
A MACHINE LEARNING ALGORITHM FOR ENERGY RECONSTRUCTION AND BINARY CLASSIFICATION OF ELASTIC AND INELASTIC NEUTRINO SCATTERING EVENTS AT THE SND@LHC

BACHELOR REASEARCH THESIS

AUTHOR:

JOYCE GERARDA HELENA COBUSSEN

RESEARCH SUPERVISORS:

DR. ELENA GRAVERINI

DR. JACCO DE VRIES

MSC. PAUL DE BRYAS

DATE:13-12-2020

MAASTRICHT SCIENCE PROGRAMME, MAASTRICHT UNIVERSITY

MAASTRICHT, THE NETHERLANDS



Acknowledgements

First of all, I would like to offer my special thanks to my external supervisor Dr. Elena Graverini, for all the time and effort she invested in answering questions, discussing ideas and providing feedback on numeral occasions. This was not only very helpful, but also consistently motivated me to keep up the work throughout the project.

A special thanks is also in order for Dr. Jacco de Vries, for giving me the opportunity to do this research in the first place and for always being available for questions, as well as providing positive encouragement at times when issues kept piling up.

My sincere gratitude also goes out to MSc. Paul De Bryas, for attending the meetings on a strictly voluntary basis and sharing his useful knowledge, insights and suggestions. Many problems would not have been solved as swiftly without his help.

I also thank the Nikhef Collaboration and EPFL, for providing the computing resources used to carry out this project and allowing me to use them extensively.

Futhermore, I would like to thank my colleague and partner in crime Shania Mitra for the many discussions we had on both professional and personal aspects of this thesis.

I also thank my dear friend Djura Knol for always motivating me to work, but especially for providing me with much-needed distraction when I needed it the most.

Finally, I want to thank my parents for always supporting me and showing interest in my work, despite it not being their field of expertise. Without them I would not have gotten this far.

Abstract

This Bachelor Research Thesis (BTR) aims to improve the accuracy of energy reconstruction for particle showers within an energy range of 200-400 GeV passing through the Scintillating Fibre (SciFi) planes of the prospective Scattering and Neutrino Detector at the Large Hadron Collider (SND@LHC). To this end, the effects of adjusting several parameters of an algorithm using a Convolutional Neural Net (CNN) are investigated. The implementation of different loss functions significantly improved the accuracy of the algorithm and greatly reduced the overestimation of the predicted shower energy. This further prepares the algorithm for realistic implementation at the SND@LHC. Additionally, the BTR aims to create a classification algorithm for the identification of elastic and inelastic electron neutrino scattering events. The energy range of interest is 100 - 2000 GeV, which corresponds to the expected neutrino energy at the SND@LHC. An accuracy of 94.36% was achieved for the most realistic scenario. Moreover, the overall results provide a useful first insight into the (dis)advantages of using a CNN for particle shower classification. Further research is recommended to improve the algorithm's ability identify particle showers that leave only a few hits in the detector.

Table of Contents

Acknowledgements.....	2
Abstract.....	3
Table of Contents.....	3
1. Introduction.....	5
1.1 Physics background.....	5
1.2 Aims and hypotheses	7
1.2.1 Energy reconstruction	7
1.2.2 Classification of elastic and inelastic events	7
1.3 Basic principles of Machine Learning.....	8
2. Methods	11
2.1 Energy reconstruction	11
2.1.1 Dataset	11
2.1.2 Pre-processing.....	12
2.1.3 Reducing dimensions.....	13
2.1.4 Structure of the energy reconstruction algorithm.....	14
2.1.5 Bias minimization	15
2.2 Classification.....	15

2.2.1 Dataset	15
2.2.2 Pre-processing	16
2.2.3 Structure of the classification algorithm	18
2.2.4 Hyperparameters	19
2.3 Computing resources	19
3. Results	20
3.1 Bias minimization	20
3.1.1 Loss function analysis	20
3.2 Classification.....	22
3.2.1 Image resolution.....	22
3.2.2 CNN structure analysis	23
3.2.3 Increased minimum number of hits	30
3.2.4 Elimination of greyscale	32
Summary and Conclusions	33
Outlook	34
Bibliography	35
Appendix	37
A. Critical Reflection	37

1. Introduction

1.1 Physics background

The Standard Model (SM) has been very successful at providing predictions for particle physics experiments throughout the past decades. Prominent examples include the prediction of properties of the W and Z bosons, as well as the recently discovered Higgs boson [1, 2]. However, there remain observations that the SM cannot explain, such as the existence of Dark Matter (DM), matter-antimatter asymmetry in the universe and neutrino masses and oscillations [3]. Moreover, several deviations from SM predictions have been building up in measured decay rates of B-hadrons by LHCb, ATLAS, CMS and Belle [4]. Although it has not been ruled out that these discrepancies may be statistical fluctuations, they might as well be the result of interactions of undiscovered particles or forces [5]. For example, one recently published paper proposes a scenario involving two new scalar fields and a Majorana fermion to explain the DM problem, based on the observed inconsistencies [6]. All things considered, it is evident that the SM is not yet complete. This calls for physics theories beyond the standard model (BSM), along with experiments to test and verify these models.

As it turns out, neutrino measurements are very suitable for testing the SM, as well as various proposed BSM theories [7, 8]. The Search for Hidden Particles (SHiP) experiment is a general-purpose experiment that intends to perform such neutrino measurements in the previously unexplored energy domain of a few hundreds of GeV, amongst other things. It will be installed in a beam dump facility at the Super Proton Synchrotron (SPS), where it can benefit from a large flux of neutrinos and other decay products from the secondary mesons produced by the collisions of 400 GeV SPS protons onto a target. The ultimate goal of this experiment is to study properties and interactions of all neutrino flavours, as well as to search for Feebly Interacting Particles (FIPs) that are predicted by several BSM theories [9].

At present, the LHC is already producing neutrinos at high energies that have not been studied before. As this provides a unique and immediate opportunity to advance the field of neutrino physics, a prototype of the Scattering and Neutrino Detector (SND@LHC), one of the parts appurtenant to the SHiP experiment, will be installed at the LHC-IP1. The detector will be placed along the proton beam direction facing the ATLAS interaction point, where it can profit from the large number of neutrinos produced at the proton-proton collisions [9]. The SND@LHC is composed of a target region followed by a muon system. The target region consists of emulsion bricks designed with Emulsion Cloud Chamber (ECC) technology, interleaved with Scintillating Fibre (SciFi) planes (Figure 1).

The inner structure of a SciFi plane is shown in Figure 2. The plane consists of a layer of densely stacked scintillating fibres glued together to form fibre mats. The interaction of a shower particle with the fluorescent material inside the detector leads to the emission of a photon. If the emissions takes place at an angle small enough with respect to the cladding of the fibre, the photon is transferred to the silicon photomultipliers (SiPMs) on the side of the plane. These voltaic devices transform the light signal into an electrical signal that can provide real-time information about the events within the detector. Hence, the SciFi tracker will allow real-time energy reconstruction of the particle showers

created by neutrino scattering events, as opposed to the emulsion bricks that have to physically be removed for analysis. As the SciFi planes will measure shower energy at several different shower depths, the SND@LHC can in principle function as a sampling calorimeter. In this case, the SciFi planes act as counters and the emulsion bricks as absorbers. Additionally, the SciFi planes will add time information to the measured events that can be used for later analysis of the emulsion brick data [10].

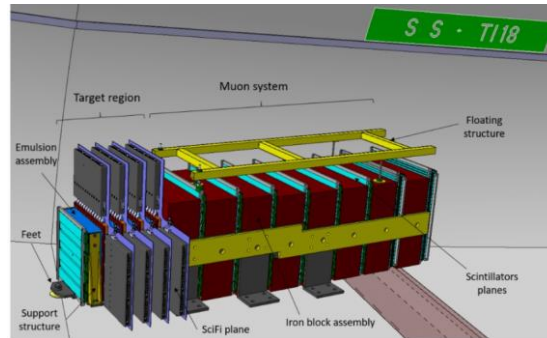


Figure 1: Overview of the main components of the SND@LHC detector [9].

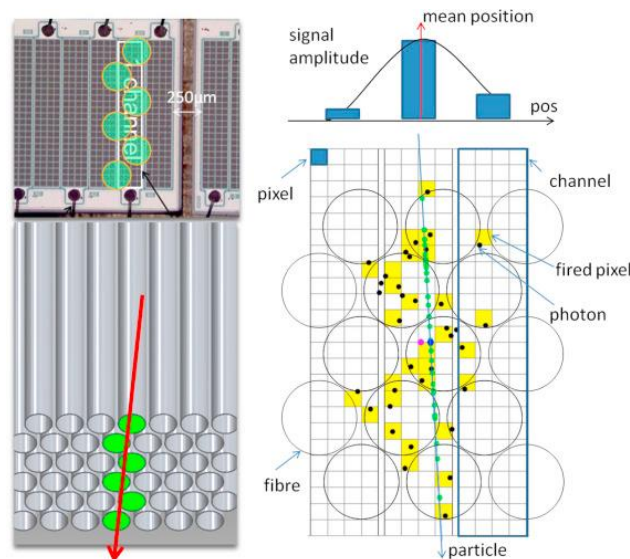


Figure 2: A schematic overview of the structure of a SciFi plane [9]. The light emitted by the interaction of a particle with the fluorescent material in the fibre is guided to the SiPMs, where it is converted to an electrical signal that can be read in real-time [9].

One of the many challenges faced by the SND@LHC is the analysis of the data gathered by the SciFi planes. The vast amount of data, in addition to its complexity, makes the use of traditional fitting algorithms unfavourable, as their inherently serial approach quickly becomes computationally expensive [11]. In these cases, Machine Learning (ML) algorithms provide a favourable strategy. These are easily parallelizable and deal better with noise and complexity, as they can find patterns without having to assume functional dependencies beforehand [12]. For that reason, this BTR will focus on using machine learning techniques to approach some of the current difficulties of the SciFi data analysis.

1.2 Aims and hypotheses

1.2.1 Energy reconstruction

Accurate energy estimation of the showers induced by particles passing through the SciFi planes is crucial the identification of these particles [13]. The basis of a ML algorithm for the energy reconstruction has already been developed by P. De Bryas [10], based on earlier work done by S. Shirobokov [14]. The algorithm uses a convolutional neural net (CNN) to predict the energy of an event based on images that represent the SciFi response to the particles, also referred to as ‘hits’. An accuracy of about 10% has been reached within an energy range of 0.3 - 100 GeV, which shows that the algorithm provides a decent base for the energy reconstruction. However, there are some limitations that need to be overcome before it can be adopted by the SND@LHC. The most prominent limitation is the overall overestimation of shower energy, which will henceforth be referred to as the bias of the algorithm. As mentioned before, accurate energy reconstruction is key to particle identification, so the accuracy must be increased before the realistic implementation of the algorithm. Hence, the main aim of this BTR is to find a way to realize the minimization of this bias. The focus will lie on investigating the effect of different loss functions on the energy reconstruction for events within a range of 200 - 400 GeV, which matches the expected energy of the particles originating at the LHC interaction point [9].

1.2.2 Classification of elastic and inelastic events

As mentioned in the introduction, another important aim of the SND@LHC is to search for so-called feebly interacting particles (FIPs) with masses in the GeV range via their scattering with detector atoms. This energy range has not been explored yet, but a number of models have been proposed that extend the interesting FIP masses to the range of GeV and below [9, 15]. A FIP of particular interest is Light Dark Matter (LDM), which can scatter elastically off the atomic electrons of the detector:

$$\chi + e^- \rightarrow \chi + e^-$$

The signature of this type of event is one EM shower induced by the scattered electron, without any additional tracks. The elastic scattering of a neutrino off an electron leaves a similar signature [16]:

$$\nu_e + e^- \rightarrow \nu_e + e^-$$

Assuming that LDM has a higher mass than the nearly massless neutrinos, plotting the shower energy against the shower direction would lead the two types of events to accumulate in slightly different areas of the plot [17]. Naturally, sufficient spatial and energy resolutions are required for this analysis. However, before such analysis can be carried out in the first place, the disentanglement of elastic events from inelastic scattering events is crucial. For this reason, the second aim of this BTR is to create a classification algorithm that is able to identify elastic and inelastic scattering events based on the SciFi response to the shower that they initiate. This algorithm will use a CNN to classify input images that are similar to those used to reconstruct the energy. The energy range of interest is 100 - 2000 GeV, which corresponds to the expected neutrino energies at the SND@LHC.

1.3 Basic principles of Machine Learning

This section will address some fundamental ideas and basic theory of machine learning algorithms, which will be referred to throughout this paper. The general goal of ML algorithms is to recognize patterns in data and use this information to treat unseen problems. They can be divided into several types based on their learning process. This BTR makes use of a so-called supervised ML algorithm, which starts with the analysis of a dataset of which the true output value is known. From this analysis, the algorithm produces an inferred function to make its own predictions for the output value. During the training process, the algorithm predicts a value for each of the elements in the dataset and compares those values to the corresponding true values. Thereafter, the quality of the predictions is evaluated by calculating the average error, also referred to as loss, for each prediction. Based on this result, each of the function parameters is updated in a certain way to improve the accuracy of the prediction during the next training round [18, 19].

