the panel, from 50 up to 100 particles, we start to observe a decreasing trend in the ratio values. Moreover, looking at the top part of the figure, we see that the data points are slightly below the red fit line towards the higher values.

Therefore, we expand our range even further up to 200 particles. Within that range, shown in Figure 29b, we clearly observe a continuity of the decreasing trend in the ratio panel towards a higher number of particles.



(a) **(b)**

Figure 26 – Hits vs particles obtained by the toy model for a pixel map of size 100x100 using the cluster size distribution shown in Figure 10b and a uniform distribution of particle positions. Each data point represents the average over (a) 3 and (b) 30 repetitions. The red line shows a linear fit between (a) 0 and 1000 and (b) 0 and 200, propagated towards higher particle values.



(a) **(b)**

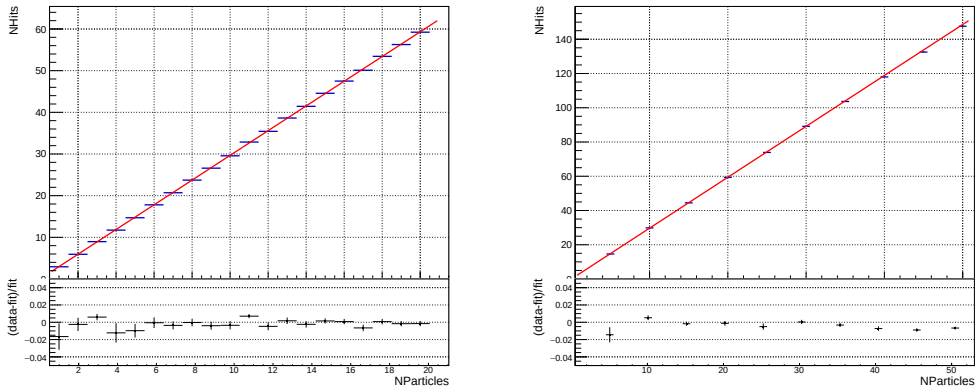
Figure 27 – Hits vs particles obtained by the toy model for a pixel map of size 100x100 using (a) the cluster size distribution shown in Figure 10b and (b) a fixed cluster size of 3, and a uniform distribution of particle positions. Each data point represents the average over (a) 30 and (b) 50 repetitions. The red line shows a linear fit between 0 and 20 propagated towards higher particle values.

4.2.5 Hit densities and particle densities

In our initial data analysis, described in Section 3.3.1, we often used the concept of hit density. We used the hit density due to the large amount of dead pixels in the FoCal prototype. Fortunately, we now have a model where we can keep track of both the hit density as well as the actual particle density in the pixel map. By comparing the two we might be able to find some form of correction that helps us interpret the FoCal data more thoroughly.

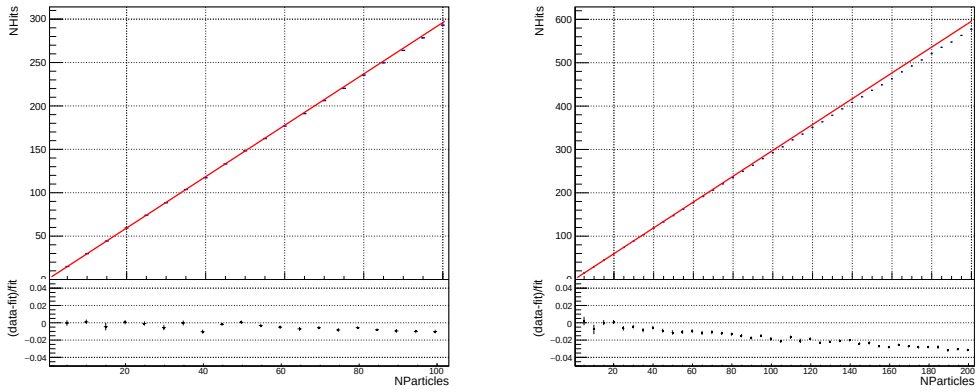
For that comparison we use a a single pixel map of size 100x100 in which we shoot 2000 particles uniformly distributed and with a fixed cluster size of 3. Similar to the rings around the central shower position used for FoCal data, we divide our map into 10 square areas around the center. For each of these squares, we calculate both the hit density and the particle density where we use $pixelSize = 1$ when calculating the total area of a square.

Figure 30 shows both densities in the 10 rings around the central grid position. It should be noted that this was the final part of our physics research and due to time considerations we were not able to finish it properly. For example, we only performed the calculation of densities once and thus we are unable to show uncertainties in the figure. Therefore, we should not draw any conclusions from this result. However, it shows a path towards further research into this subject that future students might walk upon.



(a) **(b)**

Figure 28 – Hits vs particles obtained by the toy model for a grid of size 100x100 using a fixed cluster size of 3 and a uniform distribution of particle positions. Each data point represents the average over 30 repetitions. The red line shows a linear fit between 0 and 20 propagated towards higher values. The fits have a slope (a) $p_1 = 2.97$ and $\chi^2 = 22.95$ and (b) $p_1 = 2.97$ and $\chi^2 = 6.95$.



