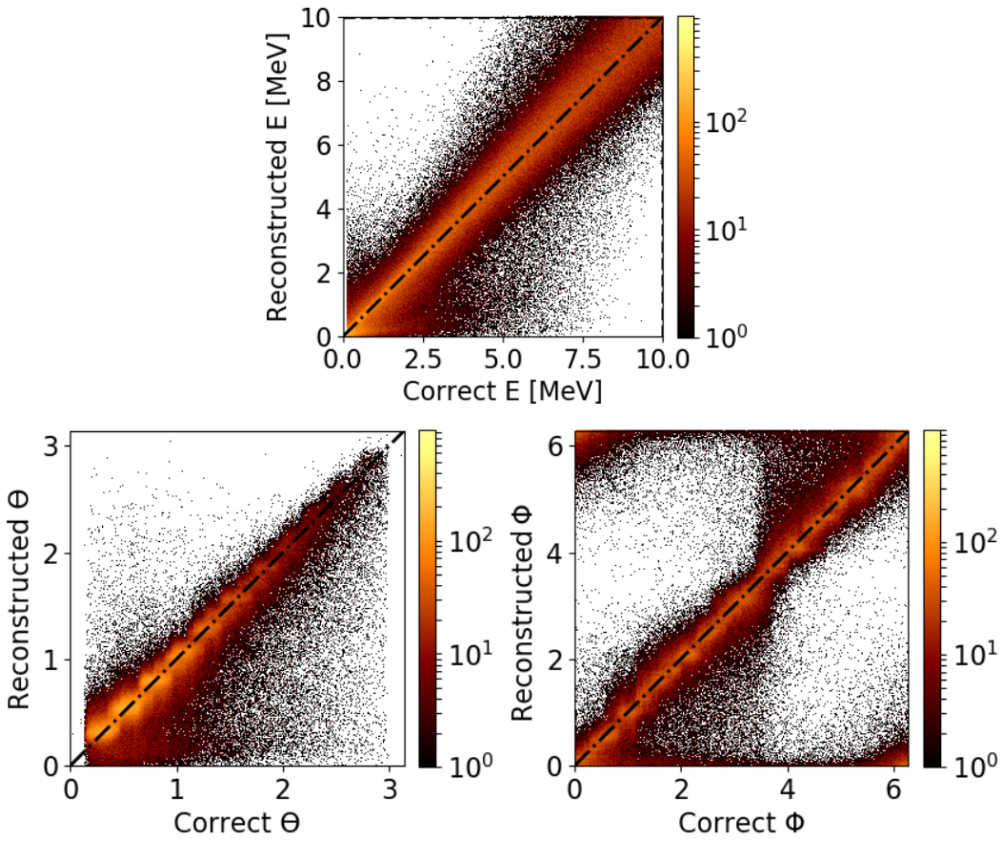
CHALMERS
UNIVERSITY OF TECHNOLOGY



Event reconstruction of γ-rays using neural networks

Going deeper with machine learning into the analysis of detector data

Bachelor's thesis in Engineering Physics: TIFX04-19-08

JESPER JÖNSSON
RICKARD KARLSSON
MARTIN LIDÉN
RICHARD MARTIN

Department of Physics
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2019

Bachelor's thesis TIFX04-19-08

Event reconstruction of γ-rays using neural networks

Going deeper with machine learning into the analysis of detector data

JESPER JÖNSSON
RICKARD KARLSSON
MARTIN LIDÉN
RICHARD MARTIN

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Physics
Division of Subatomic and plasma physics
Experimental subatomic physics
Chalmers University of Technology
Gothenburg, Sweden 2019

Event reconstruction of γ-rays using neural networks
Going deeper with machine learning into the analysis of detector data
Jesper Jönsson
Rickard Karlsson
Martin Lidén
Richard Martin

Supervisor: Andreas Heinz and Håkan T. Johansson, Department of Physics
Examiner: Lena Falk, Department of Physics

Bachelor's Thesis TIFX04-19-08
Department of Physics
Division of Subatomic and plasma physics
Experimental subatomic physics
Chalmers University of Technology
SE-412 96 Gothenburg
Sweden
Telephone +46 31 772 1000

Cover: Reconstruction of the energy, polar angle and azimuthal angle for γ-rays with beam frame energy up to 10 MeV.

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2019 3

Abstract

This study aims to develop and investigate artificial neural networks that reconstruct the energy and emission angles of relativistic γ-ray events in the CALIFA (CALorimeter for In-Flight detection of γ-rays and high energy charged pArticles) particle detector. In the present, the addback algorithm is used to reconstruct such events, but only to a limited extent because of difficulties arising from Compton scattering and pair production of the γ-rays. Both fully connected and convolutional neural networks were investigated and evaluated with detector data which was simulated with the software toolkits Geant4, ggland and ROOT. When finally comparing the neural networks with addback they showed potential to have equal or better accuracy than the addback for γ-ray energies of 3.5 to 10 MeV. However, the addback had a better signal-to-background ratio between 1 and 10 MeV whereas the neural networks also showed significant issues reconstructing lower energies. The best performing networks were fully connected neural networks trained on simulated detector data that took relativistic effects into account with a mean square-based cost function. Furthermore, this study concludes some hyperparameters of the neural networks which are suitable for the event reconstruction as well as suggestions for further investigation.

Keywords: addback, artificial neural networks, CALIFA, convolutional neural networks, Crystal Ball, gamma ray reconstruction, particle detector, TensorFlow

Acknowledgements

Firstly, we would like to express our gratitude to our supervisors Andreas Heinz and Håkan T. Johansson for giving us the opportunity to work with this project. It has been a strenuous, but educational, journey during which their help have kept us on track. Håkan, with his infallible knowledge in computer science and ever present encouragement to keep the GPUs running hot. Andreas has with his pedagogical explanations and insightful remarks helped us understand the underlying physics of the problems encountered.

Secondly, we would like to give credit to Jacob Olander, Miriam Skarin, Pontus Svensson and Jakob Wadman, whose work laid the foundation for what we have accomplished this year.

Lastly, we would also like to thank Swedish National Infrastructure for Computing (SNIC) and High Performance Computing Center North (HPC2N) for computational resources. Without the computational time on Kebnekaise we would probably still be waiting for the networks to finish training.

The authors, Gothenburg, May 2019

Contents

1
Introduction

Particle accelerator facilities are among the largest and most advanced structures mankind has ever constructed, designed to study the smallest particles of our universe. In the fields of nuclear and particle physics, progress is made in large collaborations such as the accelerator at the upcoming Facility for Antiproton and Ion Research (FAIR) in Darmstadt, Germany. Continuous effort is needed for advancements and a potential area of improvement is the process of analysing the detector data.

1.1 Background

One of the particle detectors used in the collaboration project Reactions with Relativistic Radioactive Beams (R^3B) at FAIR is the CALorimeter for In-Flight detection of γ-rays and high energy charged pArticles (CALIFA), described by Aumann *et al.* [1]. CALIFA has 1952 scintillating CsI-crystals to detect protons and photons, the latter in the γ-ray energy interval 0.1 MeV up to 30 MeV [1]. It is a relatively large detector with a complex geometry enclosing the target. In addition to measuring the photon energies, their emission angles can also be reconstructed.

Since the γ-rays can deposit parts of their energy in different crystals, the analysis of the detector data is a difficult task. To analyse the detector data, a so-called addback routine is currently used. There are different kinds of addback routines and they are all, as the name implies, based on attempting to add up all the detected energy deposits for some neighbouring crystals, assigning the total energy sum to one γ-ray [2]. A problem with this approach is when two photons simultaneously deposit energy in close proximity of each other. In this case, it is difficult to separate the two γ-ray detections from each other in the reconstruction. Therefore, new ways to analyse the detector data have to be considered.