To visualise this concept, the structure of a simple ML algorithm is shown in Figure 3. The input enters the algorithm on the left and arrives at the first neuron, which represents the first term of the function. The data passes through this function and a bias is added to the output, which is a constant that influences the likelihood of a certain final value to be assigned. Next, it is passed on to each of the neurons in the second layer. This process is controlled by so-called weights, which essentially controls the ‘strength’ of the signal between neurons based on how strongly they relate to each other. This process continues until the output layer is reached. Here, the outputs from the neurons in the inner layers are passed to an activation layer, where they are translated into a concrete output value.

As the complexity of the algorithm increases, the number of weights and biases that need to be updated rapidly increases. At this point, one can imagine that it may take several, if not many training epochs to reach an optimal weight configuration. The process of calculating by what value each weight and bias needs to be updated makes use of a first-order iterative optimization algorithm called gradient descent. The idea behind this algorithm is to approximate the gradient of the loss function, which measures the error between the prediction and the true value, and take a step in the opposite direction, in order to get to a (local) minimum. At this point, the values of the weights and biases are configured in such a way that the error between the prediction and true value are minimized.

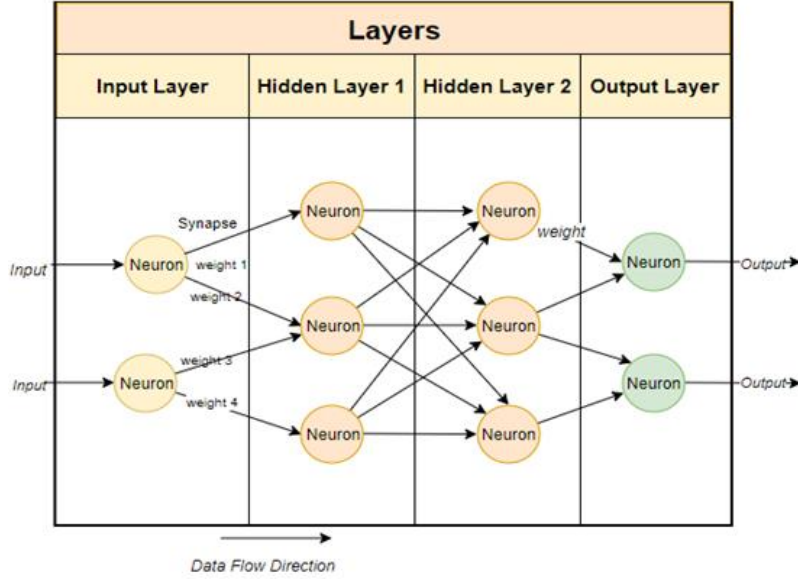


Figure 3: An example of the structure of a simple machine learning algorithm [20]. The data enters the network on the left and enters the first layer of neurons. Each element is passed on to the neurons in the next layer and finally to the output layer. A neuron represents one term of the function, the bias the constant that is added to this term, and the weight controls the strength of the signal passed to the next neuron.

Many different loss functions exist, each with their own benefits and disadvantages. For this research, three loss functions are of special interest. The first one is called the L1 loss function:

$$l(x, y) = L = \{l_1, \dots, l_n\}^T, \quad l_n = |x_n - y_n|$$

Here, x_n represents the predicted value and y_n the true value of the input. Generally, L1 loss is a robust loss function and relatively insensitive to outliers, which makes the network generally more stable [19]. The second loss function of interest was the Mean Squared Error (MSE) loss. This function is the squared version of L1 loss and looks as follows:

$$l(x, y) = L = \{l_1, \dots, l_n\}^T, \quad l_n = (x_n - y_n)^2$$

As it squares the loss, its value becomes large very quickly for any losses greater than 1. This creates a high sensitivity to outliers, but might also create a bit of instability to the network due to rapid adjustments to the weights. This may affect the performance of the overall network. The third loss function of interest, called SmoothL1 loss, combines features of the two aforementioned functions:

$$l(x, y) = \sum_n z_n$$

$$l(x, y) = \begin{cases} \frac{0.5 (x_n - y_n)^2}{\beta} & \text{if } |x_n - y_n| < \beta \\ |x_n - y_n| - 0.5\beta & \text{otherwise} \end{cases}$$

This function is relatively less sensitive to outliers than the MSE loss function, but unlike L1 loss it is a continuous function, which increases its efficiency at finding solution [18].

There exist multiple types of machine learning algorithms that all share the same basic principles, but are suited for a different types of problems or input data. A Convolutional Neural Net (CNN) is especially useful for the processing of images. It is able to identify different types of images by learning to recognize their specific features and adjusting its weights accordingly. This is the type of algorithm used for this research, as images comprise the input data. Multiple layers are often needed to properly learn characteristic features, and the configuration of those ‘hidden layers’ is key to the performance of the algorithm. There exist many types of layers that have very distinct effects on the algorithm. Below follows a brief overview of the layers of interest in for this research project.

CoordConv

The CoordConv layer adds extra channels to the incoming image containing hard-coded coordinates. This discards the translational invariance that regular convolutional layers have. As a result, the predictions of the algorithm directly depend on the absolute location and orientation of a specific feature [21].

MaxPool

A MaxPool layer effectively reduces the size of the dataset by analysing the values within a certain region or ‘kernel’ of the dataset, and only returning the highest value. Figure 4 visualizes what happens when a two-dimensional kernel of size 3x3 scans the values of a set of image pixels with a stride, or step size, of 2. Aside from reducing size of the data and thus decreasing computational cost, this operation also helps to avoid a phenomenon called overfitting. This happens when the algorithm learns features that are too specific to the training data. As a result, the predictions made for unseen data become less accurate.

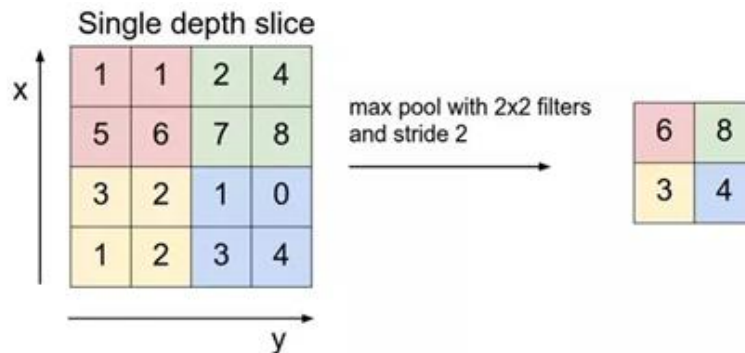


Figure 4: A schematic explanation of the effect of a MaxPool layer [18]. The coloured 2x2 blocks represent the area that the kernel searches to find and return the highest value.

BatchNorm

Batch normalization, or BatchNorm for short, normalizes each weight value by first subtracting the mean of the batch and then dividing the outcome by the standard deviation of the batch. Here, a batch refers to a subset of elements from the training data. This manipulation prevents weight values from saturating too much, which is known to destabilize the network and lead to less accurate predictions as a result [22].

Activation layers: ReLU, Linear and Tanh

Activation layers decide whether or not specific neurons should be considered to calculate the output, based on whether or not the output of the neuron is relevant to the prediction of the algorithm. This layer is key to enabling an algorithm to learn complex features. A Linear layer is a linear activation function that creates an output that is proportional to the input, thus allowing negative output values as well. A Rectifier Linear Unit, abbreviated as ReLU, is a non-linear activation function that sets any negative output values to 0 and behaves as a Linear layer for positive values. The last activation function, Tanh, compresses the output to values between -1 and 1. Activation functions are often comprise the last layer of an algorithm and their output is used to assign a final label to the data [23].

Aside from the parameters inside the CNN, there exist multiple parameters outside of the algorithm that also greatly influence the performance. As briefly mentioned before, the batch size refers to the number of elements from the data are used to make predictions and update the weights and biases every epoch. The learning rate refers to the magnitude of the step that is taken during training by the gradient descent algorithm and influences by how much the weights and biases are changed. Finally, the weight decay is a parameter that reduces the weights in the network with a given factor in order to avoid some weights from growing too large.

2. Methods

2.1 Energy reconstruction

2.1.1 Dataset

A dataset containing the Monte Carlo (MC) truth information and SciFi plane response to the events was used to train and test the algorithm. This dataset, henceforth referred to as DS1, was priorly created by De Bryas using FairSHiP, a software framework is based on FairROOT [24]. This software suite facilitates realistic simulation of events and their corresponding detector response [25]. The events are generated using Pythia6 for inelastic muon scattering [26], Pythia8 for proton-target collisions [27] and Genie for SM neutrino interactions [28]. The details about the detector geometry and material are defined by the ROOT TGeo class and the detector response to the events is simulated with Geant4 [29, 30].

DS1 was developed taking the geometry of the SND@LHC Pilot run into account, which is shown schematically in Figure 5. Contrary to the actual SND@LHC prototype has four SciFi planes instead of six. This is not a big limitation, as analysis shows that the vast majority of the hits are located in the first four planes [10]. Each plane contains six layers of scintillating fibres and have a width of $d_x = 41.6$ cm, a height of $d_y = 38.7$ cm and a thickness of $d_z = 2.3$ cm. They are placed at a distance of 9.35 cm from each other, separated by the emulsion bricks that have a thickness $d_{z,brick} = 5.6$ cm. As shown in Figure 5, the first emulsion brick is located in front of the first SciFi plane. The dataset contained 200 000 events that were initiated by electrons fired between 1 and 8 centimetres before the first plane. Their energy was uniformly distributed within a range of 200 - 400 GeV. To approach realistic

conditions, the angle at which the initial electrons were fired was equally distributed between 0 and 10 degrees [10].

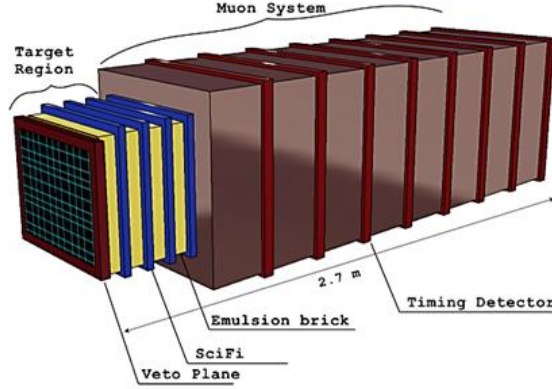


Figure 5: Schematic overview of the SND@LHC pilot detector. The Target Region consists of four emulsion bricks, interlaid with Scintillating Fibre planes., and the Muon System is located directly behind it [9].

2.1.2 Pre-processing

The detector geometry and technology places certain restrictions on the kinds of events that can be detected. For this reason, the data from DS1 was pre-processed before feeding it to the CNN. After reading the relevant Monte Carlo (MC) data from the root file, as well as the corresponding response data from the SciFi planes, the following cuts were applied to the events:

1. Only events initiated by an electron, thus those with a mother ID of -1 and PDG code of 11, were selected;
2. Events were required to have a minimum energy of 0.0001 GeV, which corresponds to the energy threshold of the detector;
3. Any hits at positions before the first SciFi plane were removed, as well as any hits located between the SciFi planes;
4. Tracks with a negative momentum were excluded.

The above cuts were mainly applied to make the data as realistic as possible. It should be noted that those cuts only eliminated a relatively small share of the events, which means that DS1 hardly decreased in size. Since the size and complexity of data are major contributors to increasing the time required to train a CNN, one more cut was applied to the dataset: all events with less than five hits were removed. The justification for this cut is as follows. By nature, a CNN learns characteristic features of images in order to assign a value to them. Since events with less than five hits are unlikely to contain many distinctive features, they can be considered as a form of noise that is fed to the CNN. Although this is not necessarily a bad thing, the benefit of speeding up the training process is a priority in this research project, considering the limited time available. After applying the cuts described above, the data was saved in batches in order to ease reading of the data at a later stage.

2.1.3 Reducing dimensions

The original input resolution of the images is $210 \times 254 \times 4$ pixels. The first two elements correspond to the x and y dimensions of the individual SciFi planes, respectively, and the last element refers to the number of planes in the simulation. Thus, the total input image is essentially a 3-dimensional structure with four layers, where each layer contains the x, y coordinates of the hits on a certain plane. Due to the limitations of the GPU and time restrictions of the project, training images of this size would take too long. One possible solution to this problem is reducing the input resolution, but this would remove a lot of valuable information, especially from regions with a high hit density. Since most of the hits are clustered around the barycentre of the event, a more feasible option is cutting the nearly empty space around it. To this end, the barycentre of each event was calculated by computing the mean $[x, y]$ coordinate of the first two planes that contained more than two hits. The path between those two coordinates was then extrapolated to compute the barycentre of the two remaining planes. Next, the x and y dimensions around each barycentre containing 3σ ($\approx 99.7\%$) of the total hits in the corresponding plane were calculated for each event, and the averages of those dimensions were used to create the reduced images. This resulted in an image size of $70 \times 72 \times 4$ pixels.