(a) **(b)**

Figure 29 – Hits vs particles obtained by the toy model for a pixel map of size 100x100 using a fixed cluster size of 3 and a uniform distribution of particle positions. Each data point represents the average over 30 repetitions. The red line shows a linear fit between 0 and 20 propagated towards higher values. The fits have a slope (a) $p_1 = 2.96$ and $\chi^2 = 1.34$ and (b) $p_1 = 2.99$ and $\chi^2 = 2.34$.

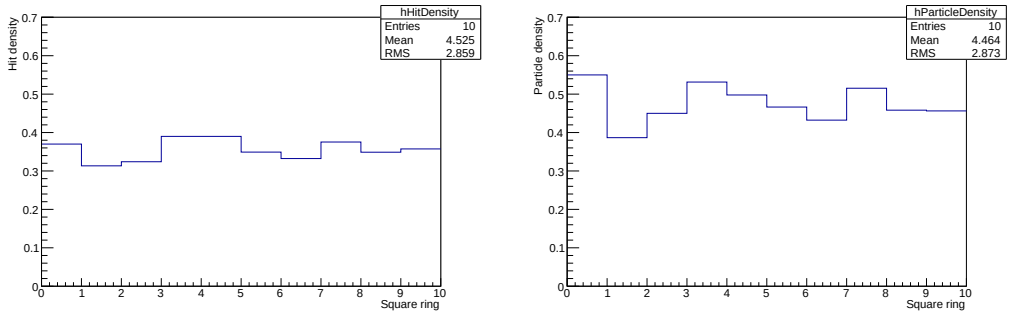
5 Discussion

In this section we give a more detailed discussion on the results described in the previous section. We do this for both the initial data analysis as well as the toy model results.

We begin with discussing our initial data analysis. As described in Section 1.3, the main goal we had for that part was to find or construct a three-dimensional model that we could fit to data obtained with the FoCal prototype. Two-dimensional fits, such as Equation 7, on lateral profiles for a single layer had already been performed. In order to extend these to an additional dimension, we had to make sure that the longitudinal and lateral profiles we used were as smooth and clean as possible.

However, as shown in Figure 11a, the longitudinal profiles for the inner rings in a layer are not smooth. We see clear signs of an effect that has similar properties to that of saturation. Thus, it would prove problematic to use those profiles for our fit. We examined implementation of calibration constants and of the effect of charge diffusion. Although, unfortunately, both of these did not aid in obtaining better profiles, we did answer many questions on how these effects were actually implemented.

That was something we came across more than once. A lot of research had already been



(a) **(b)**

Figure 30 – Hit and particle density as calculated from the pixel map showing 2000 uniformly distributed incoming particles using a fixed cluster size of 3. Figure (a) shows the hit density and (b) shows the particle density for 10 square rings around the center.

conducted on the FoCal prototype, but the specifics of what was actually used and where something was stored was often unclear. Additionally, many different versions of software packages and macros were floating around with similar names, making this issue even more complex. Therefore, in the future it would be wise to store the important parts of the code on a git repository. This allows for version control which takes away many of the issues.

Continuing our examination of the saturation effect, we decided to analyze the linearity of several observables to investigate at what energy the effect start to play a significant role. In Figure 18 we show that there are some minor indications towards saturation at an energy of 244 GeV. Moreover, the total deposited energy appears more linear but is only available in the simulation. We therefore aimed at obtaining a correction factor between these two observables, allowing us to acquire the most accurate result for the actual FoCal data and thereby possibly solving the saturation problem. We were also in the process of investigating the energy resolution for these different observables, as well as a study of the hit distributions to search for any discrepancies. At that point, the quark cluster crashed and our research was halted.

Thus, we built the toy model in order to continue our study of saturation. One clear advantage of the toy model is that we have information on both the number of hits as well as the number of particles in a pixel map. That way, we have a clear way of indicating the amount of information lost. However, our model is a greatly simplified version of the FoCal prototype. Therefore, we should be hesitant with the extrapolation of conclusions.

In Figures 20 and 21 we show pixel maps where the incoming particles follow, respectively, a uniform and a Gaussian distribution. These are both relatively straightforward distributions are can be easily implemented. A next logical step would be to implement distributions with a more physical background to them, such as Equation 7. While doing that, we argued that for our study of saturation a uniform distribution of incoming particles would allow us to get a better understanding of its impact. When the incoming particles have no preferred position but are randomly distributed, we reasoned, we can observe at what point the saturation effects become significant. Thus the necessity of implementing more complex distribution functions became obsolete. However, we should realize that by doing so we move away from the physical picture that caused us to examine saturation in the first place. Specifically, the hit density distribution in a layer of the FoCal is not uniform.

Similarly, we had a choice as to which specific cluster size distribution we would use. Figure 10 shows two of these distributions. A different option was to not use a distribution at all, but to set the cluster size to a fixed value. That way, the fluctuations between multiple runs of the model would be reduced significantly. The difference is clearly visible in Figures 27a and 27b, where we compare a plot using a distribution coming from FoCal data and a plot using a fixed cluster size of 3. For our study we chose the latter, as it provided us with more stable output from the toy model. Therefore, it would be easier to detect small deviations from the linear fit. These would then surely come from saturation effects and not from fluctuations ingrained in the model output. Again, this is a trade-off between conducting efficient research and obtaining realistic results.