One approach is to use artificial neural networks (ANN) to analyse the detector data. One of the strengths of ANN is to identify patterns in complex data sets. A common example of applications for neural networks is classification problems such as image recognition; detecting the presence of different features in images [3, p. 98]. As for the applications of this study, these features would correspond to the detected γ-rays in collision events. It is then also possible to reconstruct the actual energies and scattering angles for each detected γ-ray. In the bachelor project *Rekonstruktion från detektordata med hjälp av neurala nätverk* from 2018, Olander *et al.* [4] showed that ANN are capable of these tasks. However, it could not be shown that these reconstructions were better relative to those of the addback routine.

1.2 Purpose

The aim of this thesis is to develop and investigate structures of artificial neural networks to improve the reconstruction of the energies and emission angles of relativistic γ-ray events in a particle detector.

1.3 Delimitations

Artificial neural networks are usually trained by adjusting their response to inputs so that they reproduce known answers. There is no method to correctly label real detector data with answers since there is no way of actually knowing the answers beforehand. Therefore, simulated data will be used to train the ANN since it can be properly labelled.

The photons whose properties are of interest to reconstruct are in the γ-ray energy region. The main focus in this thesis is explicitly to reconstruct γ-rays in the energy interval 0.1 to 10 MeV. It would be of great interest to be able to study detector data of up to ten or even more simultaneously detected γ-rays. Because of the increase in network training time from using data with more γ-rays, the work will mainly be limited to detections of up to three photons at a time. The generated detector data will be limited to only contain events where more than 90 % of the γ-ray energies have been deposited in the detector. It would be an unfair task for the networks to try to reconstruct the rejected events with incomplete detections.

Since CALIFA has a complex geometry with a large amount of detector crystals, it is easier to start designing neural networks for a detector with fewer detector crystals and a more regular geometry. Therefore, this study will focus on another particle detector named Crystal Ball, which has scintillating NaI-crystals as explained by Simon *et al.* [5]. Crystal Ball has a spherical geometry and 162 detector crystals of four shapes, all of equal solid angle [5]. Generalisation to other particle detectors was investigated by Olander *et al.* [4] by using the constructed neural networks on data generated from the detector Dali2. From the results of Olander *et al.* [4], and because of the physical similarities between CALIFA and Crystal Ball, the network structures developed for Crystal Ball will likely be generalisable to CALIFA.

2
Theory

To assimilate the underlying problems investigated in this thesis and their suggested solutions, some understanding of subatomic physics, data simulation and artificial neural networks is necessary. Even though the study focuses on ANN applied on generated detector data from the Crystal Ball, the theory is also applicable to the corresponding situation of CALIFA.

2.1 Physical background and particle detection

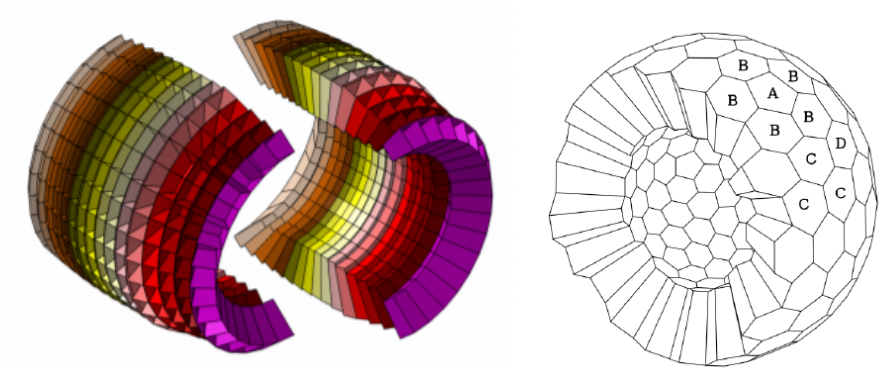
Different kinds of particles interact with matter in different ways. Particles with charge, such as protons and electrons, deposit energy by other mechanisms than the uncharged photons, here also denoted as γ-rays. How the particles of interest interact with the detector materials is fundamental for the mechanisms of the detection of particles created in the collisions of an accelerated ion beam. Both CALIFA and Crystal Ball are designed to detect both photons and protons, and the theoretical background of how these interact with matter is described by for example, Martin [6] and Leo [7]. This study focuses on the detection of γ-rays.

2.1.1 Particle detectors

One way to investigate the structure of nuclei is to study the reaction products created when accelerated ion beams collide with the target. When using beams of exotic and unstable isotopes, these experiments have to be carried out at accelerator facilities like the GSI Helmholtzzentrum fur Schwerionenforschung (GSI) and FAIR, in Darmstadt Germany, which houses the Crystal Ball [8] and CALIFA [1] detectors, respectively.

Both CALIFA and Crystal Ball consist of segments of detector crystals, where all crystals are working as individual detectors covering the same solid angle, as reviewed by Lindberg [5]. By enclosing the target with crystals, filling as much of the solid angle as possible, the detector obtains a spatial resolution depending on the areas of the individual crystals facing the target. The segmentation makes it possible to, besides measuring the energies, reconstruct the emission angles of the γ-rays. This is important when dealing with relativistic corrections of the photon energies, further reviewed in section 2.1.3. Both the CsI- and NaI-crystals are scintillating, proportionally transforming the deposited kinetic energy into visible light [7]. This light is detected by a photon multiplier producing an amplified electric signal, before being gathered as the deposited energy detector data [2].

CALIFA is made of 1952 scintillating crystal modules, ordered in the shape of a barrel with a semi-spherical end as illustrated in figure 2.1 [1]. The spherical Crystal Ball with 162 crystals covers 98 % of the full solid angle. This corresponds to a covered solid angle of 0.076 steradians (sr) per detector crystal. The crystals are of different shapes; three hexagons and a pentagon. Therefore, crystals have a varying amount of adjacent detector crystals, from now on referred to as neighbours. This is also affected by the two missing crystal modules where the radioactive beam enters and exits the detector. There is a risk that γ-rays might escape the detector without depositing all of their energy [2]. By dealing with multiple crystals, the collected data will be more complicated to analyse. By measuring the time of the γ-ray detections for each crystal, it is possible to separate detections from other reaction events. For Crystal Ball, the time resolution is approximately 3.5 ns, enough to separate different events, but insufficient to isolate single γ-ray hits from others in the same event [4]. A photon travelling with the speed of light, approximately 30 cm/ns, would be able to traverse the detector several times within the time frame set by the resolution, since the inner radius is 25 cm [5].



(a) **(b)**

Figure 2.1: Figure (a) is an illustration of the CALIFA barrel detectors, leaving out the semi-spherical structure at the forward end of the detector. Picture from [1]. Figure (b) shows the structure of the Crystal Ball, with the four different shapes of the detector crystals, labelled A-D. Picture from [9].

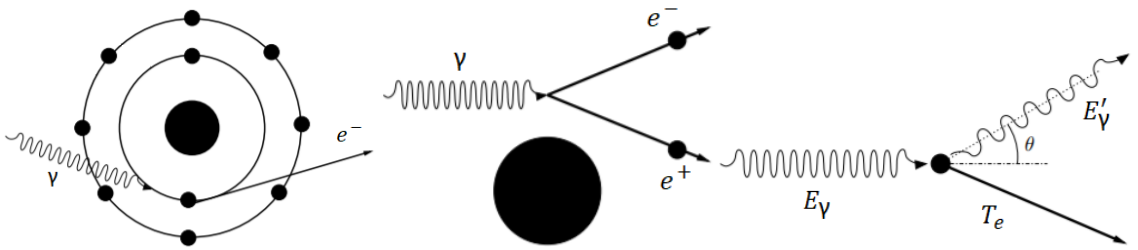
2.1.2 Interaction of photons with matter

In contrast to charged particles, like protons and electrons, γ-rays lose energy in discrete portions. In the energy range of interest, this can happen in three different ways, described in more detail by Leo [7]:

Photoelectric absorption; where the energy of the photon is absorbed by a bound electron. The electron is ejected with kinetic energy E_{kin}, as the difference of the γ-ray energy E_γ and the electron binding energy B_K: $E_{kin} = E_\gamma - B_K$.
Pair production; by which the γ-ray is annihilated and an electron-positron pair is created. This can only occur within close proximity of a nucleus, otherwise it

would violate momentum conservation. Like the photoelectric absorption, the photon deposits all of its energy, which has to be greater than 1.022 MeV; the rest masses of the electron and positron. The rest of the γ-ray energy is passed on as kinetic energy to the two particles.
Compton scattering; where only a part of the photon energy is transferred as kinetic energy to a recoiling electron, also called Compton electron. The photon can be scattered at any angle, with the possibility of losing small fractions of its energy. Compton scattering is usually the dominating interaction with matter at the energies of interest for this work.