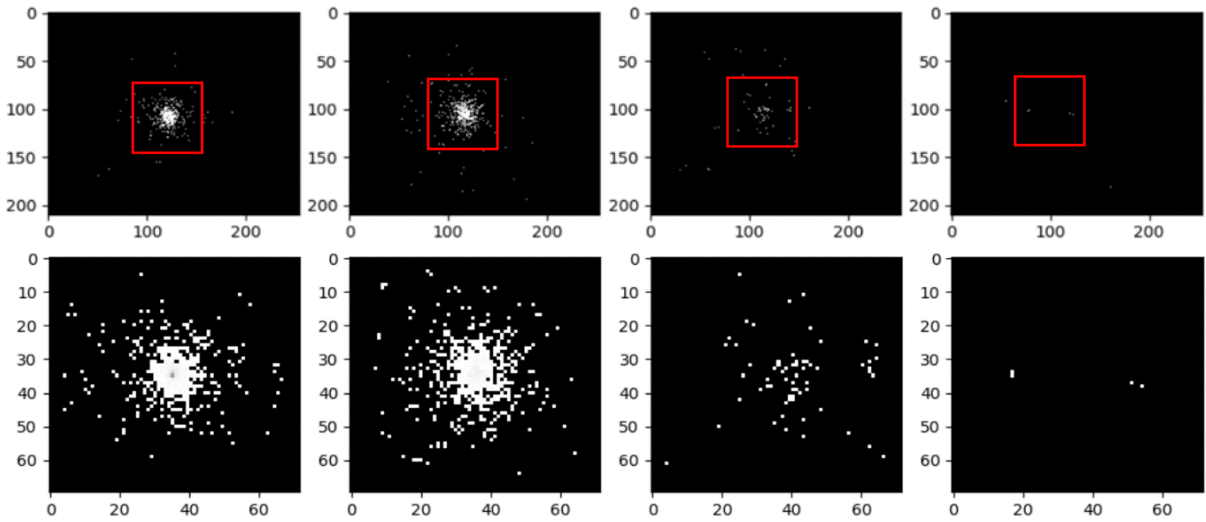


Figure 6: An example image of an event with the original (top) and reduced (bottom) dimensions, where every subplot of each row corresponds to SciFi plane numbers 1–4, counting from left to right. The red marking marks the area of the images that will be left after reducing the dimensions.

Figure 6 shows an example of an event image before and after reducing the image dimensions. The squares on the unreduced images mark the area centred around the barycentre that will be left after reducing the dimension. A closer look shows that the barycentre of the event shifts slightly to the left throughout the four planes. The corresponding reduced images of each plane show that most hits are indeed conserved. To verify that this was the case for the majority of the events, Figure 7 shows the percentage of events lost in the reduced dimensions for 100 randomly selected events, which amounts to a loss of 4.60% on average.

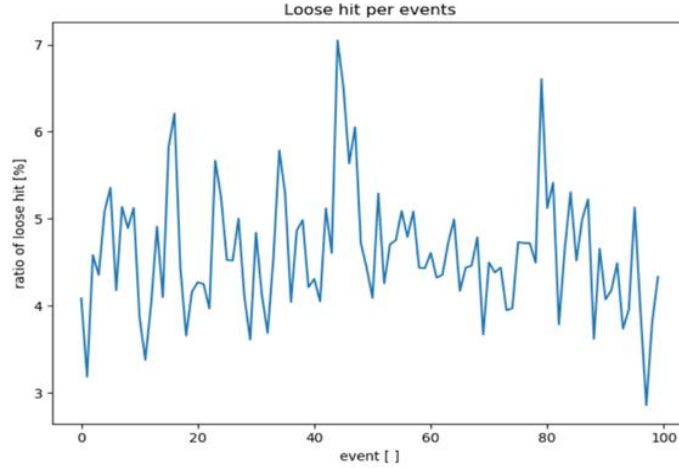


Figure 7: Percentage of lost hits for 100 randomly selected events. The average loss is 4.60%.

2.1.4 Structure of the energy reconstruction algorithm

A schematic overview of the CNN for the energy reconstruction is shown in Figure 8. The initial input consists of a so-called ‘batch’ of images, defined as a set of n images that are used to calculate the gradient descent of one training epoch. Those images, with a size of $n_{batch} \times n_x \times n_y \times n_z$ pixels, are led through four convolutional blocks containing several layers (Figure 8). Each convolutional block requires a specified input- and output size of the z dimension, which corresponds to the batch size. Additionally, the output size of one block must match the input size of the subsequent block.

A block starts with a CoordConv layer in order to discard translational invariance [21]. The set of layers inside the Pool section is an optional extension to the block. It contains a MaxPool layer with a kernel size of 2×2 and a stride of 1, which reduces the image size and consequently lowers computational cost. Next, the output is put through a BatchNorm layer to prevent weight values from saturating too much. This is known to destabilize the network and lead to less accurate predictions as a result [31]. Lastly, a ReLU layer is implemented to eliminate any negative pixel values and put them to 0 instead. Those layers finally lead to an output image of a size $n_{batch} \times n'_x \times n'_y \times n'_z$, depending on whether the Pool layer was invoked. For the energy predictions, all blocks included the Pool section. The final output from the blocks is converted from a 4-dimensional to a 2-dimensional vector of size $n_{batch} \times [n_x \cdot n_y \cdot n_z]$ in the fully connected Flatten layer. This output is subsequently put through a Linear layer that maps the combination of elements to a single output value.

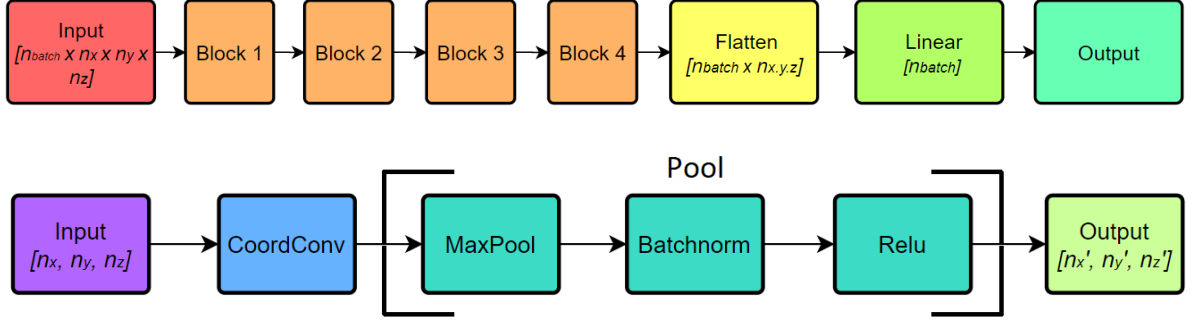


Figure 8: A schematic overview of the overall architecture of the CNN (top) and the structure of a single block (bottom). The latter contains a convolutional layer and an optional set of layers containing a MaxPool, BatchNorm and ReLU layer.

2.1.5 Bias minimization

The performance of the CNN using three different loss functions was investigated. Those loss functions were L1 loss, SmoothL1 loss and MSE (Section 1.3). Additionally, the performance of a normalized version of those loss functions was investigated. To this end, L1 was divided by y_n and SmoothL1 and MSE were divided by the squared term $(y_n)^2$. All loss functions were trained under the same conditions. Training was done with 150 000 events divided into batches of 150 events, and a standard training comprised 40 epochs. Regarding the structure of the CNN, the Pool layer was invoked for each of the four blocks. The learning rate was set at 10^{-3} and the weight decay at 10^{-2} , following the results of extensive research into those parameters by De Bryas [10]. Finally, the true energy of the input value was divided by 400 following that the full energy range spanned from 200-400 GeV. This was done to reduce the spread of the input data, which stabilizes the network with better performance as a result [31].

2.2 Classification

2.2.1 Dataset

Two new types of datasets to train and test the classification algorithm were created using the Genie neutrino interaction generator from the FairROOT software framework, as described in Section 2.1.1. The first type (DSI) solely contained events initiated by Charged Current Deep Inelastic Scattering (CCDIS), whereas the second type (DSE) exclusively contained showers induced by Neutrino Electron Elastic (NuEElastic) scattering events[28]. For each type, two datasets of 200 000 events were generated for proper training, as well as a smaller set with 1000 events for testing the classification algorithm in its early stages. The energy range of the events was between 100 - 5000 GeV. In contrast to DS1, the new datasets were based on the latest version of the actual SND@LHC dimensions, in which the planes have a width of $d_x = 41.0$ cm, a height of $d_y = 41.0$ cm, a thickness of $d_z = 5.03$ cm and are separated by emulsion bricks with a thickness of 5.37 cm [9].

2.2.2 Pre-processing

The data was pre-processed in a similar manner as DS1, although a few modifications were made due to the different nature of the events. The following steps were taken in the process:

1. Only events initiated by an electron neutrino, thus those with a mother ID of -1 and PDG code of 12, were selected;
2. Events were required to have a minimum energy of 0.0001 GeV, which is the minimum energy threshold of the detector;
3. Any hits at positions before the first SciFi plane were removed, as well as any hits located between the SciFi planes;
4. The events were labelled according to their type: -1 represented inelastic events and elastic events were labelled as 1.

Several other steps of pre-processing were changed as a result of differences in the data generation. First of all, the barycentre of DSI and DSE events was not centred in the middle of the plane, but rather to the left-hand side (Figure 9). This is a consequence of the slight off-axis placement of the SND@LHC with respect to the beam from the interaction point, which propagates in a straight line as neutrinos are not bent by the LHC magnets [9]. To make those events compatible with the image production process of the CNN, the y coordinate of each hit was translated down by 28.5 cm and every x coordinate to the left by 35.5 cm. A consequence of the non-homogenous coordinate distribution of the hits was that most events had their barycentre located on the lower right area of the planes. Furthermore, the data contained a considerable number of hits associated with particles with a negative and considerably high p_z . Considering that the data already contained relatively few high-energy events, those back-scattering hits were not removed from the dataset, in contrast to DS1. Lastly, a large share of the generated events did not interact with the SciFi planes. Consequently, only 28.7% of DSI and 11.7% of DSE events were left after reading the SciFi response data. On top of those losses, 34% of the events contained five hits or less. Although it seemed unlikely that events with such a few number of hits could be labelled correctly, the minimum number of hits required for the CNN to predict a sensible outcome was unknown. For this reason, all events with at least 1 hit were preserved to avoid having to train with too few events.

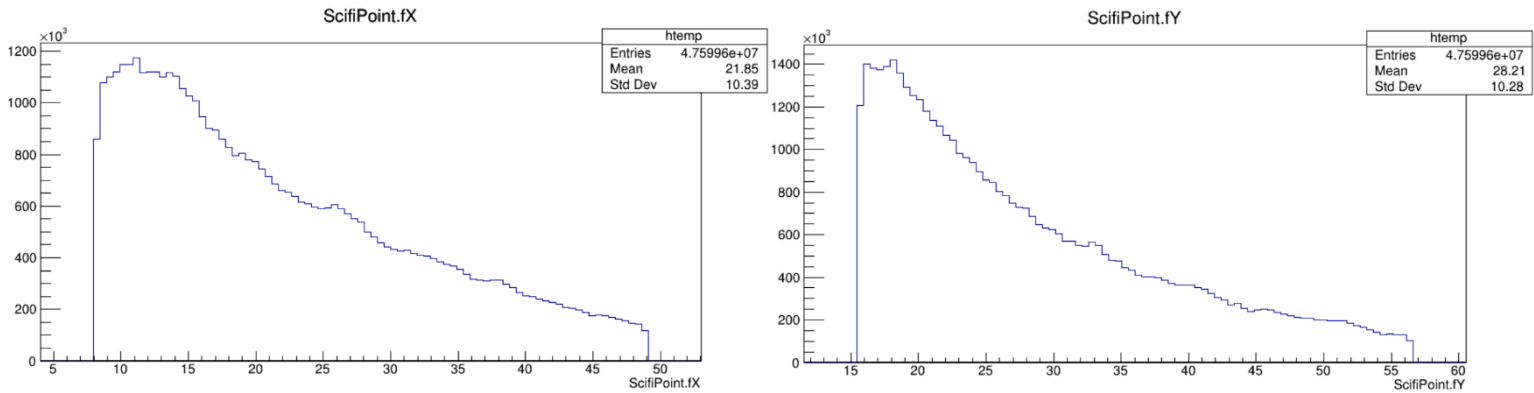


Figure 9: Example of the x and y coordinate distribution of hits in DSI and DSE for 200 000 events. The majority of the hits is clearly located in the corner of the planes, which is a consequence of the slight off-axis placement of the detector with respect to the beamline propagating in a straight line from the LHC interaction point.

The input images of the DSI and DSI events had a resolution of $6 \times 208 \times 172$ pixels. Training with such large images would highly increase the time needed for training the CNN, which was not feasible given the limited time span of the project. However, it was not possible to reduce the image dimensions using the barycentre method as described in the previous section due to the large number of events that contained only very few hits, or had their barycentre located on the edge of the image or event beyond. The first case led to (nearly) empty input images, as few spread out hits may result in a central barycentre that is too far away from any of the hits to preserve them (Figure 10). The latter situation led to images containing only one part of the event that was not centred correctly in the first place, as a result of a miscalculation of the barycentre (Figure 11). Both cases result in images that have very different characteristics from the actual events, which would most likely affect the performance of the CNN.

Given the implications of reducing the image dimensions of DSE and DSI, it was decided to train the classification algorithm without reducing the dimensions. As discussed before, this implied that the images contain a vast amount of empty space that may slow down the training process albeit not containing much useful information. Therefore, the standard number of epochs used for training was set at 20, rather than 40 as was done for the energy reconstruction.