As to the saturation effects themselves, we show our study of these in Figures 28 and 29.

What we observe in these four plots is a decreasing trend in the ratio panel starting around an amount of 50 simulated particles. Such a decreasing trend means that the fit starts to overestimate the values of the data points. In other words: the data starts to deviate from linear behaviour. This deviation reaches a value of 2% at 100 simulated particles and continues to decrease towards higher particle numbers. This suggests that saturation effects are present, even at these low particle values. The effect is still small though, but it becomes more and more prominent towards higher values. We should therefore always wonder what we find an acceptable value for a specific project. When we now make the connection back to a more physical interpretation of the results, we would imagine the effects of saturation to be more striking when using a distribution based on FoCal data. The distribution of hits is then focused around a central position, meaning clusters have a higher probability of overlapping and thus creating saturation effects.

Finally, another way to improve the comparison with actual data is to examine the hit densities and particle densities. Unfortunately, we did not get the chance to do this thoroughly. However, this would be a good start for any further research into this subject.

6 Applied Data Science

Part of this research project is devoted to the Applied Data Science profile at Utrecht University. The following section will cover everything that we did with respect to this subject. In particular, we explore a widely used data science algorithm and investigate its applicability in our line of research, namely physics.

For completeness, we start with a short introduction on Applied Data Science (ADS). The field of ADS is easily mistaken for fundamental Data Science, which focuses on the development of new techniques and algorithms to perform Data Science tasks with [8]. However, with Applied Data Science we only use existing techniques and algorithms and we try to incorporate those into the daily practices of domain experts. Figure 31 shows a clear visualization of what ADS entails and how we can differentiate it from its fundamental counterpart.

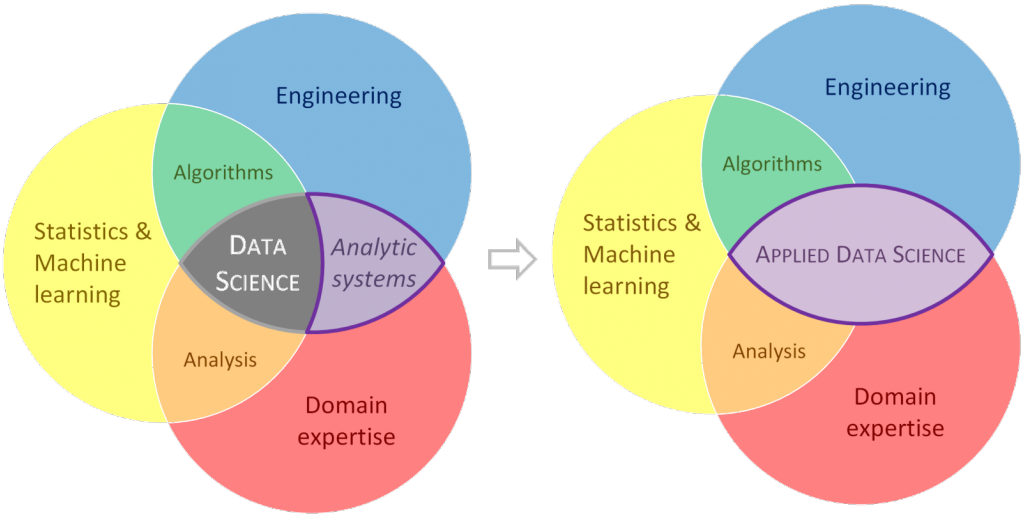


Figure 31 – [11] Data Science versus Applied Data Science.

The bulk of this research project is performed on subjects related to physics. Physics, therefore, is the domain on which we aim to apply existing techniques or algorithms. As we already did a great deal of research, we can safely assume that we are the domain experts for this specific project. Furthermore, we use results obtained from our physics research which can only be comprehended with intimate knowledge of the subject.

Nowadays, typical data science projects follow a standard process model. The most widely used model is CRoss-Industry Standard Process for Data Mining (CRISP-DM) [2]. We use the following five phases of the model: domain understanding, data understanding, data preparation, modeling and evaluation. In the rest of this section, we describe our ADS project in accordance with the CRISP-DM model.

6.1 Domain understanding

The first phase is mainly aimed at determining project objectives. Specifically, we try to formulate a data science problem using the knowledge that we have from our physics domain.

6.1.1 Determine objectives

In the first part of this thesis, one of the main objectives was writing and using a toy model in order to obtain data similar to data acquired with a prototype of an actual detector. A description of that prototype, called FoCal, is given in Section 2.2. Moreover, a detailed explanation of the toy model can be found in Section 3.3.2.

The FoCal is a digital calorimeter which specializes in measuring the energy of photons. By combining information from 24 layers of pixel detectors, profiles can be constructed from which we can subsequently determine the energy of particles that traversed the entire prototype. However, in the collision of for example two protons, many more particles than just a single photon are created. We, on the other hand, are only interested in single photons with a high energy that come directly from the collision point and we try to focus specifically on these. For more information on the physics behind this, see Section 1.2. As mentioned there, we deal with background processes in which two photons are created with a somewhat lower energy. When these two photons enter the prototype simultaneously, they yield a profile similar to that of a single high energy photon. Our final goal is to be able to distinguish between these two cases and select only the events with a single photon that are interesting for us.