These three reactions are illustrated in figure 2.2. It is the scattered electrons and positrons from these three reactions that are detected and measured by the CsI- and NaI-crystals, not the actual γ-rays, as described by Leo [7, p. 174]. In contrast to the γ-rays, the electrons and positrons are not likely to escape the detector [7].



(a) **(b)** **(c)**

Figure 2.2: Illustration of photoelectric absorption (a), by which the γ-ray is absorbed. In pair-production, (b), a γ-ray is converted into an electron and a positron in the vicinity of a massive object. Compton scattering is presented in (c), where the γ-ray is scattered on an electron. Pictures adapted from [10].

2.1.3 Relativistic effects

The beams used for the collisions studied by CALIFA and Crystal Ball use particles at relativistic velocities, in practice 50 to 70 % of the speed of light, c [1]. The ion beam reacts with the target material, which is detected in the laboratory frame with the surrounding detector. The γ-ray properties of interest have to be studied in the coordinate system moving relative to the laboratory frame; the beam frame. Since it is the detections in the laboratory frame that are measured, relativistic effects occur and have to be dealt with when analysing the detector data. The most prominent effects are the energy Doppler shift and headlight effect. The name of the latter originates from the fact that the emitted photons appear to be scattered conically in the forward direction. The transformations for the γ-ray energy and polar emission angle in the beam frame, E' and θ' respectively, to the stationary laboratory coordinate system, E and θ, are given by

$$E' = \frac{1 - \beta \cos\theta}{\sqrt{1 - \beta^2}} E, \qquad \cos\theta = \frac{\beta + \cos\theta'}{1 + \beta \cos\theta'}, \tag{2.1}$$

where the beam travels with the speed βc along the z axis relative to the laboratory frame [11, p. 78-82]. This energy is called Lorentz boosted energy. Since it is the Lorentz boosted energy that is of interest, it is of great importance to be able to reconstruct the emission angle in the laboratory system in order to perform the energy transformation.

2.2 Data analysis and addback routines

Since the detectors themselves cannot determine the properties of the different reaction events, a method to reconstruct the detected γ-ray properties is needed. As mentioned earlier, γ-rays deposit their energy in the detector crystals in three different ways; photoelectric absorption, pair production and Compton scattering. If photoelectric absorption would be the only way, all γ-rays would deposit all of their energy at once. It would then be almost trivial to determine the number of simultaneously emitted photons from the reaction, called the multiplicity of an event, and their energies. Pair production is also relatively easy to track. The problem of analysing the detector data occurs when the photons deposit parts of their energy by Compton scattering in multiple crystals.

In Crystal Ball, a γ-ray of energy up to 10 MeV deposits energy in typically 1-4 different crystals [2]. Also, γ-rays escaping the detector, as illustrated in figure 2.3, make it impossible to detect the total γ-ray energies in these events. Thus, there is an upper limit for the achievable accuracy of the data analysis.

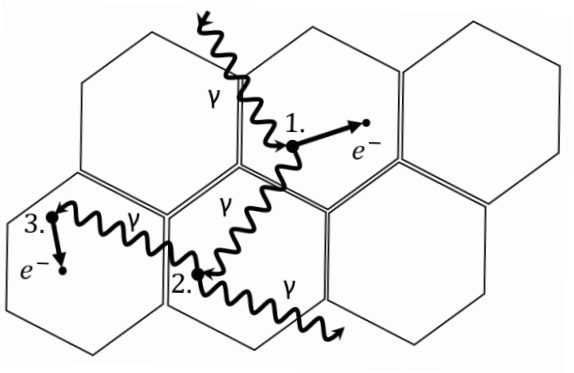
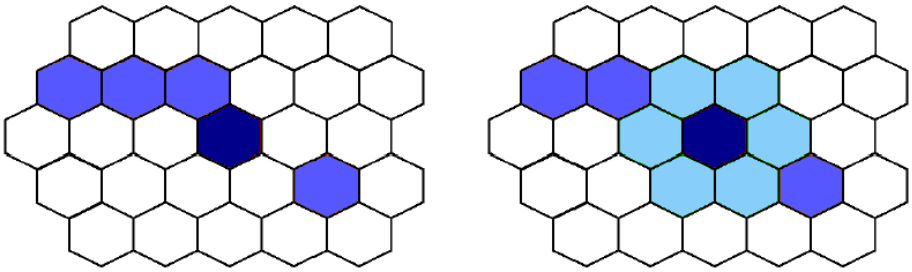


Figure 2.3: An incoming γ-ray is Compton scattered (1.) by which the new γ-ray is converted into an electron-positron pair by pair-production (2.). The positron is almost immediately annihilated, producing two new γ-rays where one is escaping the detector crystals. The other γ-ray is photoelectrically absorbed (3.). Note that it is the scattered and produced electrons and positrons that are being detected by the crystals, and that the distance covered by the electrons is exaggerated. Energy has then been deposited in three of the illustrated detector crystals.

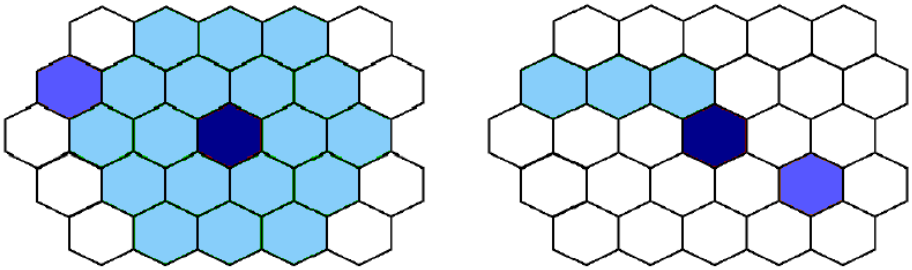
The common algorithm for reconstructing properties of γ-rays in a detected event is the addback routine, described by for example Lindberg [2]. According to [2], the basis of addback is to proceed from crystals with local maximum of deposited

energies, which are assumed to originate from primary γ-ray hits. By taking the sum of the deposited energies in the nearby crystals, the total energy of each γ-ray is calculated. The problem is to avoid counting two primary hits close to each other as one, for example in the case when multiple γ-rays deposit their energy in the same crystal. In the study by Lindberg, three different addback routines were compared for the Crystal Ball, as illustrated in figure 2.4. They were almost equally precise. All of the addback routines have an accuracy significantly lower than the theoretical upper limit [2].



(a) Crystals with energy deposited.

(b) Crystals added in the neighbour routine.



(c) Crystals added in the second neighbour routine.

(d) Crystals added in the bunch routine.

Figure 2.4: Figure (a) shows the primary hit of a γ-ray, dark blue, with secondary energy depositions in ordinary blue colored crystals. Figure (b) and (c) show examples of addback routines taking the sum of energies from neighbouring crystals (b) and also second neighbouring crystals (c) to the primary hit. The third addback routine can be seen in (d) where it follows a track of crystals in which energy was deposited. The crystals taken into account by the addback is marked with light blue. Observe that all of the methods misses at least one crystal in which energy has been deposited. Figure adapted from [2].

2.2.1 Data generation tools: Geant4, ggland and ROOT

To simulate high-energy accelerator physics experiments, a C++ based toolkit called Geant4 has been developed in a world-wide collaboration [12]. By using Monte Carlo based methods, Geant4 is able to simulate the interactions of particles with different matter configurations.