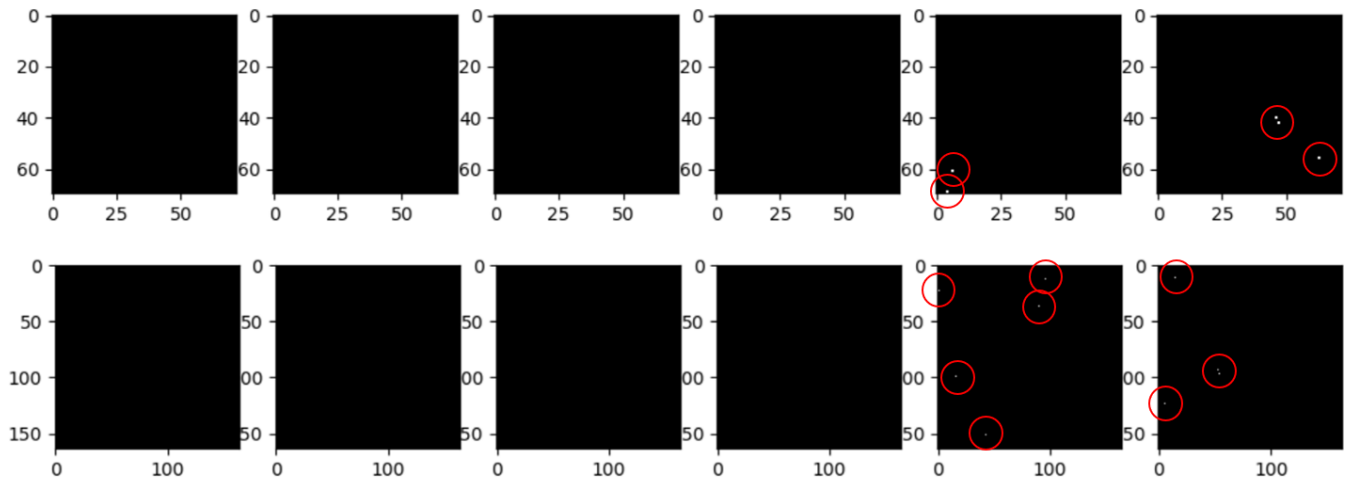


Figure 10: Reduced images (top) produced from an unreduced event with few hits (bottom). The black squares represent plane numbers 1-6 from left to right. The red circles point out the individual hits, which are hardly visible due to the many pixels that comprise the images. Evidently, relative many hits are lost after reducing the image size due to the large spread of the individual hits in the original image.

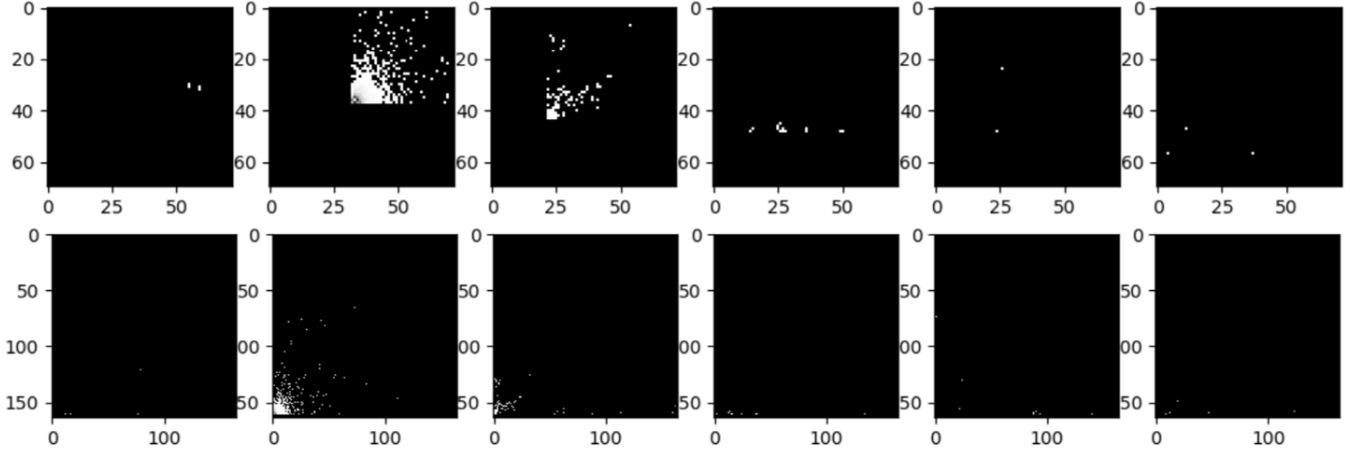


Figure 11: Reduced images (top) from an unreduced event located to the far side of the plane (bottom). The individual squares represent the planes 1 - 6 (l-r) containing the hits associated with the charged particles passing through them. The true barycentre of the event lies in the lower-left corner and is well-centred in the second plane of the reduced image, but unrightfully shifted to the right from the second plane onwards. Moreover, many hits are lost and a lot of empty space is present around the artificial edges of the event in the unreduced images.

2.2.3 Structure of the classification algorithm

The structure of the classification algorithm was largely based on the structure of the energy reconstruction algorithm. A schematic overview of the architecture of the classification algorithm is shown in Figure 12. The input consisted of images with a dimension of $6 \times 208 \times 172$ pixels. Those images were led through a variable number of blocks, of which the architecture is shown in Figure 12. The final output of the blocks had an image of a size $n_{batch} \times n'_x \times n'_y \times n'_z$, depending on the number of blocks and whether or not the Pool layer was invoked. The Flatten layer compressed the last three dimensions into one to obtain a vector of shape $n_{batch} \times [n_x \cdot n_y \cdot n_z]$. This vector was passed on to the Linear layer, where the individual elements were mapped to a single output value. Finally, this value was translated to a value within the range $[-1, 1]$ to directly represent the probability with which the data would be assigned to one of the labels, which, to recall, were -1 for inelastic events and 1 for elastic ones. The reason for the -1 label rather than 0, or any other arbitrary number, was to ensure symmetry in the data and facilitate equal treatment for both types of events in the algorithm. This tends to stabilize the CNN, which often leads to better results [31].

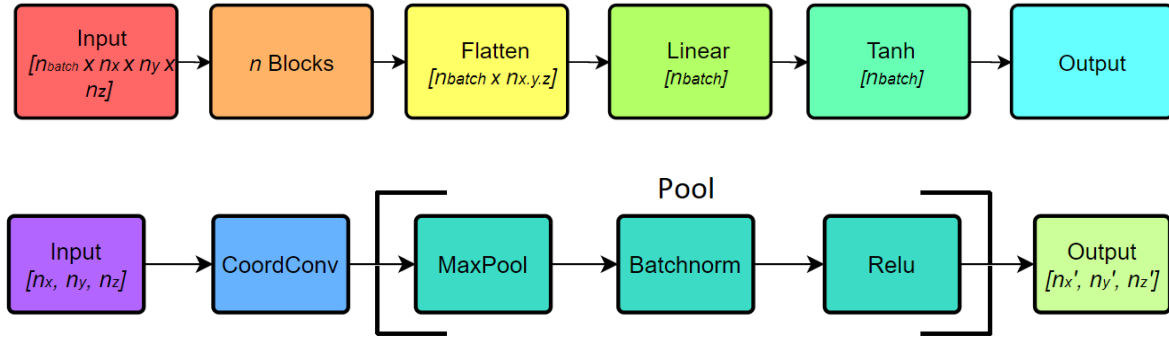


Figure 12: The overall structure of the classifier (top) is very similar to that of the algorithm for energy reconstruction. The Tanh layer was added to compress the input within a range of $[-1, 1]$. Training was done with various numbers of blocks, hence the n . The structure of a single block (bottom) has the same architecture is the same as the blocks used for the algorithm for energy reconstruction. The Pool layer is optional.

2.2.4 Hyperparameters

The MSE loss was used for the classifier, as this function produced the most accurate results for the energy reconstruction. Moreover, this loss function treats positive and negative output values equally as it takes the square of the difference between the true and predicted label. This avoids bias toward either side of the output range. The weight decay was set at 10^{-2} and the learning rate at 10^{-3} . As mentioned before, the standard number of training epochs was 20 due to the large size of the input images, which increased the training time significantly. The batch size was chosen to be 150, which optimized the balance between training speed and low CPU and memory usage, where the latter were crucial to keep the training from being killed by the server.

2.3 Computing resources

The first part of the energy reconstruction was done on the EPFL server, which uses an Intel(R) Xeon(R) CPU E5-2680. However, the limited available memory of this CPU prevented multiple training sessions to be executed simultaneously. Given the restricted time span of the project, the remainder of the research was done on the Nikhef cluster. Training was done on the GeForce GTX 1080 GPU and pre-processing was mainly done on the Tesla V100 GPU. The latter had a lower tolerance for processes using a large percentage of CPU but was well-suited for less intense but time-consuming processes.

3. Results

3.1 Bias minimization

3.1.1 Loss function analysis

To evaluate the overall training process, the training loss epoch was examined for each of the loss functions (Figure 13). To recall, the training loss is a measure of the error between the actual and predicted values for the energy (Section 1.3). An important feature of the graphs is the rapid decline of the training loss at first, which is quickly followed by a stabilisation. This generally an indication that the algorithm is quickly learning dominant features in the beginning, and continues to learn more refined characteristics afterwards. However, the latter can lead to overfitting, which is why a second measure is needed to verify this is not the case. The validation loss fulfils this purpose, as it measures how well the network performs on unseen data at the end of an epoch. There is clearly a lot of variation in the validation loss for the first 10 epochs, but it stabilizes fairly well after epoch 20. As its value is not increasing during the training epochs, the network appears to be learning subtle but useful features until the end of the training.

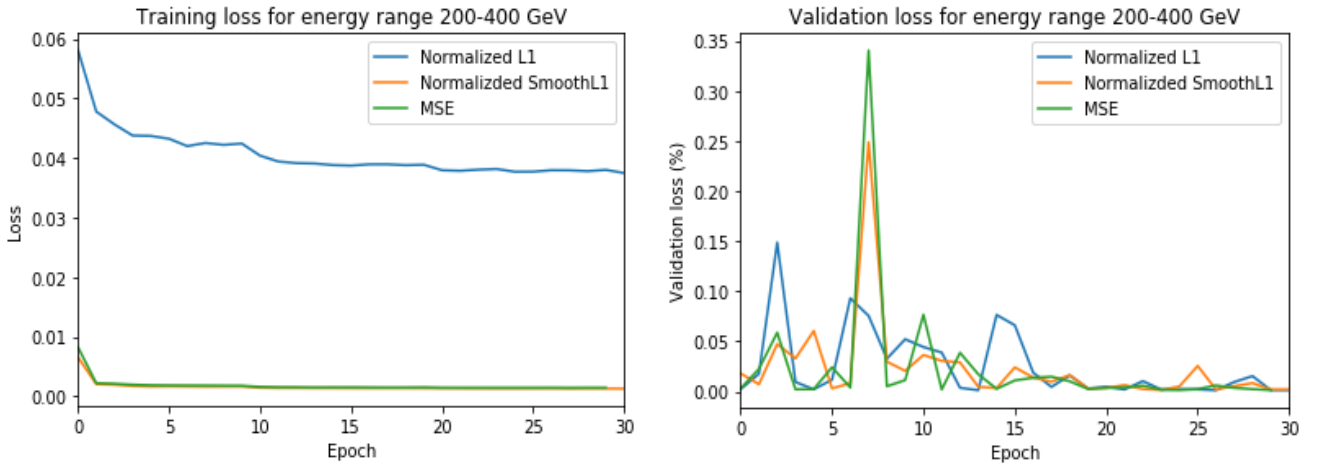
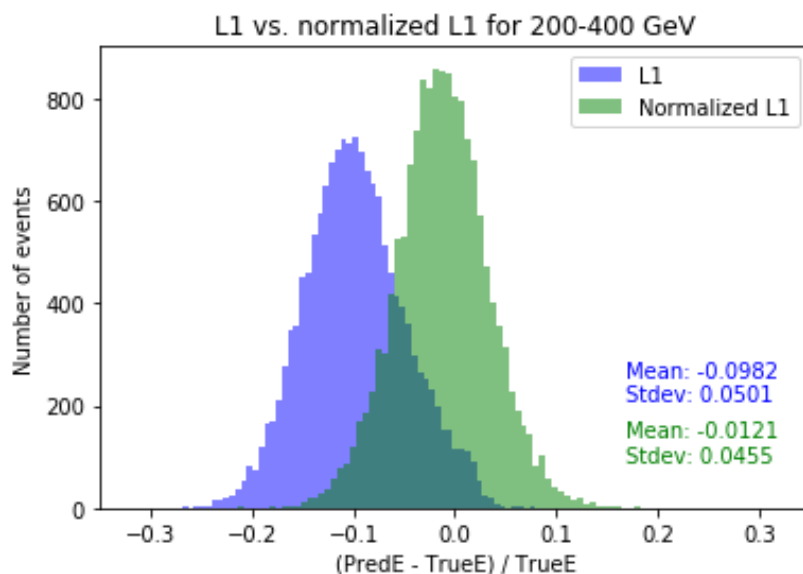


Figure 13: Training loss and validation loss for the three different loss functions. Normalized SmoothL1 and MSE loss show a comparable performance for both variables. The Normalized L1 loss follows similar trends, but with a higher absolute value for its training loss.