To achieve that, we first try to do this for a simpler case. More specifically, we first distinguish between the case in which there is no photon in the detector and the case in which there is a high energy photon in the detector. As we have little experience in this area, achieving this would already be an accomplishment.

6.1.2 Assess situation

In order to accomplish the objectives described in the previous section we expand upon several important factors. The most prominent factors include the used data, computing resources and software. We briefly touch upon all of these in the following section.

At first, the data obtained with the FoCal prototype seemed adequate to use for this type of data science project. That data had been processed and cleaned into a format that was easily usable. Moreover, it was stored on a centralized cluster, allowing us to run our code faster on bigger chunks of data. However, due to a crash of that cluster, we lost the processed data and were left with only the raw data. Processing that raw data would have taken up too much of our time, so we decided to write our own toy model to mimic the data that we wanted to use at first. This is far from ideal, but on the other hand it does allow us to have more influence on the content of the data. Now, we can easily create data of the cases of interest described in the previous section. Furthermore, it allows us to play with the dimensions of the data. That way we are able to keep the data size within the limits of our computing resources and software.

We were constrained to the use of a laptop as the main computing resource. That laptop has a dual-boot Windows/Linux, where Linux is used for writing and executing code. Linux is useful in that aspect, but the dual-boot inevitably shrinks down the available memory for the operating system. Therefore, we should follow the memory usage of our software applications closely to prevent missteps.

The software applications that we use are Python [9] and the corresponding library Keras [3]. Python is a dynamic, open source programming language used in many contemporary data science projects. Keras is a deep learning library written in Python, mainly used to create high-level neural networks.

6.1.3 Determine data mining goals

We translate the objectives mentioned before to specific data mining goals. That way, we are able to use the obtained results to evaluate the success of this project using objective criteria.

Firstly, our goal is to obtain a binary classifier that has pixel maps as input and a binary classification as output. The binary classification allows us to distinguish between different types of

photon data. When we have our binary classification technique, we require a criterium to determine when the classification method performs adequately. For this project we use the following criterium: if we obtain a prediction accuracy of 90% using our binary classification technique, the technique can successfully distinguish between the two used data classes.

6.2 Data understanding

The goal of this next phase is to become familiar with the data that we would like to use. As we have written the code to construct the data ourselves, we start with a good understanding of our data. Moreover, we have explained the steps in this process in Section 3.3.2, so we do not go into too many details here.

6.2.1 Collect initial data

We already performed our initial data collection during the physics part of this thesis. Using the toy model, we obtained square pixel maps with a size of 100x100. These pixel maps are shown in Section 4.2.1.

6.2.2 Describe data

Our toy model creates data that has similar properties to data coming from the FoCal prototype. Both consist of a pixel map where simulated particles contribute to the value of the pixel. Within the FoCal data pixels are binary, so they can either be fired (1) or not (0). As shown in Figure 19, the data we use contains more information. Each pixel value corresponds to the number of particles contributing to the value of that pixel.

The maps used for our physics analysis have a square size of 100x100, adding up to 10000 pixels per event. The output format of these pixel maps is flexible. For the physics part we used *.root*-files as we mainly worked with the ROOT programming language. However, because we use Python for this data science part of the project, we decided to write our maps to *.txt*-files as they are easily writable and readable.

We use three distinct data classes for this project. From now on we refer to these data classes as no-photon, 1-photon and 2-photon data. The pixel maps belonging to these classes have already been shown in Section 4.2.1. Nevertheless, we match each class with the corresponding output to prevent confusion.

Firstly, we represent the no-photon class with a uniform pixel map where the incoming particles have no preferred position. An example of such a map is given in Figure 19. For the 1-photon class we decided to use a Gaussian distribution of particles such as shown in Figure 21. We could have also implemented more physical distributions representing a photon like Equation 7. However, the implementation time to include that combined with the fact that this is a proof of concept convinced us that a Gaussian distribution will suffice for this project. Finally, the 2-photon class then simply amounts to a combination of two Gaussian distributions representing a photon.

Each of the previously discussed data classes contains information on a simulation of 100 particles that we shoot at the pixel maps. Moreover, the particles have a fixed cluster size of 3. This means that a single particle fires not just one pixel, but 3 pixels. Some of these fired pixels can overlap, introducing a small fluctuation in the total number of fired pixels per event.

6.2.3 Explore data

In Section 6.1.2 we discussed the necessity of keeping the data usage within limits. Any chance we have to reduce the size of our data should therefore be explored. Therefore, we examined Figures 19 and 21 closely and concluded that we are able to shrink down the size of the pixel map while keeping the essential information intact.

When we keep the middle part of the two types of pixel maps and remove the border part, we achieve just that. The uniform maps, resembling the non-photon class, contain randomly distributed particles which does not change when we use only a fraction of the grid. In the photon class data on the other hand, the particle positions are focused towards the center of the grid. That leaves the border with relatively little information. Thus, the center of the two types of grids is where we will find the most pronounced differences.