To facilitate the use of the tools for accelerator physics, a wrapper program called ggland has been developed by Johansson [13], mainly for simulations of R^3B exper-

iments. Through ggland, it is possible to simulate high energy particle reactions in detectors like the Crystal Ball. The program is able to simulate a vast number of events with various multiplicities of emitted particles or γ-rays.

To handle the large amounts of data encountered in accelerator physics, a framework for data processing called ROOT has been developed at CERN [14]. ROOT makes it possible to save, access and handle the generated data from experiments as well as programs such as Geant4 and ggland.

2.3 Artificial neural networks

Artificial Neural Networks are computational frameworks for machine learning algorithms, inspired by the neuron networks in the human brain. The strengths of ANN lie in identifying and learning complex patterns, often too tedious for a human to recognise. Thus, ANN are used for tasks such as image recognition and detection of neural diseases for preventive purposes [15]. In practice, ANN should be considered as an approximation of an arbitrary function $\hat{f} : X \rightarrow Y$ which describes the problem of interest. In the case of this study, ANN are used instead of the addback routine for reconstruction of detected emission events from reactions induced by accelerated beams. This has also been investigated by Olander *et al.* [4]. The task of the network is to use the data from the detector to identify the correct number of detected γ-rays, called the multiplicity, as well as the kinetic energy and the emission direction of each individual γ-ray.

The task is of two different kinds; **classification** of the multiplicity and the **regression** problem of reconstructing for example the γ-ray energies. In standard classification problems, the network output is restricted to a discrete set of possible answers or as a probability distribution of different alternatives. For regression, the network output lies in some continous spectra, which in the case of this study is the energies and angles of the γ-rays. In the following section the basics of two commonly used ANN structures are introduced, based on the works of Goodfellow *et al.* [3] and Theodoridis [16].

2.3.1 Fully connected neural networks

Fully connected neural networks (FCN) are ANN based on a sequence of layers. Each layer consists of several nodes where every node is connected to all nodes in the layer before and after.

In this thesis, only feedforward networks are used, which is the case where the neurons propagate their information in the forward direction of the sequential layers. Thus, the data from the first layer, the input layer, is passed to the nodes in a second layer and so on until the last layer of the network, the output layer. This is illustrated in figure 2.5.

The number of nodes in the input and output layers are specified by the data to be analysed and the format of the desired results, respectively. In between, the intermediate layers can be adjusted to contain any desired amount of nodes. This will alter the number of node connections and it may change the capacity of the network, which will be disscused below.

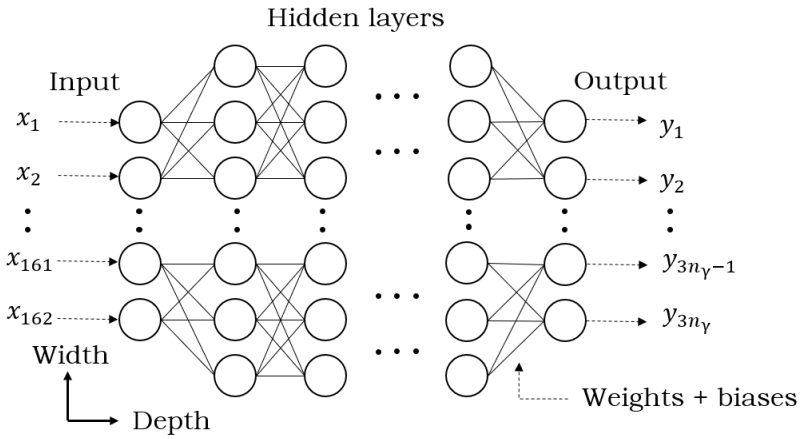


Figure 2.5: The structure of a fully connected network. Observe that the first layer of nodes is limited to having the same shape as the input data. This also applies to the last layer which has the shape of the desired output, further discussed in section 3.3.1. To clarify, each node in every layer is connected to all the nodes in the previous and following layer.

The connections between the layered nodes consists of a linear transformation followed by a non-linear activation function. The transformation is a sum of the weighted values of all the n nodes in a layer, where each node is denoted x_j. A bias b_i is added to the sum, used as an offset. Thus, the transformation is

$$z_i = \sum_{j=0}^{n} w_{ij} x_j + b_i, \tag{2.2}$$

where w_{ij} is a weight. The transformation can also be expressed in matrix notation with a weight matrix $\mathbf{W}$ and the column arrays $\mathbf{z}$, $\mathbf{x}$ and $\mathbf{b}$,

$$\mathbf{z} = \mathbf{W}\mathbf{x} + \mathbf{b}. \tag{2.3}$$

The elements of the weight matrices $\mathbf{W}$ and biases $\mathbf{b}$ are adjustable parameters which are changed during the training of the neural network.

The activation function is a non-linear function that is applied to $\mathbf{z}$. There exist multiple types of activation functions which are used for different purposes. A common activation function is the Rectified Linear Unit, ReLU, defined as $g(z_i) = \max\{0, z_i\}$. The ReLU function returns the input value as long as it is positive, and otherwise zero. A variant of the standard ReLU is the Leaky ReLU, which also is linear for negative inputs, but with a positive gradient smaller than one [17]. The Leaky ReLU deals with the so-called dying ReLU problem and is potentially more stable than the ordinary ReLU [17].

2.3.2 Cost function and back propagation

For every given input data $\mathbf{x}$, the network returns some output $f(\mathbf{x}) = \mathbf{y}$. Since the network is an approximation of some function $\hat{f}$, returning the output $\hat{f}(\mathbf{x}) = \hat{\mathbf{y}}$, the task of ANN are to as accurately as possible approximate $\hat{f}$, such that $f(\mathbf{x}) \approx \hat{\mathbf{y}}$. The deviation of f from $\hat{f}$ is quantified by a cost function $C(\mathbf{y}, \hat{\mathbf{y}})$. The purpose of the training process is to adjust the network weights and biases to minimise $C(\mathbf{y}, \hat{\mathbf{y}})$. Commonly used cost functions are often based on the mean square error,

$$C(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2, \tag{2.4}$$

where n is the number of network output nodes.

Back propagation is a technique that is used to change the network weights and biases in order to minimise the cost function. The principle of back propagation is to iteratively calculate the gradient of the cost function to find a minimum by adjusting the parameters of the network. A deeper explanation accompanied with a derivation of the back propagation can be found in appendix A. The name originates from the fact the network calculates the gradient backwards through the network, starting from the output layer. The approach of calculating the gradient to find a minimum is also known as gradient descent. An optimiser algorithm is usually implemented to perform the gradient descent [18].

2.3.3 Network capacity and overfitting

To solve demanding tasks, the ANN needs to have enough capacity. For example, the capacity is increased by the width and depth of the network. One disadvantage of using larger networks is that the training time increases with the number of nodes.

Another problem related to the network capacity is overfitting. A large network may overtrain and start to memorise the desired output for each specific input data. This results in loss of generalisation when the network is applied to new data that it has not been trained on. Overfitting primarily occurs when using too small data sets. To detect overfitting, the data is divided into two parts: a training set and an evaluation set. The latter is used to measure the performance of the network on data which it has not been trained on. Overfitting can be detected during the training process by calculating and comparing the cost from the training and evaluation data.

2.3.4 Convolutional neural networks

One of the most commonly used artificial neural network types is convolutional neural networks (CNN), which are usually applied to image recognition problems [3]. The name derives from the fact that CNN utilise the mathematical operation of convolution to detect local features in data samples. Figure 2.6 illustrates a convolution layer. The layer consists of a kernel that contains adjustable weights which will be altered by training. The kernel is usually of much smaller size than

the input and has a fixed stride, i.e. it is applied to the input in increments equal to the stride. In the example in figure 2.6, the kernel has three weights and a stride equal to its length. If the stride is lowered, the kernel would overlap when moving through the elements.