To analyse the network performance for each loss function, the predicted energy values from the test batch were subtracted from the actual energy and divided by the latter. This gives the relative error of the prediction with respect to the true value. Ideally, one observes a mean value of 0, accompanied by a minimal spread around this mean. In this context, the mean represents the bias, which is the average percentage with which shower energies are over- or underestimated. Figure 14 shows these results for the normalized and unnormalized version of the three different loss functions. An immediate observation is that the unnormalized MSE loss clearly yields the best performance. Its bias is merely 0.07% which is very low, especially compared to the best performances of L1 and SmoothL1. However, the MSE standard of 3.57% is rather similar to that of the other two functions, which makes

it relatively high with respect to its bias. Nevertheless, this result means that the algorithm has to potential to providing an unbiased too to predict shower energies.

An interesting result is the effect of normalization. Purely judging by the loss functions, the effect of normalizing a loss function should be avoiding unequal loss calculations due absolute differences prediction values, and give relative a relative loss instead. Intuitively, this would stabilize the network, which has shown to yield better overall performance [31]. Whereas L1 and SmoothL1 loss seem to follow this logic, MSE shows a very contrasting behaviour. For L1 and SmoothL1, the bias significantly decreases by a factor 2.15 and 4.14 respectively when the functions are normalized, and the standard deviation only decreases slightly. Although the standard deviation hardly changes for MSE either, the bias actually becomes 38.4 times larger after normalization. At this point there is no clear explanation for this phenomenon. However, this need not necessarily raise concern. After all, machine learning algorithms are sometimes referred to as ‘black boxes’ for a reason. Although all individual components of the algorithm are accessible and open to inspection, the overall behaviour arises from the interaction between those components. This feature is the great strength of machine learning algorithms, but it is also what can make their performance relatively unpredictable once they reach a certain level of complexity [32]. Nonetheless, further investigation into the effect of loss function normalization may yield useful insights and is certainly recommended.



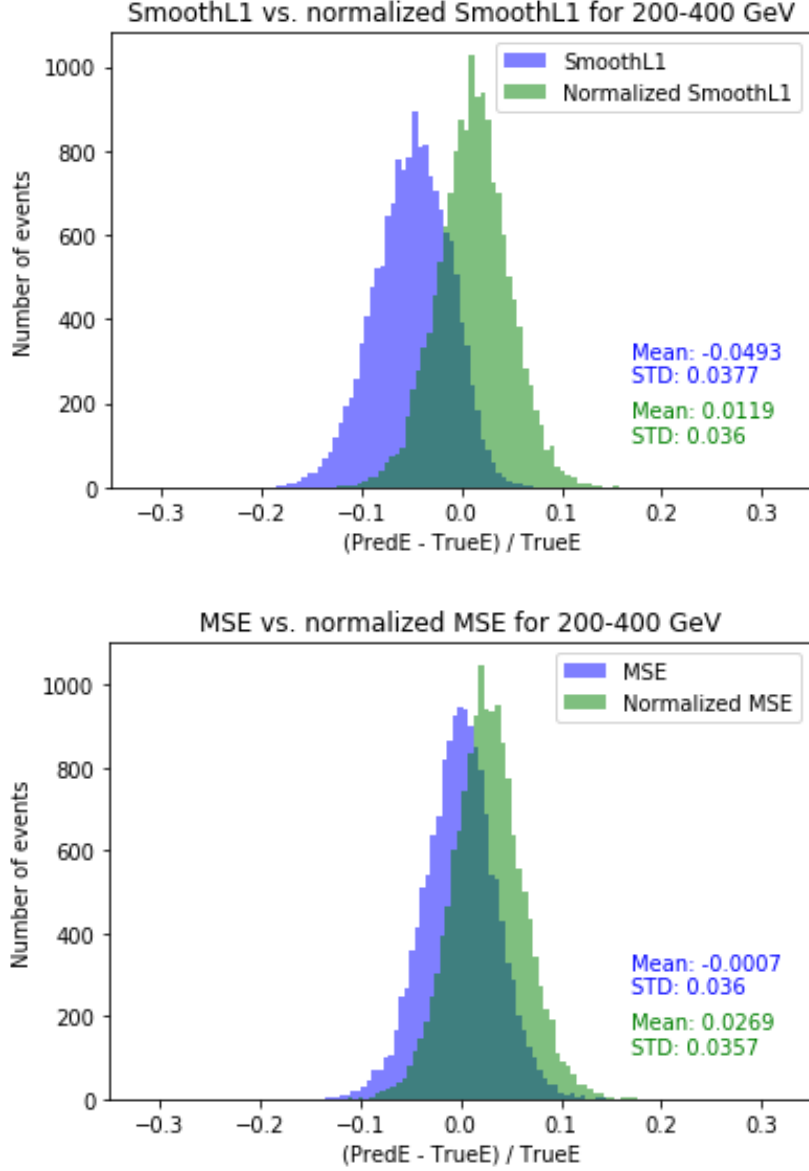


Figure 14: Histograms of the relative error of the predicted energy values for the normalized and unnormalized version of different loss functions. The relative error was calculated by subtracting the predicted energy from the true energy value and dividing this by the true energy.

3.2 Classification

3.2.1 Image resolution

The input resolution of the input images was $6 \times 208 \times 172$ pixels. Since the detector planes are squared planes of 41.0×41.0 cm, the actual useful information of the hits is only located in a square of $6 \times 164 \times 164$ pixels. This effectively adds a strip of black pixels on the sides of the input image, which might have slightly decreased the pace of learning. Overall, since the black strips are the same for every input image, it seems fair to assume that the CNN learns to discard this information quickly. However, proper training with appropriate dimensions is necessary before any conclusions can be drawn. This is especially important for further research into the effect of changing the input resolution of the

image. At this point, the input images would suggest a detector resolution of $410\,000\,\mu\text{m} / 164 = 2500\,\mu\text{m}$ per fibre, while in reality the resolution of the SND@LHC will be $50\,\mu\text{m}$ per fibre [9].

3.2.2 CNN structure analysis

To get an impression of the overall performance of the classification algorithm, the training loss and validation loss were plotted as shown in Figure 15. The training loss decreases steeply during the first 5 epochs and starts to stabilize somewhat afterwards. This is especially true for the case of the structures with 4 and 5 blocks; the loss of structure with 6 blocks keeps decreasing more significantly for subsequent epochs. The validation loss does stabilize more or less after 12 epochs. However, a slight upward trend is already visible during several epochs for the structure with 6 blocks, which might indicate that the network is coming closer to the point where it starts overfitting the data. It might be interesting to verify whether this is the case by training for more epochs. Unfortunately, this could not be realized due to time restrictions for running high-intensity processes on the server.

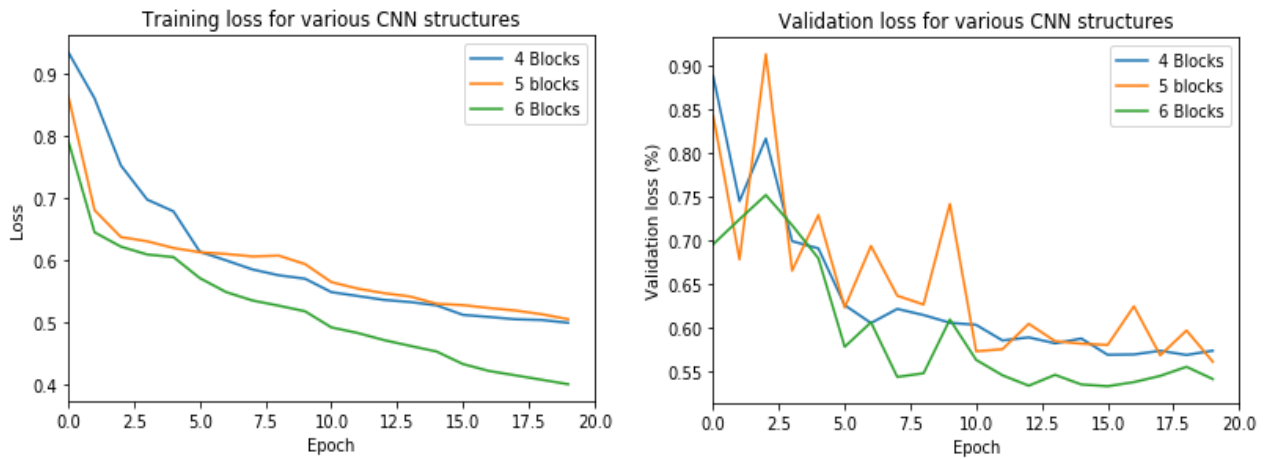


Figure 15: The training loss and validation loss for different structures of the classifier. The trends and actual values of both parameters are very comparable and show a clearly decreasing trend, that flattens after several epochs.

To evaluate the accuracy of the classification algorithm, the receiver operating characteristic (ROC) curve was plotted for each of the CNN structures. A ROC curve is a measure of the performance of a binary classification algorithm that plots the false positive rate (FPR) versus the true positive rate (TPR) at all possible classification thresholds. The classification threshold refers to the predicted label value at which the positive class is chosen over the negative class. In context of this research, the threshold classification is the predicted value at which the algorithm decides to assign a label of 1 to an event, thus classifying it as an elastic scattering event. The area under the curve (AUC) is a direct measure of the accuracy of the classifier. To illustrate, an AUC of 0.5 means that the algorithm predicts labels with 50% accuracy, which implies that the labels are randomly assigned. On the contrary, an AUC of 1 indicates that the labels are assigned with 100% accuracy. Hence, the ultimate goal of the classification algorithm is to maximize its AUC.

The ROC for each of the CNN structures after 20 epochs is shown in Figure 16. Their AUC values vary closely around 0.88, which shows that the classifier correctly identifies the majority of the events. A closer look reveals a slight increase of the AUC as the number of blocks increases. Taking into consideration the decreasing loss and validation loss discussed as before, this might indicate that the added complexity of the network allows for the distinction of small, detailed features that are too refined for the CNN to learn with less blocks.

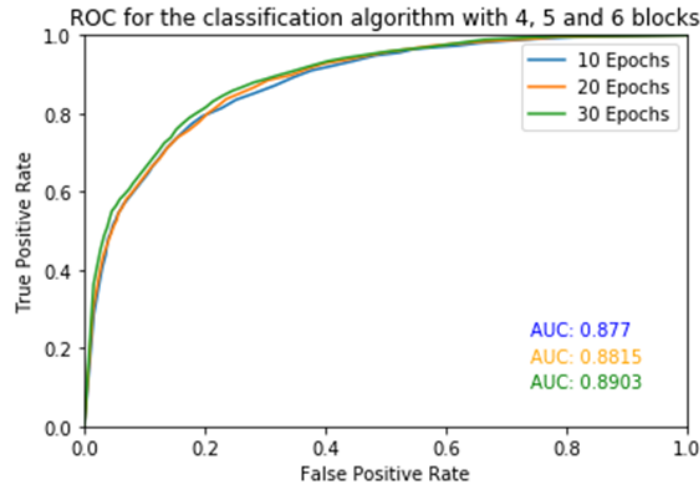
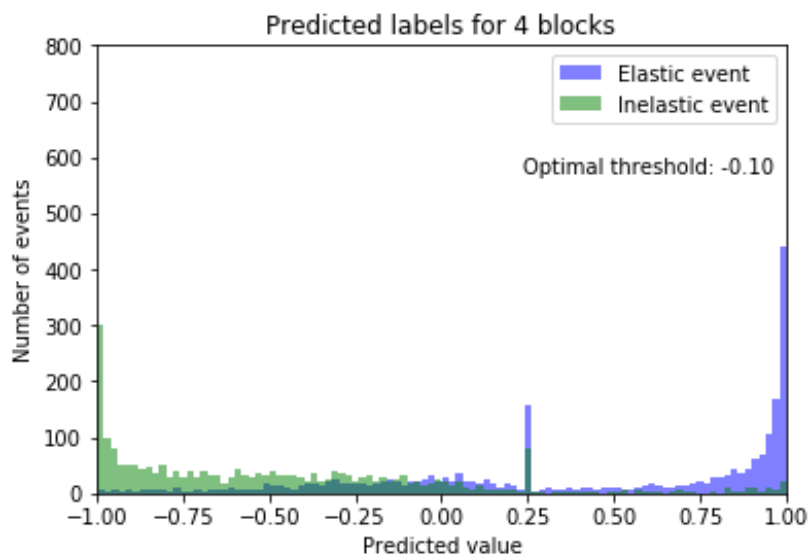


Figure 16: ROC curves for the classifier with 4, 5 and 6 blocks, respectively. The AUC increases slightly as more blocks are added.

The histograms in Figure 17 provide concrete information about the labels assigned to the events. To recall, elastic events were given a true label of 1 and inelastic events a label of -1, and the optimal threshold refers to the cut-off value that maximizes the number of correct predictions. This value was calculated by maximizing the G-mean [33]. Its value is slightly below zero for the first two CNN structures, while the 6-block structure has an optimal threshold of exactly 0. This shows that the network incorrectly predicts roughly as many elastic as inelastic events. The predictions are overall fairly accurate, and the evident peaks close to -1 and 1 indicate that the algorithm predicts many events with considerable certainty.