6.2.4 Verify data quality

Before starting the process of data preparation, we first examine the quality of our data. This largely comes down to an analysis of the completeness and correctness of the pixel maps.

First of all, we acquire the data with our self-written model. As far as we know, we introduced no missing values and deliberate errors. We can thus safely assume our data to be correct in that sense. However, how well it resembles the FoCal prototype data is a more difficult question to answer. We use the Gaussian particle distribution for data resembling a photon, which is not as realistic as a distribution coming from actual data. We should therefore be hesitant in extrapolating our results to serve as conclusions for FoCal data.

Furthermore, we start our project with solely data for the no-photon and 1-photon data classes. More data for those classes can easily be obtained by running the toy model again. On the other hand, data for the 2-photon class is not yet available. We must add a method to our toy model to provide us with a way to generate data for that class. Our data is therefore not yet fully complete.

6.3 Data preparation

We continue with a description of all the data preparation steps performed to aid us in the modeling process.

6.3.1 Select data

During the course of this project we realized that adding more code to the toy model in order to obtain data representing two photons in a pixel grid was not feasible. Therefore, we decided to use just the no-photon and 1-photon data classes for our analysis. In that respect, we let go of the deeper physics question and focus more on the parametrization and implementation of data science techniques.

6.3.2 Clean data

Our selected analysis technique, which we describe in more detail later on, requires our data to have a specific size. This required size is smaller than the original of 100x100 pixels. Nevertheless, we noted that the central part of our data is where the differences between the data classes are most prominent. We thus select the central part as a subset of our data and use that as input for our analysis technique. Explicitly, we select the 32x32 most central pixels of our pixel maps. One additional advantage is that we thereby reduce the size of the data significantly.

6.3.3 Construct data

As mentioned, we acquire data from the toy model in the form of pixel maps. These pixels, as shown in Figure 19, are not binary pixels, i.e. they can obtain any integer value corresponding to the number of particles contributing to that pixel value. To represent the pixel data acquired by the FoCal prototype more accurately, we transform these integer values to binary pixel values. Any integer value equal to or above 1 returns 1, and the rest remains 0. More specifically, we define a function f as

$$f(x) = \begin{cases} 1 & \text{for } x \geq 1 \\ 0 & \text{for } x = 0 \end{cases} \tag{9}$$

that we map on all values of our pixel map. As a result, we obtain a pixel map consisting of only 1's and 0's.

6.3.4 Format data

The final data preparation step comprises data formatting. We divide this step into two main assignments: labeling the data and providing the correct dimensionality of the data.

When reading in the *.txt*-files with Python we ensure proper labeling of the two data classes. To accomplish that we have our toy model return the first half of the output as no-photon maps and the second half as 1-photon maps. That way we can simply add a binary integer as a label to each

of the pixel maps, where 0 represents the no-photon class and 1 the 1-photon class. Subsequently, we shuffle the data randomly while keeping the data-label pairs intact.

Furthermore, our analysis technique requires the data to be inserted as a four-dimensional array. This array should have the following format:

$$format = \begin{bmatrix} data \\ size\ x \\ size\ y \\ size\ z \end{bmatrix} \tag{10}$$

where x, y and z represent the dimensions of the input data. In our case, this corresponds to $(x, y, z) = (32, 32, 1)$. Additionally, *data* represents the number of pixel maps that we use for the training and testing phases of our model. We discuss that in more detail in the next section.

6.4 Modeling

The following section describes the actual building and tuning of the model, as well as an assessment of the obtained results.

6.4.1 Select modeling technique

As described in the previous sections, our goal is to distinguish between the no-photon and 1-photon data classes. Therefore, we require a modeling technique that serves as a binary classifier for these two types of classes. Furthermore, the model should be able to handle data in the form of a pixel map, allowing us to insert our data without producing many derived attributes. Finally, we use labeled data and should thus use a supervised learning technique. By combining these requirements we were guided towards the use of a Convolutional Neural Network (CNN) with back-propagation to act as a binary classifier for the two photon data classes.

6.4.2 Generate test design

In order to test the quality and validity of our model, we generate a test design before building the model in the next section. To that extent, we separate our data into two parts: a train set a test set. This is a typical approach when performing a classification task. We use the train set to build and train our model, which in our case involves tuning the weights of our neural network. Subsequently, we use an independent test set to estimate the quality of the model. We use the test accuracy of our trained model as an estimation of its quality, where we define:

$$\text{accuracy} = \frac{\text{Correctly classified samples}}{\text{Total amount of samples}}. \tag{11}$$

To ensure that there is no correlation between the train and test sets we generate them with independent executions of the toy model. First, we generate a certain amount of training samples and store them in a separate file. Then we generate a set of test samples, where the number of test samples equals 20% of the number of training samples. For this project, we generate a training set containing 5120 samples and a test set with 1024 samples.

Most neural networks do not achieve optimal results after going through the training set once, which is called one epoch. More iterations are required to tune the weights appropriately within the network. Therefore, we assess the test accuracy of our model for multiple epochs. Moreover, we are unable to feed the entire training set to the CNN at once due to memory issues. We divide the set into batches with a batch size of 512 to resolve that.