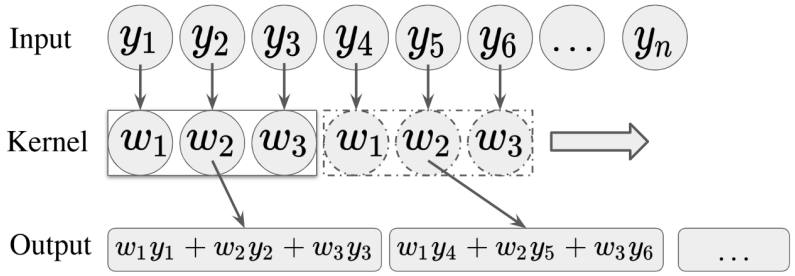


Figure 2.6: Schematic overview of the one-dimensional convolutional operation in a network layer. In this example, the kernel contains three elements. It is applied to adjacent sets of elements in the input without overlaps since the stride size is equal to the length of the kernel.

The convolution makes it possible to obtain a local weighted average of the input. The kernel can then be trained to recognise certain features. If more kernels are introduced they can learn other features. Moreover, the convolutional operation inherits the property of equivariance to translation which means that a change in the input corresponds to a similar move in output [3]. This equivariance as well as the the local property are what makes CNN suitable for image recognition problems [3].

The common way of constructing layers in CNN is to subject the input to a series of convolutions. Afterwards it would go through a non-linear activation function. Finally, a pooling function can be implemented to further adjust the output.

2.3.5 Pooling

The purpose of the pooling function is to summarise the information from a layer, which may be used to reduce the number of nodes for later layers. There are different types of pooling, where two of the most common are max pooling and average pooling [3]. Max pooling outputs the largest value of a group of adjacent nodes while average pooling outputs their mean value. There are different ways to handle the edges of a layer when applying a pooling operation. This is called padding and comes in two common types; valid and same [19]. Valid means that the pooling stops when the edge of the pooling kernel reaches the last node in the previous layer. Same means that the previous layer receives zeros around the edges in order to continue with the pooling all the way to the edge of the layers.

Pooling shares similarities with the convolutional layer by acting like a kernel and having a fixed length and stride, exemplified in figure 2.7. It is important to note

that pooling does not introduce any trainable parameters to the network but can in fact be used to decrease the amount of trainable parameters in the rest of the network [3].

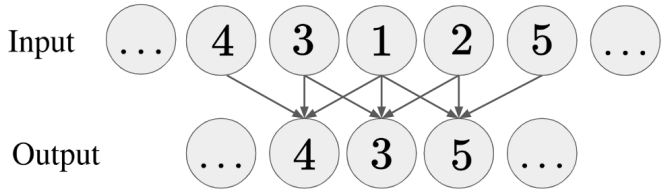


Figure 2.7: Overview of max pooling between two layers. The pooling has a stride of one and length three, by which it takes the maximum of three adjacent elements.

3
Method development

The aim of this study is to develop structures of ANN that are superior to the addback routines. These ANN will be based on convolutional and fully connected neural networks. By starting from a basic network structure and gradually testing different approaches to improve this network, the goal is to find an optimal neural network to reconstruct Doppler-shifted events.

The first sections of this chapter will explain how neural networks are implemented as well as how these networks are trained and evaluated. From section 3.5 and onward, different aspects of the neural networks are studied, such as the impact of the cost function or the structure of the convolutional neural network.

3.1 Previous work

In the study by Olander *et al.* [4] it was shown that ANN can be of aid in reconstructing certain properties of γ-ray events from detector data. Based on data generated with ggland for the Crystal Ball detector, they used fully connected networks for reconstructing γ-ray energies and polar angles. Different cost functions were compared, all based on the mean square cost function in equation (2.4). Determination of the event multiplicities was also performed for multiplicities one to six. For this problem a classification network was used. By applying the techniques of ANN to a similar detector, Dali2, it was shown that the ANN to some extent were generalisable to other particle detectors [4].

3.2 Utilising high-performance computing

The ANN structures presented in section 2.3 are typically used to design large networks consisting of thousands to millions of trainable parameters. This means that the matrices involved in the computations grow large and thus pose memory usage problems for ordinary computers. With increasing network size, the amount of training iterations needed for the performance of the networks to converge also increases, which causes the training process to be time consuming. A way of handling such problems is to utilise a high performance computing (HPC) resource for the training process. In this study, the Kebnekaise computer cluster at the High Performance Computing Center North (HPC2N) has been used. Kebnekaise offers both CPU- and GPU-based compute nodes [20]. The CPU nodes are based on Intel Xeon E5-2690v4 processors while the GPU nodes use NVidia Tesla K80 graphics

cards.

Standard CNN used in this study took roughly 400 minutes to train on one GPU, while the same training took twice the time using one CPU. Thus, parallelising the code to simultaneously utilise multiple processors is important when training large networks.

3.3 Building the neural network

TensorFlow was used to implement the neural networks designed in this study. It is an open-source machine learning platform developed by Google [21]. The following section explains some aspects of the implementation of the neural networks and also presents solutions to encountered problems.

3.3.1 Training and evaluation data

The neural networks were given two sets of data; the detector data and the event labels. For each event, the detector data was a one-dimensional array with 162 elements containing the deposited energy in each of the detector elements of Crystal Ball. The length of the detector data array is fixed, independently of the amount of detected γ-rays. However, the amount of activated detector crystals varied for different event multiplicities. Each detected γ-ray has two properties of interest to reconstruct; its kinetic energy E and the spatial position of the primary hit. The latter was represented with two spherical coordinates; the polar and azimuthal angle, θ and ϕ respectively.

The event labels were used to train and evaluate the networks. Their sizes depend on the multiplicity of the analysed detection events, containing the correct energies and the spherical emission angles for each of the γ-rays. Thus the label arrays had lengths of three times the multiplicity n_γ, as can be seen in figure 2.5. For example, the label for an event with two γ-rays would be

$$\left(\hat{E}_1,\ \hat{\theta}_1,\ \hat{\phi}_1,\ \hat{E}_2,\ \hat{\theta}_2,\ \hat{\phi}_2\right). \tag{3.1}$$

The output data of the neural networks have the same size and format as the label data as a consequence of how the network was designed. If a network was trained to identify different multiplicities, the labels of fixed length were filled with so-called zero γ-rays. These non-existing photons were represented with zeros for the energy and spherical angles to demonstrate that no γ-ray was present.

3.3.2 Fully connected neural networks

The fully connected neural network (FCN) is one of the simplest neural networks. The networks used 162 input nodes, corresponding to the detector crystals. In-between there are intermediate fully connected layers with adjustable depth and width. If nothing else is specified, this hidden structure consisted of ten layers with

128 nodes in each layer. This is the standard that was used when doing tests on other parameters with FCN.

3.3.3 Convolutional neural networks

The convolutional neural networks used throughout this study usually consisted of repeatedly stacked convolutional layers with several kernels and intermediate pooling layers. Thereafter, the networks ended with a couple of fully connected layers. The input given to the network is a one-dimensional array with the information about the deposited energies in each crystal of the particle detector. In contrast to the input given to the fully connected neural network, the input array to the convolutional neural networks also contains information about the geometry of the particle detector.

As a reminder, convolutional networks are good for obtaining local information in the input. This is utilised by clustering data from neighbouring crystals in the particle detector, where each crystal is associated with a deposited energy. The clusters consist of one central crystal and its nearest and second nearest neighbour crystals. A map of the clusters was created by indexing each crystal and specifying which indices that belonged to the same cluster. These are sorted in an array as seen in figure 3.1. The crystals are sorted by first picking the crystal that is closest to the forward direction and then the remaining crystals that followed in the anti-clockwise direction. This sorting was made separately for the nearest and next-nearest neighbouring crystals[1]. The final step of creating the neighbour cluster map is to concatenate each individual cluster in a single row. This is the actual input to the convolutional neural network.