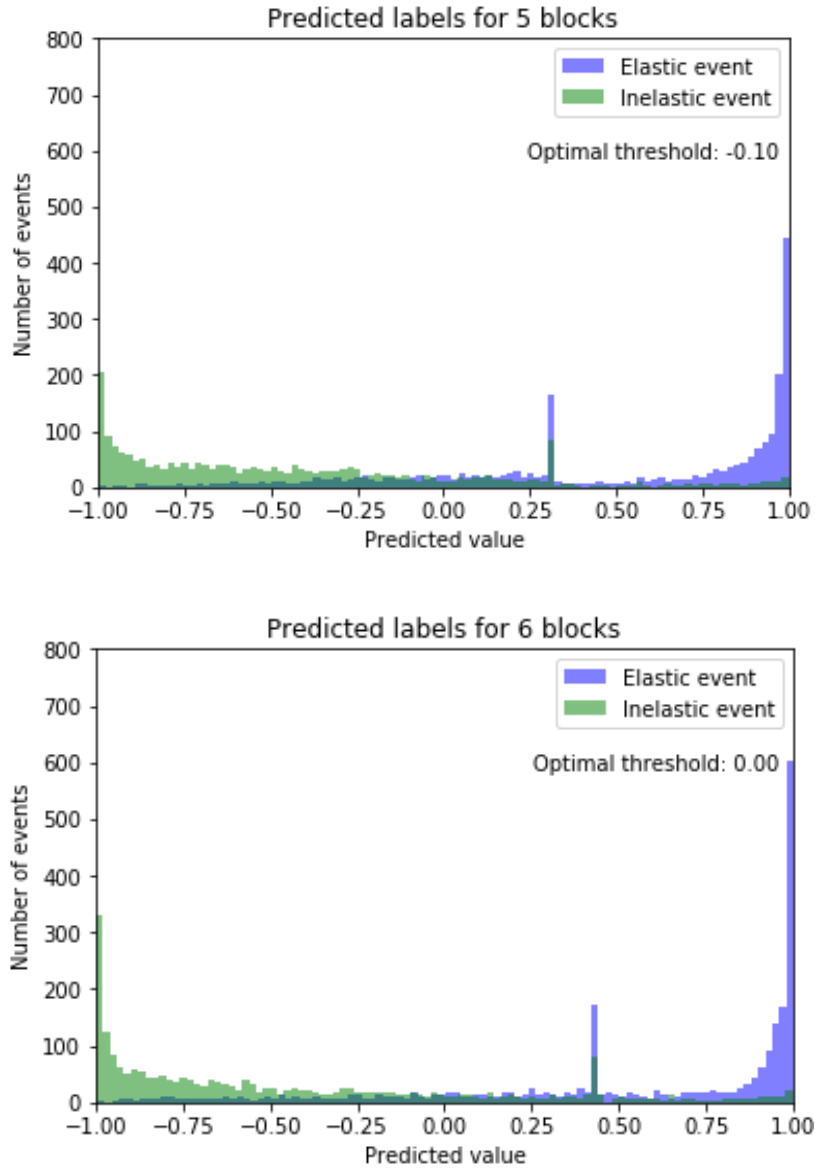


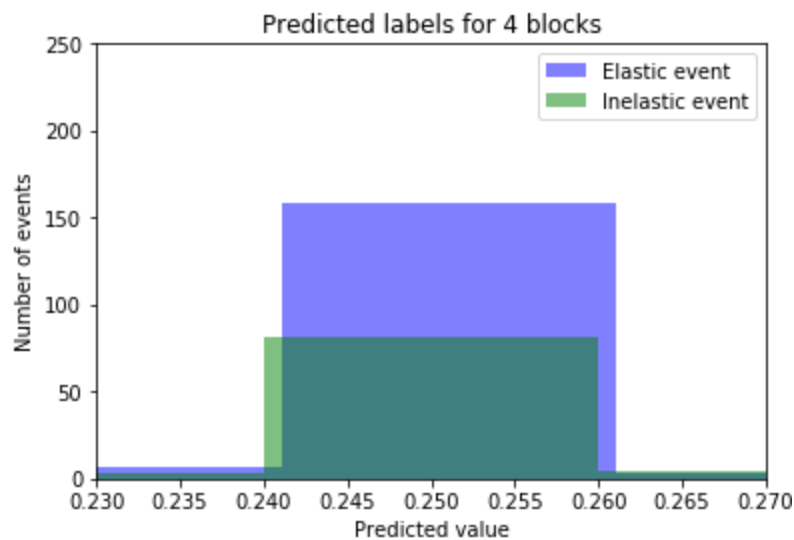
Figure 17: Histograms of the predicted values for each event for CNN structures with 4, 5 and 6 blocks. Overall, a clear division can be seen between predicted labels of elastic and inelastic events. An exceptional feature is the sudden peak between 0.25 and 0.50, which seems to relate to events with that left very few hits in the SciFi planes.

One of the most interesting features is the prominent peak that occurs between 0.25 and 0.50 for each CNN structure. Those peaks can be seen more clearly in Figure 18. The ratio of elastic/inelastic events seems very similar for each peak, whereas their locations shift to the right as the number of blocks increases. Further analysis of the events associated with those peaks showed that the vast majority contained less than 5 hits. As the same events were used for testing and training the different CNN structures, this is a plausible explanation for why the elastic/inelastic ratio of the peak does not change. Moreover, it might also explain the specific location of the peak. After all, events with very few hits in the detector most likely do not contain any recognizable features that the CNN has learnt, which turns the prediction into a game of chance.

This hypothesis can best be illustrated with an example. The ratio between the number of elastic events with less than five hits, $elastic_{hits < 5}$, and the number of inelastic events with less than five hits $inelastic_{hits < 5}$, within the peculiar peak for the CNN structure with 5 blocks is $\frac{elastic_{hits < 5}}{inelastic_{hits < 5}} = 1.88$. If the prediction of the label is indeed a matter of chance for events with less than five hits, the lowest average error would theoretically be achieved with a standard predicted label of $\frac{1.88}{1 + 1.88} \cdot 2 - 1 = 0.306$ for such events. According to Figure 18, the average prediction label for the events underneath the peak is 0.310, which comes close to the theoretical prediction value that would minimize the loss. The theoretically optimal prediction value for the other two CNN structures and the actual prediction value of their peaks is shown in Table 1. Clearly, the theoretically optimal prediction values of the CNN with 4 and 6 blocks does not come as close to the observed value of their peaks as the CNN with 5 blocks. This might indicate that the labelling of events with less than five hits is not as much a matter of chance for the CNN with those structures. However, too little information is available to draw any conclusions from these observations. Nevertheless, it may be an interesting feature to analyse if further information is required about how the CNN treats events that only leave a few hits in the detector.

Number of blocks	Ratio elastic/inelastic events	Observed average prediction value of the peak	Theoretical optimal prediction value for prediction based on chance
4	1.97	0.250	0.327
5	1.88	0.310	0.306
6	1.85	0.420	0.298

Table 1: The average prediction value of the peaks consisting of events with less than 5 hits, as well as the prediction value that would minimize the error if the prediction was based on chance, given the ratio elastic/inelastic hits underneath the peak. The theoretical optimal value for the CNN with 5 blocks is close to the observed label that was given to the events. This is not necessarily true for the CNN with 4 and 6.



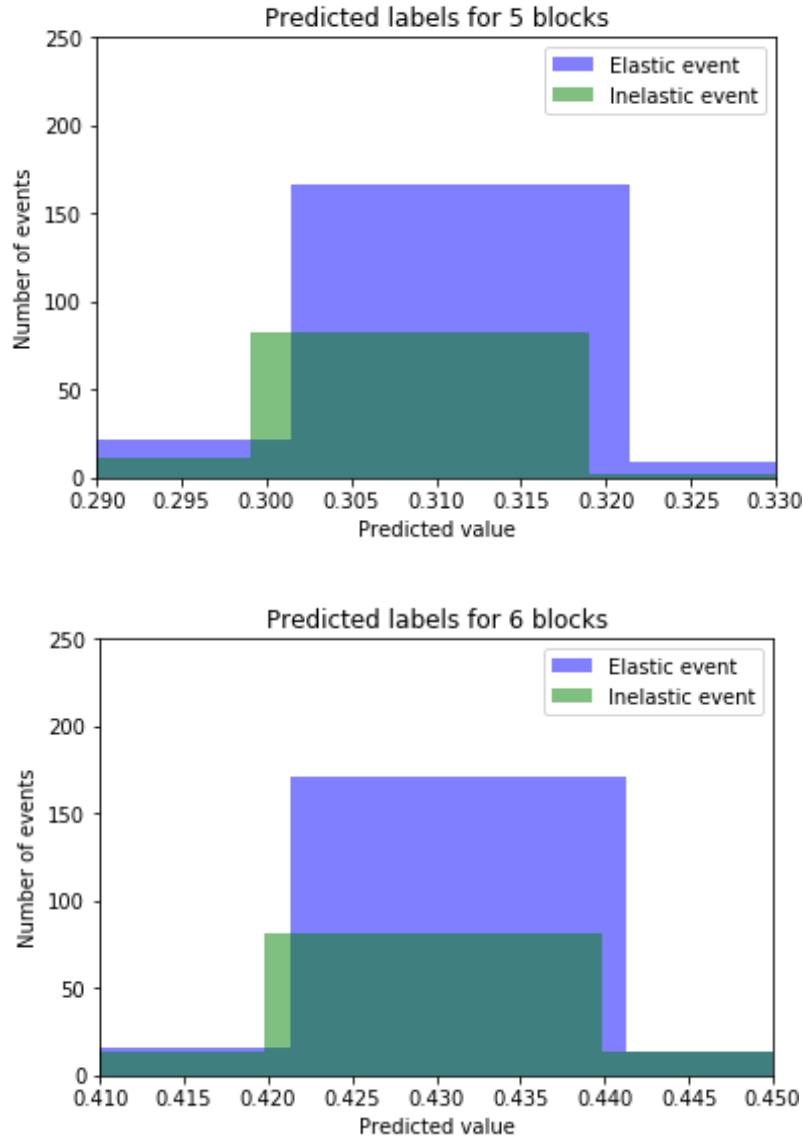


Figure 18: Enlarged view of the peaks seen in the histograms of Figure 17. The height of the peaks, as well as the relative height of the elastic and inelastic peaks, is very similar for all CNN structures. The average prediction value of the peak increases as more blocks are added to the structure.

Another interesting feature of the predictions is the small peak of inelastic events on the far right of the histogram, where estimated labels are between $[0.9, 1.0]$. These inelastic events are apparently labelled as elastic events with quite some certainty. This might indicate that the images of those events contain many elastic features, lack inelastic features, or both. A few representative examples of the elastic and inelastic events with labels between $[0.95, 1.0]$ of all three CNN structures are shown in Figures 19 and 20, as well as a few examples of inelastic events that were given a label between $[-0.95, -1.0]$ in Figure 21.

At first glance, some prominent similarities are visible between the truly elastic (Figure 18) and wrongly labelled inelastic events (Figure 19). Figures 19b and 20c look very similar, apart from being 'mirrored' in the vertical axis. Figures 19c and 20b also look rather similar in their appearance, with half of the main impact visible and a few hits showing in the subsequent plane. Although this is not a very significant result, it does show that the mistakes made by the CNN are not completely random or unjustified. Perhaps the clearest distinction is the plane at which most of the hits are located. The elastic events do not clearly have a plane that contains most of the hits (Figure 20), whereas most signature inelastic events have their hits concentrated in the last three SciFi planes (Figure 21). The inelastic events that are wrongly labelled as elastic events do indeed have the majority of their hits located in the first four planes, which makes it seem plausible that the plane number of the hits is a prominent feature used by the CNN. Although there may be a correlation between the type of event and the average z-location of the hits, this is apparently not an exclusive feature. This example illustrates a limitation of a CoordConv based network. On the one hand, the translational invariance of regular convolutional networks has to be discarded as the location of hits on the plane is a direct consequence of the type of event and its energy, but on the other hand it might lead to spatial dependencies that are not necessarily exclusive features, such as the z-location of the hits [34].

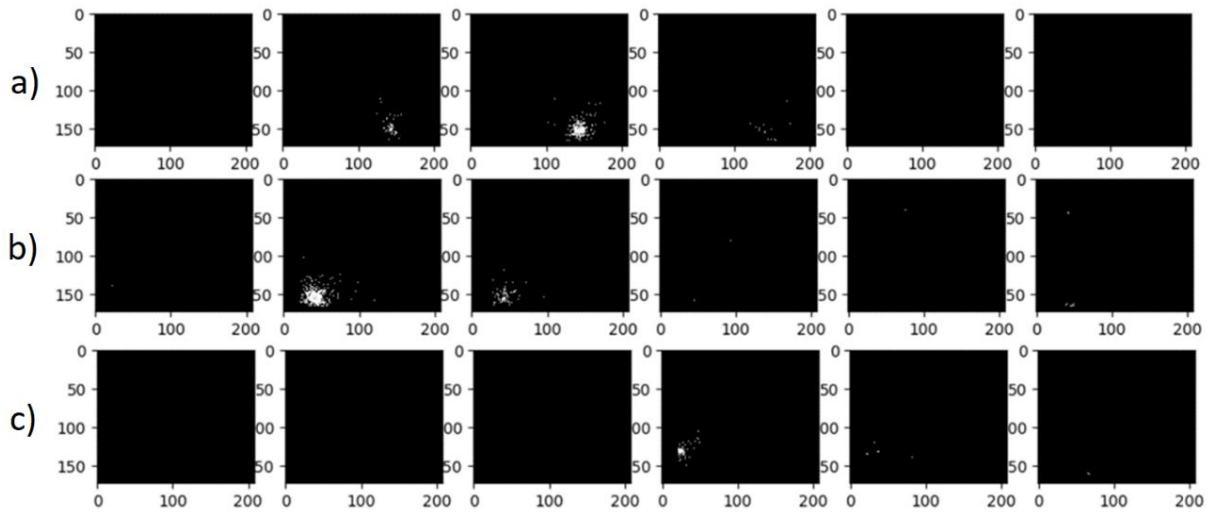


Figure 19: Inelastic events in the with labels in the range $[0.90, 1.0]$. Each row represents a single event and the separate elements of each row represent the planes of the detector from left to right. The axis of the images represent the number of pixels on the axis. The white dots represent the individual hits caused by the shower particles. The hits are mostly found in the middle planes, although this is not a very significant pattern.