6.4.3 Build model

At long last, we describe the overall architecture and the parameter settings of our CNN. After a period of playing with the model combined with a lot of trial and error, we decided to take the classical LeNet-5 architecture [6] as a guideline for our own model. The LeNet-5 architecture is

shown in Figure 32 and it consists of convolution layers, subsampling layers and full connection layers.

This architecture assumes an input image with a size of (32, 32, 1), which is why we selected only the central part of our pixel maps in Section 6.3.4. Subsequently, the first layer in the CNN is a convolution layer with a (5x5) kernel and stride 1. Kernels are used to detect features in an image or a pixel map. Essentially, a kernel is a matrix containing weights. Performing a convolution operation on part of an image with a corresponding kernel then returns a large number if a known feature is observed, and a small number if nothing is recognized. An example of a convolution operation on an image using a (3x3) kernel is shown in Figure 33. The stride determines how many times we perform a convolution operation on a pixel map. The value we choose for the stride signifies the number of pixels we move until we perform another operation. One can play with this value in order to reduce the dimensions of the data within the hidden layers of the neural network, resulting in faster performances. In our case we use stride 1, so we do not significantly reduce the dimensions with the convolution layer.

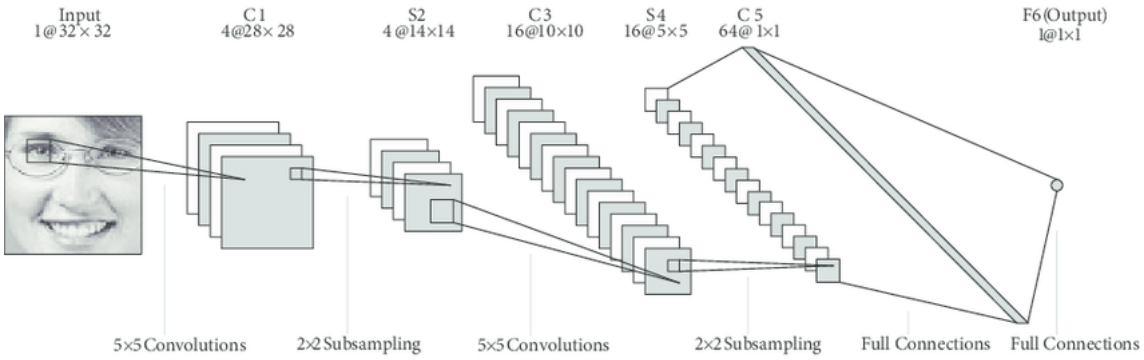


Figure 32 – [6] Architecture of the classical LeNet-5 Convolutional Neural Network.

However, we reduce the dimensions using a subsampling layer. We specifically use an average pooling layer with (2x2) pooling and stride 1. The pooling operation also examines part of the pixel map, but instead of performing a convolution, it takes the average value of the pixels inside the (2x2) pooling window. Afterwards, we have a similar convolution layer with a (5x5) kernel and stride 1, followed by another average pooling layer with (2x2) pooling and stride 1. Finally, we end with two full connection layers that map the processed input onto a binary classification. The first full connection layer has a size 64 and the final full connection layer has size 2, representing the two classes to classify.

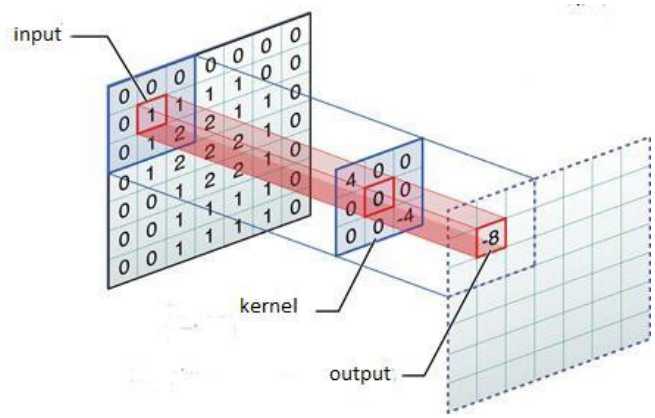


Figure 33 – [4] Example of a convolution operation in a CNN using a (3x3) kernel.

Apart from a kernel and a stride, a convolution layer requires an activation function. The output of a convolution operation passed through such a function, thereby allowing us to introduce non-linear properties into our neural network. After the output is passed through the function, it is used as input for the next layer in the network. The most widely used activation function for a

hidden convolution layer is the rectified linear unit (ReLU):

$$R(x) = max(x, 0). \tag{12}$$

This function has output 0 if the input is less than 0 and raw output otherwise. We use the ReLU function in both our convolution layers.

Furthermore, the full connection layers require an activation function. For that we use the Softmax function. Softmax ensures that all values in the layer are between 0 and 1, and that the total sum of all values equals 1. Essentially, it performs a normalization that produces a categorical probability distribution of all values in the layer. The final layer in our CNN with size 2 then indeed represents a binary classification.