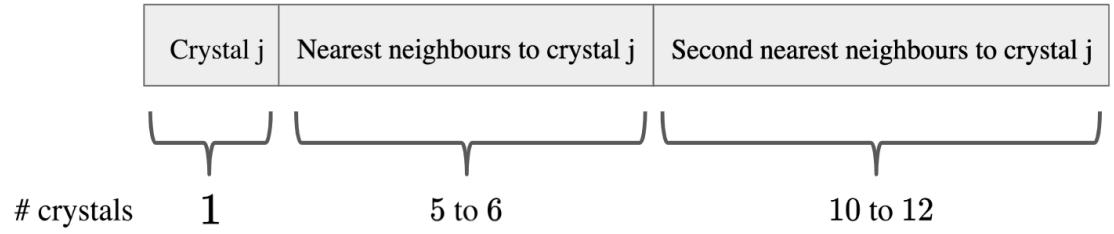


Figure 3.1: Example of the one-dimensional map that contains the index of crystal j and its neighbours. The number of crystals in each neighbour section depends on the shape of crystal j, whether it is a pentagon or a hexagon.

The kernels of the first convolutional layer have the same length and stride as the clusters. It sweeps through the array to summarise the deposited energy data for each neighbourhood of crystals. It is important to note that there are four types of crystals in the Crystal Ball, three with different hexagonal shapes and one pentagon. Each crystal type is denoted with a letter A, B, C or D, and can be seen in figure

[1]In hindsight, the sorting should not have been made separately. There is a chance that the inner and outer layer of neighbours will shift with one crystal, which ruins the consistency of the sorting.

2.1. Since the shape of the center crystal varies, so will the number of neighbours as well. Therefore, the total size of the clusters was either 16 (A), 18 (B and C) or 19 (D) crystals. Consequently, the clusters belonging to different crystals can not be treated the same way since they require different kernel sizes in the first convolutional layer. The solution is to separate the A clusters from the others by dividing the network into parallel sequences of layers. These parallel channels end up being concatenated and flattened after their last layer. Thereafter they are passed on to a set of fully connected layers. This process is illustrated in figure 3.2.

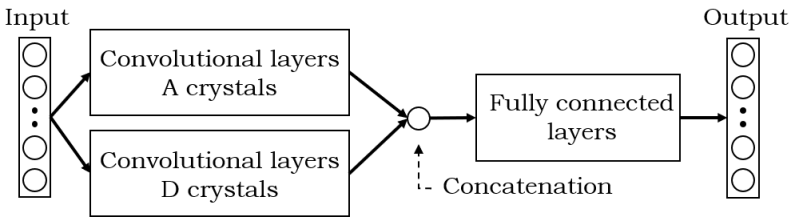


Figure 3.2: A schematic for the general structure of the CNN composition. After the input, the convolutional layers are split into two parallel channels to account for the geometry of the crystals. The outputs from the convolutional layers are then concatenated and followed by fully connected layers.

Various combinations of mappings were tested, where each test used clusters with different crystal types as the center crystal. The final mapping only used clusters including A and D crystals in the center. These crystals are evenly spaced and still cover the entire detector, with som overlap. No major differences were seen when comparing this approach to creating a map with more types of clusters.

3.3.4 Hyperparameters

Different network settings and parameters such as kernel sizes, the number of layers or the choice of cost function are the **hyperparameters** of the network. What distinguishes these parameters from others is the fact that they are set before the training. The different hyperparameters are essential for the understanding of the behaviour of the neural networks, and have to be investigated to optimise the performance of the networks.

To perform the back propagation in the network training process, the Adam optimiser was used; an algorithm based on stochastic gradient descent [22]. Six common optimisers were tested and Adam turned out to give the best performance. The learning rate of the optimiser was set to 10^{-4}. This was decided after having tested different learning rates without observing any significant improvement. Furthermore, the Leaky ReLU was used as the activation function with the slope of 10^{-3} for negative values.

3.4 Event reconstruction

As described in section 3.3.1, each detected γ-ray has three observables that are of interest to reconstruct: its energy E, and the two spherical coordinates θ and ϕ. The studied collision events are generated with γ-ray energies uniformly distributed in the intervals 0.1 to 10 or 15 MeV with an isotropic angular distribution. To each deposited energy in the simulated detector data, a normally distributed error with a standard deviation of 5 % of the energy is added. The following reconstructions have been made with FCN and the standard cost function in equation (3.2) which is presented in section 3.5.

3.4.1 Single γ-rays

A basic task for the networks is to reconstruct the γ-ray properties in events with only one γ-ray. For multiplicity one, the reconstruction is presented in the two-dimensional histograms in figure 3.3 for γ-rays of energy up to 10 MeV.[2] For an ideal network, the reconstructed data follows the identity line, shown as a black, dash-dotted line. For the polar angle, there are no γ-rays detected in the beginning and end of the interval $[0, \pi]$. That is due to the fact that the two corresponding crystals are removed to let the beam enter and exit the Crystal Ball [23].

For the polar and azimuthal angle reconstructions, a pattern can be seen with recurring circular shapes along the identity line. The amount of these circular shapes distributed in $[0, \pi]$ seems to correspond to solid angles matching those of each detector crystal, reviewed in section 2.1.1. The γ-rays depositing energy in different parts of a crystal are detected and represented the same way, independent of where on the crystal the hit actually was. However, the labels with the correct angles of incidence are not limited to a discrete set of values. This results in an inescapable error in both the training and evaluation processes and explains the observed circular pattern.

[2]When viewed in grayscale, the white background of the histogram should not be confused with the maximum intensity of the colorbar.

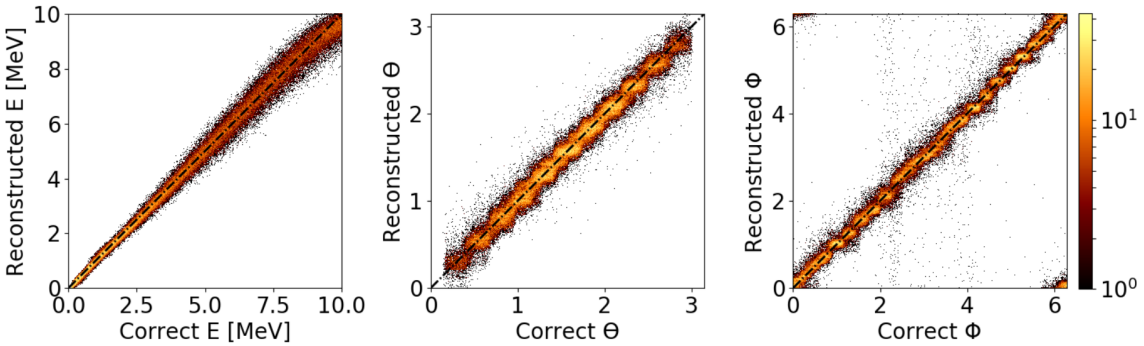


Figure 3.3: Reconstruction of energy, polar angle and azimuthal angle for γ-rays in events with multiplicity one. The correct values are shown on the abscissa with the corresponding reconstructed values on the ordinate, forming a two-dimensional histogram. Note that the same colorbar applies to all of the plots.

3.4.2 Multiple γ-rays

For events with multiple detected γ-rays, the event reconstruction requires more effort from the network since the events are more complex. The reconstruction is shown in figure 3.4 with one to three γ-rays with energies up to 10 MeV in each event. Note that this reconstruction is less accurate than with multiplicity one in figure 3.3. The patterns in the two angles are also more blurry than for multiplicity one.