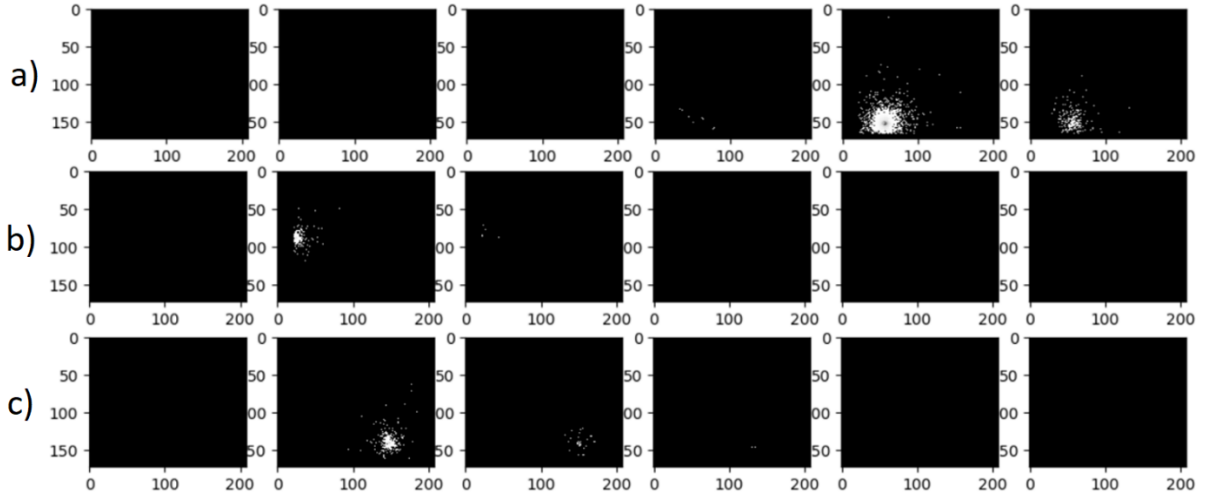


Figure 20: Images of elastic events in the with labels in the range $[0.90, 1.0]$. Each row represents a single event and the individual elements of each row represent the planes of the detector from left to right. The axis of the images represent the number of pixels on the axis. The white dots represent the individual hits caused by the shower particles. At a first glance, there are not many distinct features shared by all events.

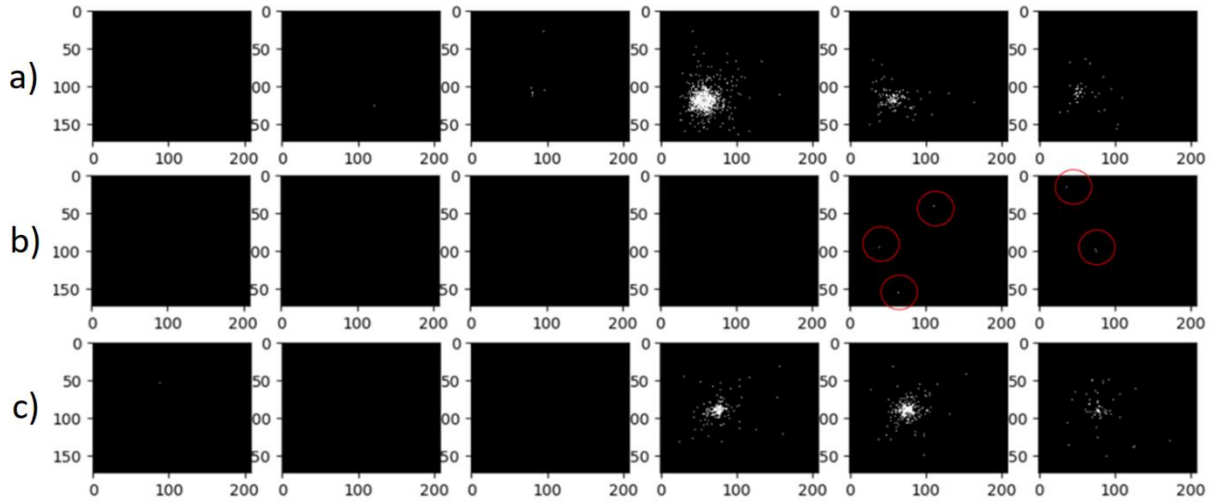
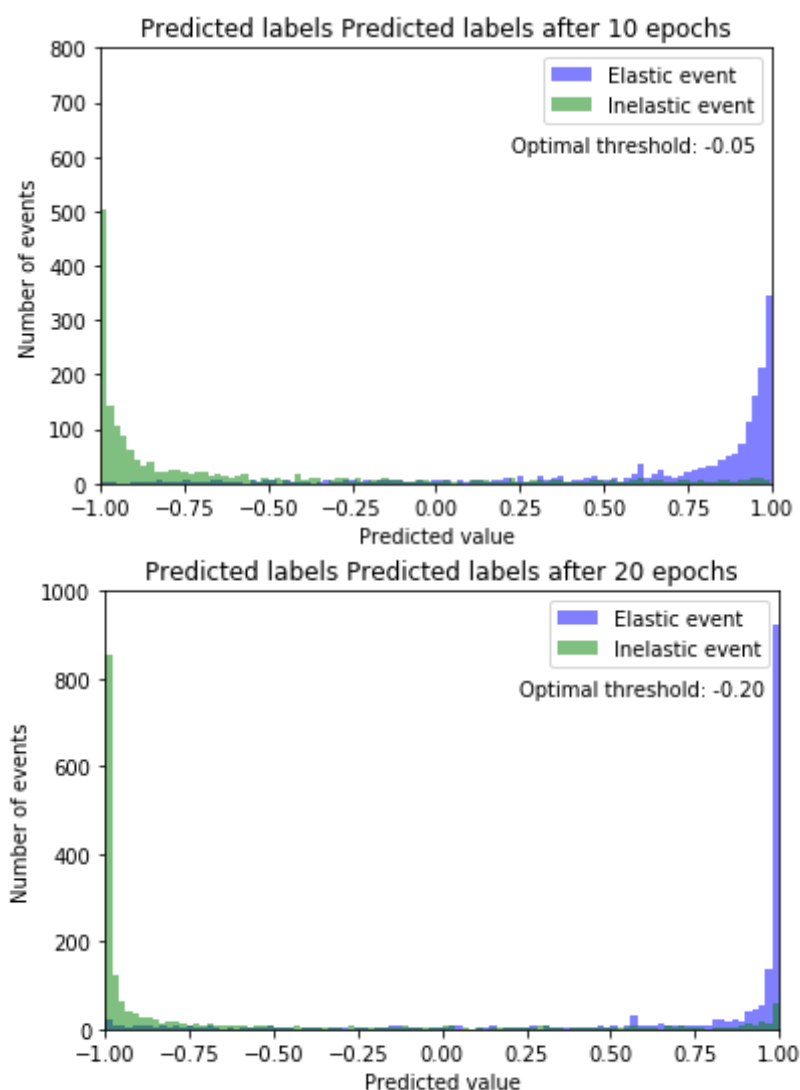


Figure 21: Inelastic events in the range $[-1.0, -0.95]$. Each row represents a single event and the individual elements of each row represent the planes of the detector from left to right. The axis of the images represent the number of pixels on the axis. The white dots represent the individual hits caused by the shower particles. The red circles drawn in the second event point out the location of the individual hits, which are otherwise barely noticeable. Interestingly, all events seem to have hits accumulated in the last three planes.

3.2.3 Increased minimum number of hits

Following the complications caused by events with few hits in the detector, the data was pre-processed with a threshold of minimum 5 hits. The network was trained and tested with the 6-block architecture given the results discussed before. A histogram of the test predictions of the labels after 10, 20 and 30 epochs are shown in Figure 22.

In contrast to the results of training with all events, these histograms only show a very small peculiar peak around 0.25. Further analysis showed that this was almost entirely the result of events containing less than 10 hits. An interesting feature is that the network actually performed best after only 10 epochs. Here, the AUC of the corresponding ROC had a value of 0.934, whereas those values were 0.923 and 0.9161 for the network after 20 and 30 epochs, respectively. This might be explained by the fact that most events containing more than 5 hits contain clear features that are learned quickly by the CNN. Since there is no point in learning more complex features in order to correctly label events with few hits, and thus few significant features, the network starts to overfit quickly. This can be seen more clearly in Figure 23, where the training loss and validation loss are plotted. Although the training loss is decreasing, the validation increases again after epoch 9, which can be interpreted as an indication of overfitting [35].



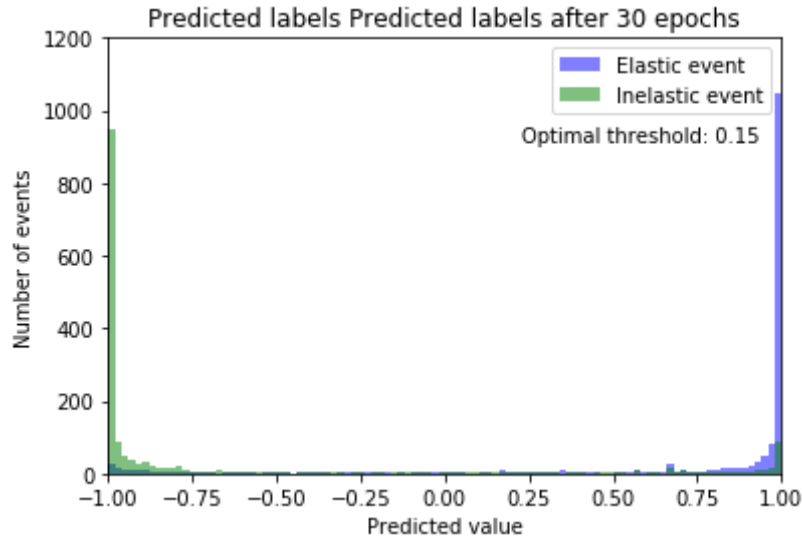


Figure 22: Histograms of the predicted labels after training with data only containing events with 5 hits or more. A CNN with 6 blocks was used and predictions were analysed after 10, 20 and 30 epochs (l-r). Note that the range of the y-axis changes for each histogram as a result of higher peaks near -1 and 1.

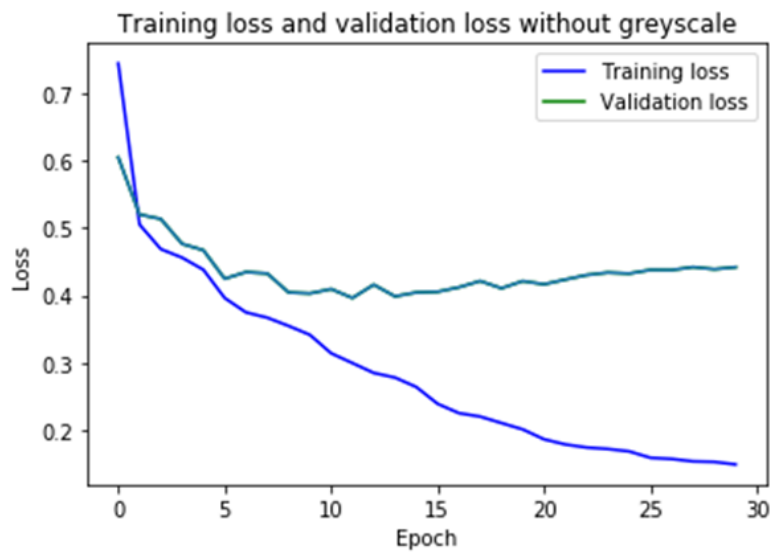


Figure 23: Training loss and validation loss per epoch when training a 6-block CNN structure with data solely containing events with 5 hits or more. The training loss decreases steadily throughout the training, whereas the validation loss already starts increasing around epoch 10.

Another interesting detail is that the number of events with very high and low labels increases with the number of epochs. This means that the CNN is increasingly certain about the label of certain events. However, this comes with an increased number of events that are labelled very wrongly, signified by the small peaks below the large peaks close to -1 and 1. This is not a plausible development as those events cannot easily be filtered away by invoking a certainty threshold.

3.2.4 Elimination of greyscale

The input images were produced in greyscale due to the digitization process of the data. This increased a certain pixel value by 1 whenever hit was located on that pixel. Any value higher than 1 was converted into a shade of grey, which resulted in images that essentially contained information about the number of hits per pixel. However, the SND@LHC will only be able to provide a binary signal even when multiple particles hit a fibre simultaneously. Consequently, the realistic input images only contain black and white pixels, which takes away a fair share of information. To investigate the ability of the algorithm to function without this degree of freedom, the algorithm was tested with a 6-block architecture for 30 epochs. The results are shown in Figure 24.