A common pitfall when working with neural networks is over-fitting the model on the train set. Then the model performs excellent on the train data, but worse when presented with an uncorrelated test set. To diminish the effect of over-fitting, we introduce a dropout at three points in our CNN. After each of the two subsampling layers we set the dropout equal to 0.25, signifying that 25% of the nodes in that layer and their connections to next layers stop contributing to the classification process. These nodes are chosen randomly and thus vary between different executions. Moreover, after the first full connection layer we introduce a dropout of 0.5, thereby dropping 50% of the nodes and their connections.

6.4.4 Assess model

Using the test design described earlier, we assess the quality of our Convolutional Neural Network. More specifically, we assess its binary classification capability with respect to our no-photon and 1-photon data classes. We do this on the basis of the test accuracy of the trained model. In order to maintain a systematic approach, we study the accuracy for a variable number of epochs. Moreover, for each epoch we perform the training and testing of the model several times to acquire a sense of the stability of the model.

Figure 34 shows the result of that for up to 10 epochs, where we executed the training and testing phases 5 times for each of the epochs. The plot shows the mean accuracy over those 5 iterations and the uncertainty bands correspond to the standard deviation of the mean (SDOM).

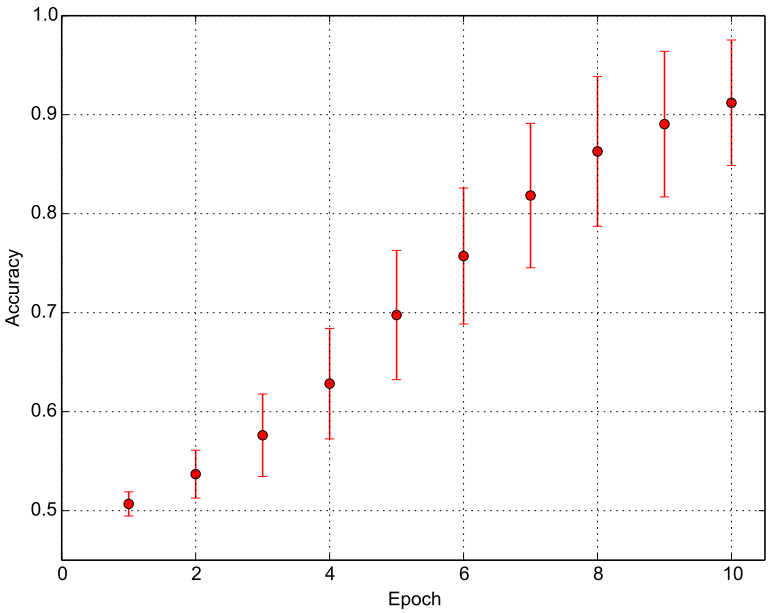


Figure 34 – Accuracy of the binary classification of the CNN model for up to 10 epochs. The accuracy represents the average over 5 executions of the model with the standard deviation of the mean as uncertainty.

The first thing that stands out is the fact that after 1 epoch the model has learned practically nothing. Our test data consists of one half no-photon and one half 1-photon data samples. Hence, an accuracy of 0.507 ± 0.012 comes down to the accuracy we would obtain when we predict all samples to be one of the two classes. After that, we observe a gradual increase in accuracy with a higher number of epochs. If we look closely, we notice that the increase in accuracy starts relatively slow, is maximal between epochs 4 and 7 and finally falls off again. We acquire the highest accuracy after 10 epochs and it equals 0.912 ± 0.063.

In Section 6.1.3 we discussed the technical data mining goals set for this project. First, we succeeded in building a binary classification method for our pixel maps. Furthermore, we obtain an accuracy that is in accordance with 0.9 (90%) after 10 epochs. Of course, the uncertainty on that result is still relatively large, but it does show a firm indication that we can obtain an even more successful classification with either more iterations, or by increasing the number of epochs. We thus conclude that we have achieved both data mining goals.

Finally, we examine the success of our domain objectives described in Section 6.1.1. In principle we discussed two objectives with a relevant physical background. First, we aimed for a classification between the no-photon and the 1-photon data classes obtained with our toy model. Second, we would introduce the 2-photon class in order to perform a similar classification between the 1-photon and 2-photon data classes. Unfortunately we were only able to focus on the first classification problem in this project. However, we achieve a classification accuracy of 0.912 ± 0.063, from which we conclude that we are able to successfully distinguish between the no-photon and 1-photon pixel maps.

6.5 Evaluation

The final phase covers the evaluation of the general success of the project. This section is not as technical as the previous assessment, but focuses more on the process itself.

6.5.1 Evaluate results

As we discussed in the previous section, we were able to achieve one of the two domain objectives that we set beforehand. Although it sounds like we thus only performed half of the work, that is not the case. We successfully built and tuned a Convolutional Neural Network and used it as a binary classification technique. The only thing that was missing was data for the 2-photon data class, which hindered us in achieving the second domain objective.

For the CNN we used a classical LeNet-5 architecture. This specific type of architecture is used, among other things, for gender classification and the classification of handwritten digits. Thus, such a network is able to detect detailed features in an image. In hindsight, we might have done well to use a slightly less complex architecture in order to save time or to reduce the memory usage of our model. However, this might be the case for the classification of no-photon and 1-photon data classes, but whether or not it holds for the 2-photon classification case is questionable.