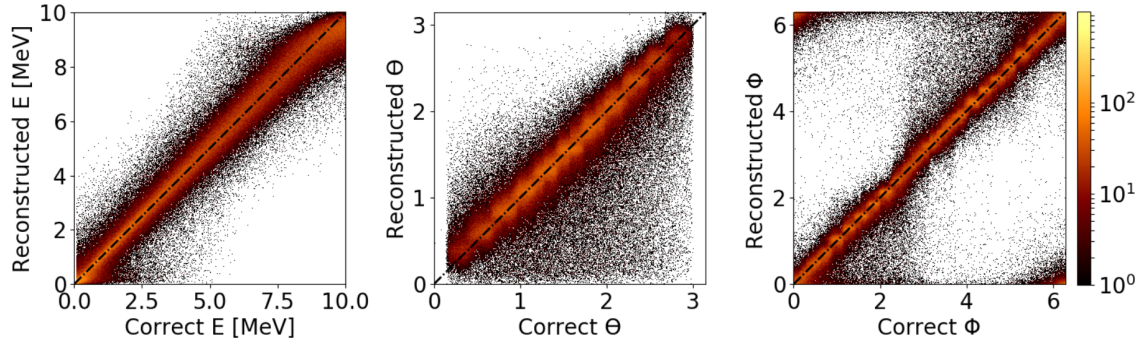


Figure 3.4: Reconstruction of the energy, polar angle and azimuthal angle for γ-rays in events with multiplicity one to three. Note the deterioration of the reconstruction in comparison with the multiplicity one reconstruction in figure 3.3.

3.4.3 Measurement of performance

To measure the performance of the networks, each of the correct values of the observables are divided into a number of intervals. For each of these intervals, the difference of the reconstructed and correct values are computed. As a measurement of performance, the averages of the mean errors and standard deviations were used. The mean error can tell if the quantities were reconstructed too low or high, while the standard deviation corresponds to how accurate the reconstructions are.

3.4.4 Training with higher γ-ray energies

When training with output quantities defined on a finite interval, the networks will adapt and rarely report values outside the given limits. In this particular adaption of neural networks, the two spatial angles are strictly defined in two intervals; $\theta \in [0, \pi]$ and $\phi \in [0, 2\pi]$. However, for the energy, the γ-rays can carry almost any positive energy, by which there is no theoretical upper limit to what the network should be able to reconstruct. Therefore, by training the networks on data with energies up to just 10 MeV the networks will not reconstruct energies above this value. As a consequence, the networks will often reconstruct too low energies for γ-rays close to 10 MeV, as seen in figure 3.3 and 3.4.

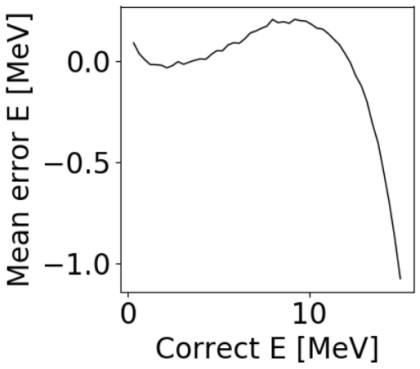


Figure 3.5: The mean error of a network trained with γ-rays of energy up to 15 MeV. Note the drop in the reconstructed energies from above 10 MeV.

The same undesirable artefact is visualised by the energy mean error plotted in figure 3.5, where a network was trained for γ-rays with energies of up to 15 MeV. Above 10 MeV, the energy reconstructions are too low, resulting in a negative mean error. This effect is avoided by only evaluating the network with γ-rays of energies up to 10 MeV. However, the standard deviation in energy of this network is larger than for the reconstruction in figure 3.4 when only training with energies up to 10 MeV. The dip in the energy reconstruction is a non-symmetric artefact which is undesirable. Therefore, the FCN used are trained with γ-ray energies of up to 15 MeV, and evaluated up to 10 MeV.

3.5 Cost functions

The cost function determines how the networks are trained in the back propagation process. The standard cost function used to reconstruct the energies and angles of the γ-rays is based on the mean square error;

$$C(\mathbf{y}, \hat{\mathbf{y}}) = \sum_{i=1}^{n_\gamma} \left[\lambda_E \left(\Delta E_i\right)^2 + \lambda_\theta \left(\Delta\theta_i\right)^2 + \lambda_\phi \left((\Delta\phi_i + \pi) \bmod 2\pi - \pi \right)^2 \right], \qquad (3.2)$$

where $\Delta E_i = E_i - \hat{E}_i$ and analogous for θ and ϕ. The cost is defined as the mean square error of all n events in each data batch, with n_γ γ-rays in each event. A batch size of 300 events was used in each training iteration. For each γ-ray, the energy and angles cost terms were weighted relative to each other by λ_E, λ_θ and λ_ϕ, all set to one if nothing else is specified. The weighting of the cost function terms are further investigated in section 3.5.5. The modulo in the azimuthal term was added to allow the reconstruction of the azimuthal angle in the periodic interval

$[0, 2\pi] + p \cdot 2\pi$ where p is an integer that is studied in section 3.5.2. The weighting factor λ_E has the unit of MeV^{-1}, which further on will not be written out.

3.5.1 Permutation cost function

For multiplicities n_γ greater than one, each reconstructed γ-ray has to be paired with the corresponding γ-ray in the label array. To find the correct pairing, a permutation cost function based on the work of Olander *et al.* [4] is used. The cost is calculated for all of the $n_\gamma!$ possible pairing permutations, from which the one generating the lowest cost is used. Thus, the choice of output pairing depends on the cost of the energy and the two angles. This means that all three observables are used to separate the γ-rays in an event. The reconstruction of the spatial angles can benefit from the network correctly reconstructing the γ-ray energies and vice versa. The reconstruction of events with multiplicities one to three with a cost function that uses permutation sorting is presented in figure 3.4.

An alternative to the permutation cost function is to require the network to assign the γ-rays without any external aid. A reconstruction using this approach is shown in figure 3.6. Comparing this reconstruction with the permutation cost function shows that the reconstructed energies and angles are much less accurate when no sorting was used. This strongly indicates that the networks benefit from the permutation cost function. Thus, the permutation sorting algorithm is used in all the following cost functions. A negative aspect of the permutation cost function is that the time complexity scales with $n_\gamma!$. This substantially increases the training time for greater multiplicities, which was quantified by Olander *et al.* [4].

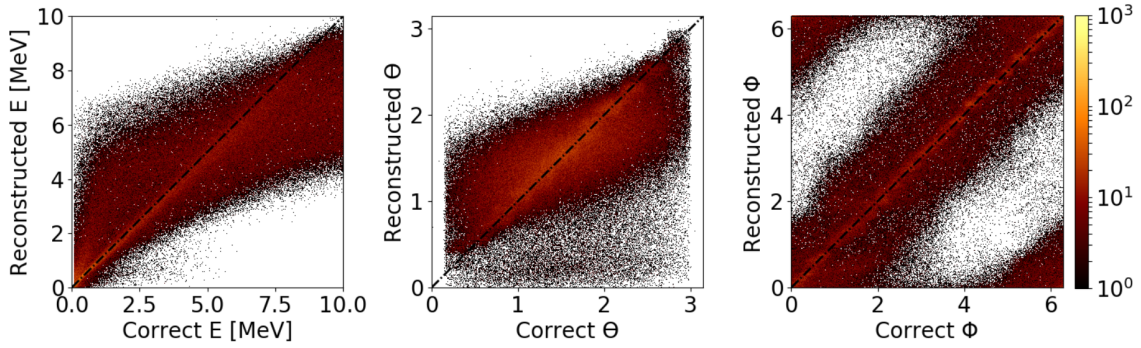


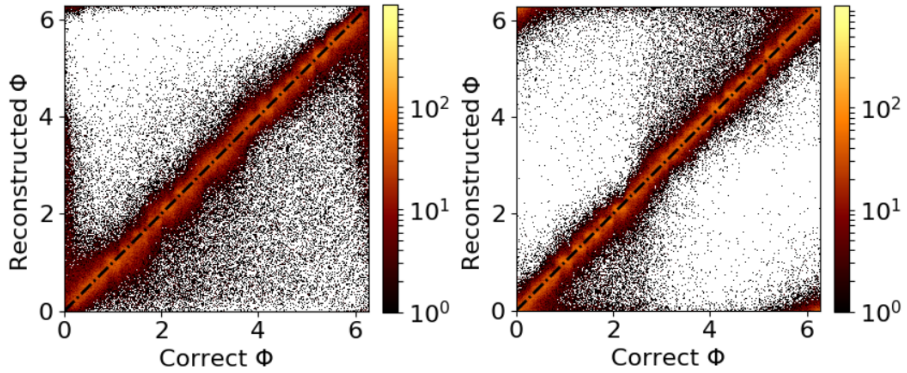
Figure 3.6: The event reconstructions without using the permutation cost function.