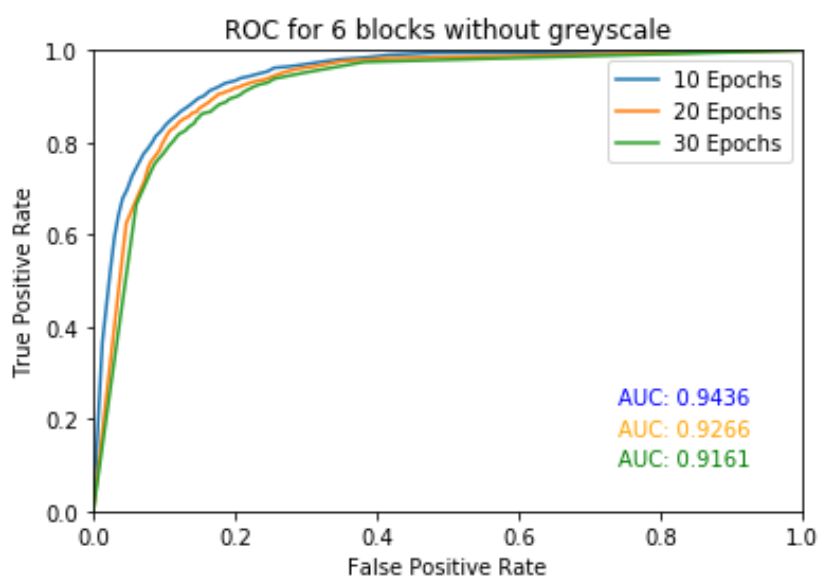


Figure 24: ROC and corresponding AUC values of the algorithm after 10, 20 and 30 training epochs. The AUC clearly decreases as the number of epochs increases.

The AUC values for the network are rather high compared to the previously achieved values when the network was trained with greyscale images. Additionally, the best performance is reached after only 10 epochs. This is in contradiction with the greyscale results, where more longer training led to a slightly improved performance. This could be attributed to the fact that the absence of greyscale greatly reduces the complexity of the classification. As a result, there are fewer characteristic features to learn and the CNN reaches its optimized point more quickly, after which overfitting quickly becomes a risk. Figure 25 shows the training loss and validation loss of the algorithm. Around epoch 10 the validation loss starts to increase despite the decreasing training loss, which indicates that overfitting is indeed a problem.

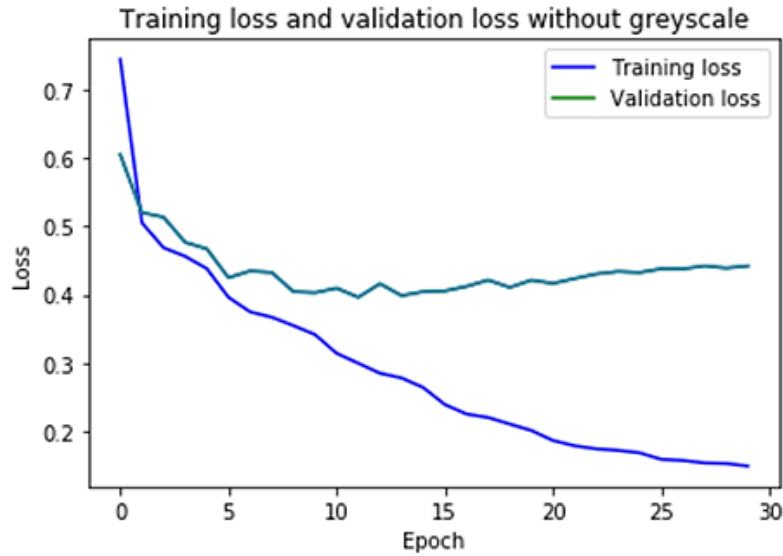


Figure 25: Training loss and validation loss of the network after each epoch. Although the training loss continues to decrease, the validation loss starts to increase again after 10 epochs, which may indicate overfitting.

Summary and Conclusions

The first goal of this research was to minimize the bias of the reconstructed energy by the machine learning algorithm. To this end, several different loss functions were implemented and their effects on the performance were analysed. A bias of only 0.007% with a standard deviation of 2.69% was achieved by using the MSE loss function. This result shows that the algorithm is a viable and accurate tool for shower energy reconstruction, specifically within the energy range for 200-400 GeV. This is the expected energy range at the SND@LHC, so the result of this research brings the algorithm closer to realistic implementation at the experiment.

The second aim was the construction of a classification algorithm to identify elastic and inelastic scattering events. To this end, the effects of several parameters on the performance of the algorithm have been investigated, including the architecture and training length, as well as various consequences of modification of the input data. This research led to a prediction accuracy of 94.36% for the most realistic scenario, in which the images were composed of binary coordinate information rather than using a greyscale. Overall, the biggest limitation of the algorithm was its ability to identify showers that produced only around five hits in the detector. This shows that the algorithm has the potential to be used for classification under realistic circumstances, but its accuracy will need improvement before its implemented at the SND@LHC can be realized.

Outlook

Although the results for the energy reconstruction are rather satisfactory with respect to the bias, the standard deviation is still relatively large. Further research into the structure of the energy reconstruction algorithm may provide insights to reduce this. Additionally, the performance of the current algorithm has not been tested with realistic data. After all, the SciFi planes can only provide alternating x and y coordinate information, which will lead to a considerable number of ghost hits in the detector. This is a serious limitation that needs to be addressed before the algorithm could be used in a realistic scenario. A possible way to approach this problem may be to use time information from the SciFi planes to eliminate as many ghost hits as possible and retraining the algorithm with the new data afterwards.

There are several aspects of the classification algorithm that require further optimization and investigation. First of all, the low resolution that was used for this research may have been a limiting factor for the overall performance of the algorithm. A higher input resolution provides more information to the algorithm without increasing the degrees of freedom, which might improve its predictive power. Although the fact that highly compressed data can still yield fairly accurate predictions can be considered as a promising result, investigating its performance using a higher resolution is strongly recommended to prepare the algorithm for a realistic scenario. Secondly, the data contained many events that were generated with Genie, but did not leave any traces in the detector. The abundance of such events in the data set greatly reduced the amount of useful data left to train with, which may also have been a limiting factor in reaching a higher prediction accuracy. It may therefore be worthwhile to investigate those events in order to avoid big data losses for future data sets. Furthermore, the algorithm struggled with providing correct predictions for events with only a few hits in the SciFi planes. A first approach to solve this problem might be to train the algorithm explicitly with such events, in order to see whether they contain any characteristic features to be learned in the first place. The algorithm also seemed to show a strong dependence on the z -location at which most of the hits were accumulated. However, it was beyond the scope of this research to verify this hypothesis. Nevertheless, it is strongly recommended to investigate how the z -location of the hits, as well as other spatial features such as the location of the barycentre, energy etc., are connected to the characteristic features of the events.

Bibliography

1. Arnison, G., et al., *Experimental observation of isolated large transverse energy electrons with associated missing energy at $s=540$ GeV*. Physics Letters B, 1983. **122**(1): p. 103-116.
2. ATLAS, C., *Observation of a new particle in the search for the Standard Model Higgs boson with the ATLAS detector at the LHC*. Phys.Lett. , 2012. **B716** p. 1-29.
3. Lee, H.M., *Lectures on Physics Beyond the Standard Model*. Lectures at 2019 IBS-CTPU Summer School on Cosmology and Particle Physics, 2019.
4. Aebischer, J., et al., *B-decay discrepancies after Moriond 2019*. arXiv: High Energy Physics - Phenomenology, 2019.
5. Biswas, A., et al., *New physics in b to sll decays with complex Wilson coefficients*. arXiv: High Energy Physics - Phenomenology, 2020.
6. Cerdeno, D., et al., *B anomalies and dark matter: a complex connection*. The European Physical Journal C, 2019. **79**: p. 1-14.
7. Aartsen, M.G., et al., *Measurement of the multi-TeV neutrino interaction cross-section with IceCube using Earth absorption*. Nature, 2017. **551**(7682): p. 596-600.
8. Mertens, S., *Direct Neutrino Mass Experiments*. arXiv: Nuclear Experiment, 2016. **718**: p. 022013.
9. Ahdida, S.C.C., et al., *SND@LHC*. arXiv: Instrumentation and Detectors, 2020.
10. Debryas, P., *Development and commissioning of a machine learning algorithm for real time reconstruction of electromagnetic showers with a scintillating fibres tracker*. 2020.
11. Farrell, S., et al., *Novel deep learning methods for track reconstruction*. arXiv: High Energy Physics - Experiment, 2018.
12. Mali, M., *Seminar 1*, in *Jet Imaging with Deep Neural Networks*. 2019, University of Ljubljana: Ljubljana.
13. Akchurin, N., et al., *Particle identification in the longitudinally unsegmented RD52 calorimeter*. Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment, 2014. **735**: p. 120-129.
14. Shirobokov, S., 2019. https://github.com/shir994/SND_trackers.
15. Alekhin, S., et al., *A facility to search for hidden particles at the CERN SPS: the SHiP physics case*. Reports on Progress in Physics, 2016. **79**: p. 124201.
16. Di Crescenzo, D. *Neutrino Physics and Dark Matter Search with SHiP at CERN*. in *International Conference of New Frontiers in Physics*. 2019.
17. Antoni, T., et al., *Electron, muon, and hadron lateral distributions measured in air showers by the KASCADE experiment*. Astroparticle Physics, 2001. **14**(4): p. 245-260.
18. University, S., *Convolutional Neural Networks for Visual Recognition*, F.F. Li, Editor. 2018, Stanford University: United States.
19. Bishop, C., *Neural Networks For Pattern Recognition*. Vol. 227. 2005.
20. Malik, F. *Understanding Neural Network Neurons*. 2019 [cited 2020].
21. R. Liu, J.L.P.M.F.S.E.F., *An intriguing failing of convolutional neural networks and the CoordConv solution*, in *Conference on Neural Informaion Processing System*. 2018: Montreal.
22. Brownlee, J., *Better Deep Learning*. Vol. 1. 2020.
23. Agostinelli, F., et al. *Learning Activation Functions to Improve Deep Neural Networks*. 2014. arXiv:1412.6830.
24. SHiP, C., *FairShip Code*. 2017. <https://github.com/ShipSoft/FairShip>.

25. Al-Turany, M., *The FairRoot framework*. J. Phys. Conf. Ser. 396, 2012: p. 022001.
26. Sjostrand, T., S. Mrenna, and P.Z. Skands, *PYTHIA 6.4 Introduction to PYTHIA 8.1*. JHEP, 2006. **5**: p. 26.
27. T. Sjostrand, S.M., and P. Z. Skands, *A Brief Introduction to PYTHIA 8.1*. Comput. Phys. Commun, 2008. **178**: p. 825-867.
28. Andreopoulos, C. and e. al, *The GENIE Neutrino Monte Carlo Generator*. Nucl. Instrum. Meth. , 2010. **A614**: p. 87-104.
29. Brun, R. and F. Rademakers, "*{ROOT}* — An object oriented data analysis framework". Nuclear Instruments and Methods in Physics Research Section A: Accelerators,, 1997: p. 81-86.
30. Agostinelli, S. and e. al, *GEANT4: A Simulation Toolkit*. Nucl. Instrum. Meth. , 2003. **A506**: p. 250-303.
31. Kothari, S.C. and O. Heekuck, *Input normalization and Encoding*, in *Neural Networks for Pattern Recognition*. 1993, Elsevier. p. 297-300.
32. Rudin, C., *Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead*. Nature Machine Intelligence, 2019. **1**(5): p. 206-215.
33. He, H. and E.A. Garcia, *Learning from Imbalanced Data*. IEEE transactions on knowlegde and data engineering, 2009. **21**(9): p. 1263-1284.
34. Kayhan, O.S. and J.C.v. Gemert. *On Translation Invariance in CNNs: Convolutional Layers Can Exploit Absolute Spatial Location*. in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020.
35. Anzanello, M.J. and F.S. Fogliatto, *Learning curve models and applications: Literature review and research directions*. International Journal of Industrial Ergonomics, 2011. **41**(5): p. 573-583.

Appendix

A. Critical Reflection

The bachelor research thesis has overall been a very good experience. The project seemed rather daunting to me at first, as my knowledge about particle physics was very limited and my understanding of machine learning simply non-existing. However, the great support and encouragement of my supervisors to ask questions and discuss ideas quickly made me feel more confident. There was also a lot of opportunity to do so in the weekly meetings, which I truly appreciated. Another aspect of my BTR that I am particularly happy with is the plethora of skills it has taught me, which I will be able to use throughout the rest of my studies and career. These skills are mostly rather basic ones, such as working with a terminal and working with several python packages, but useful nevertheless. There are a few skills that I have improved to a more advanced level, for example the efficient identification of an error and finding an effective solution to it (which, in all fairness, comprised a large portion of the time that I spent on this thesis due to my lack of prior knowledge about coding and machine learning in general). That being said, I also realize that my lack of skills was a big limitation to the results I have been able to achieve in this research in the end. Therefore, I would like to do more research into the skills and knowledge required for a project and prepare myself better in advance the next time I find myself in such a situation. The only regret I have is not being able to travel to CERN in person to work with my supervisors and the rest of their team in person, but hopefully I will be able to do so at some point in the future.