6.5.2 Review process

Unfortunately, this data mining project had a rather bumpy process with respect to the data itself. We started the full research project with a large amount of data from the FoCal prototype. With that data we intended to perform a classification on the number of incoming particles, which would have direct relevance to research conducted in our specific physics domain. However, due to a server crash this data was lost and we were forced to generate our own data. Such data is unavoidably simplified and therefore less realistic.

Nevertheless, we decided to stick with the goal of a classification on the number of incoming particles. Now, however, using our self constructed data instead of the prototype data. Thus, the results of this project would serve more as a proof of concept rather than having any direct relevance for the FoCal detector itself.

6.5.3 Determine next steps

Finally, because of the extra time it took to generate our own data we decided to focus solely on performing one accurate, thorough classification instead of two. Therefore the logical next

step would be to generate 2-photon data and perform the 1-photon vs 2-photon classification. Furthermore, this could be extended to a classification using three distinct data classes. Lastly, it would be ideal to be able to use the actual FoCal prototype data and perform the classification to solve the original physics problem.

7 Conclusion

Throughout our conducted research we acquired results for three distinct subjects. We discuss the the main conclusions for each of these parts.

First of all, we performed a data analysis on the FoCal prototype data. The first preparation steps towards constructing a three dimensional model of shower profiles have been taken. These steps include obtaining longitudinal and lateral profiles from both the FoCal prototype data and Monte Carlo simulations. Moreover, calibration factors have been implemented in the FoCal data. As an attempt to solve the issue of saturation in longitudinal hit density profiles, simulations have been performed both with and without charge diffusion applied. The total number of hits as well as the total deposited charge in all layers can be obtained from these simulations. Finally, by combining the previous steps we checked the linearity of the two described observables as a function of energy. From the linearity plot it appears that saturation starts to play an important role at an energy of 244 GeV.

Moreover, we constructed a toy model to continue our study of the effect of saturation. Using a uniform distribution of incoming particles combined with a fixed cluster size of 3 and grid of size 100x100, we observed the first signs of saturation at a value of 50 simulated particles. Furthermore, we observe a deviation from linear behaviour of 2% at 100 simulated particles. Although the saturation effect is small at that point, it increases steadily towards higher numbers of simulated particles. We expect saturation effects to be more significant when using a more physically reasoned distribution of incoming particles.

Finally, part of this research project was devoted to the Applied Data Science (ADS) profile. To that extent, we generated two types of data classes with our toy model: no-photon data and 1-photon data. We built and tuned a Convolutional Neural Network following a classical LeNet-5 architecture to act as a binary classification technique for these two data classes. Using 5120 training samples and 1024 test samples, we reached a classification accuracy of 0.912 ± 0.063 after 10 epochs. Thereby, we successfully achieved the data mining goals for one of the two domain objectives.

References

[1] G. Bathow, E. Freytag, and K. Tesch. Measurements on 6.3 GeV electromagnetic cascades and cascade-produced neutrons. *Nucl. Phys.*, B2:669–689, 1967.

[2] Pete Chapman, Julian Clinton, Randy Kerber, Thomas Khabaza, Thomas Reinartz, Colin Shearer, and Rudiger Wirth. Crisp-dm 1.0 step-by-step data mining guide. Technical report, The CRISP-DM consortium, August 2000.

[3] François Chollet et al. Keras. https://keras.io/, 2015.

[4] Alejandro Escontrela. Convolutional neural networks from the ground up. https://towardsdatascience.com/convolutional-neural-networks-from-the-ground-up-c67bb41454e1, 2018.

[5] P. Hansson. The parton model, 2004.

[6] Shan Sung Liew, Mohamed Khalil-Hani, Feeza Radzi, and Rabia Bakhteri. Gender classification: A convolutional neural network approach. *Turkish Journal of Electrical Engineering and Computer Sciences.*, 24:1248–1264, 2016.

[7] Walter R. Nelson, Theodore M. Jenkins, Richard C. McCall, and Joseph K. Cobb. Electron-induced cascade showers in copper and lead at 1 gev. *Phys. Rev.*, 149:201–208, Sep 1966.

[8] P. Pritzker and W May. Nist big data interoperability framework: Volume 1, definitions. *NIST Special Publication 1500-1*, September 2015.

[9] Python Core Team. *Python: A dynamic, open source programming language*. Python Software Foundation, 2015.

[10] Martijn Reicher. Digital calorimetry using pixel sensors, 2016.

[11] Marco Spruit and Raj Jagesar. Power to the people! - meta-algorithmic modelling in applied data science. In *Proceedings of the 8th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management - Volume 1: KDIR, (IC3K 2016)*, pages 400–406. INSTICC, SciTePress, 2016.

[12] Yung-Su Tsai. Pair production and bremsstrahlung of charged leptons. *Rev. Mod. Phys.*, 46:815–851, Oct 1974.

[13] Hongkai Wang. Prototype studies and simulations for a forward si-w calorimeter at the large hadron collider, 2018.

[14] Richard Wigmans. *Calorimetry - Energy Measurement in Particle Physics*. Oxford Science Publications, 2000.

[15] Chunhui Zhang. Measurements with a high-granularity digital electromagnetic calorimeter, 2017.