3.5.2 Modulo of the azimuthal angle

Because of the spherical geometry of the Crystal Ball, γ-rays can be detected both at azimuthal angles just above 0 or below 2π, in practise corresponding to similar angles. Therefore, the networks should not be punished for reconstructing the angles near 2π when the correct answer was slightly larger than 0 and vice versa. To achieve this, the modulo 2π is added to the azimuth term in the cost function, see equation (3.2).

A comparison of the outcome from using or not using the modulo 2π is presented in figure 3.7. Observe that without the modulo in the cost function, the network had difficulties reconstructing the azimuthal angles at the end intervals, as seen in figure 3.7a. With the modulo cost function in figure 3.7b, two blobs appear in the upper left and lower right corners of the histogram. The blobs imply that the modulo works as intended.

In figure 3.7a the network tends to evenly reconstruct azimuthal angles below the identity line, closer to the abscissa. With the modulo cost function, the same feature is seen for correct angles up to approximately three radians. For γ-rays with correct angles above this value, the network reconstructed the angles in the interval $[-2\pi,\ 0]$ instead of in $[0,\ 2\pi]$. Angles reconstructed in the interval $[-2\pi,\ 0]$ cannot be seen in figure 3.7b since these reconstructions have been shifted to the interval $[0,\ 2\pi]$ before being plotted. The tendency of reconstructing the angles closer to 0 then led to reconstructions above the identity line. This corresponds to the vertical line seen in figure 3.7b when the reconstructed angles are all plotted in the interval $[0,\ 2\pi]$.



(a) **(b)**

Figure 3.7: The reconstruction of the azimuthal angle for cost functions without and with modulo 2π, in (a) and (b), respectively.

3.5.3 Effect from reconstruction of the azimuthal angle

As discussed in section 2.1.3, to perform the relativistic Doppler shift there is a need of being able to reconstruct the polar angle. For experiments where only the energy is specifically of interest, there is no need of being able to reconstruct the azimuthal angle. In this case, it is desirable to not waste any of the capacity of the network to reconstruct the azimuthal angle if it will affect the reconstruction of the energy and polar angle negatively.

To test this, two cases were compared; where the azimuth weighting λ_ϕ was set to zero and one, respectively. Setting λ_ϕ to zero ensures that the network does not take the azimuthal angle into account in the training process. In this case, the standard deviation in energy and polar angle was 0.592 MeV and 0.303 rad. Analogous for $\lambda_\phi = 1$ the standard deviations were 0.597 MeV and 0.331 rad, respectively.

The networks are able to reconstruct the energies and polar angles almost equally accurate in both cases. Since reconstruction of the azimuthal angle is the most general case, using non-zero λ_ϕ is preferable. Adding the azimuthal angle to the cost function also corresponds to adding an extra spatial dimension used to distinguish and pair γ-rays in the permutation cost function. Therefore, it is possible that using the azimuthal angle can have positive effects on the reconstructions, especially for events with many γ-rays.

3.5.4 Doppler shift correction

As reviewed in section 2.1.3, the detected data from high-energy particle accelerators are affected by relativistic effects; the Doppler shift and the headlight effect. Therefore, the ANN have to be able to reconstruct data that can be successfully relativistically corrected. The following networks are based on training with relativistic data.

With a typical beam velocity of $\beta = 0.7c$, the γ-ray energies can be more than twice as high in the laboratory frame as in the beam frame. This can be seen in figure 3.8 for γ-rays reconstructed in the laboratory frame with beam frame energies between 0.1 and 15 MeV. Note that the headlight effect can be seen for the polar angle as a greater concentration of the γ-rays are detected in the forward direction. As expected, no relativistic effects apply to the azimuthal angle.

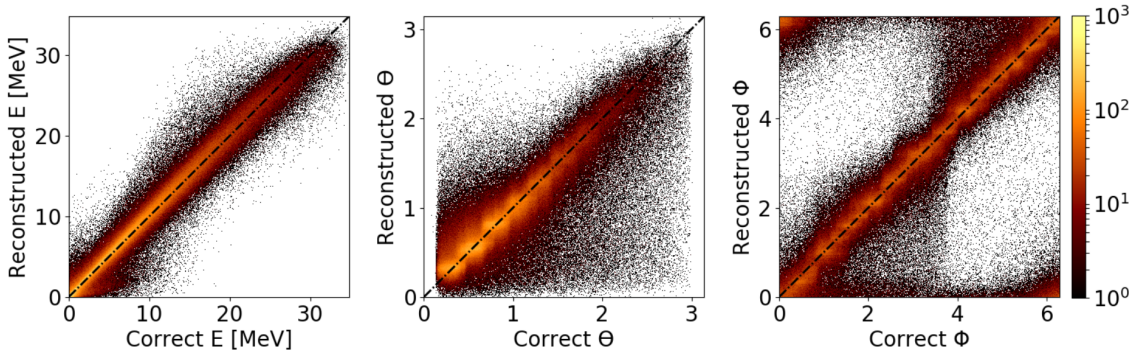


Figure 3.8: Reconstructed energies and angles in the laboratory frame for γ-rays with beam frame energy 0.1 to 15 MeV. The Doppler shift can be seen as the detected γ-ray energies vary up to almost 40 MeV. In addition, the headlight effect causes the γ-rays to be concentrated in the forward direction of the beam, which corresponds to small polar angles.

The energy in the beam frame can be reconstructed by a network using the standard cost function in equation (3.2) by training with labelled energies in the beam frame. These networks will further on be referred to as beam frame (BF) networks. Reporting the energy in the beam frame implies that the network has to use its limited capacity to implement the Doppler shift correction with its weights and biases. To relieve the network from having to learn how to perform the Doppler shift correction, alternative methods were used. In both cases, the alternative networks

were trained to provide the energies in the laboratory frame. Thereafter the Doppler shift correction was applied before evaluation with energies in the beam frame. Note that the correction is now dependent on the reconstructed polar angle, which adds an extra uncertainty to the reconstruction. This network approach is referred to as the laboratory frame (LF) networks.

Even if it is more straightforward for the network to reconstruct the energies in the laboratory frame it is the reconstruction of the energies in the beam frame that is of interest to optimise. Therefore, an alternative relativistic cost function was used;

$$C(\mathbf{y}, \hat{\mathbf{y}}) = \sum_{i=1}^{n_\gamma} \left[\lambda_E \left(\frac{1 - \beta \cos\theta}{\sqrt{1-\beta^2}} E_i - \hat{E}'_i \right)^2 + \lambda_\theta \left(\Delta\theta_i\right)^2 + \lambda_\phi \left((\Delta\phi_i + \pi) \bmod 2\pi - \pi \right)^2 \right]. \tag{3.3}$$

Networks using this cost function will be called relativistic cost function (RCF) networks. Note that E_i is in the laboratory frame while $\hat{E}'_i$ is in the beam frame. Here the Doppler shift correction of the network output is applied in the training process and, therefore, it is the reconstruction of the energies in the beam frame that will be optimised. Since the network still provides the energies in the laboratory frame the Doppler shift correction has to be applied before evaluation.

3.5.5 Relative weighting of the cost function terms

How the different observables are prioritised by the network during the training process depends on the relative size of each term in the cost function. The weighting is controlled with the weighting factors λ_E, λ_θ and λ_ϕ. Tests were performed by varying the weighting factors. Since the weighting of the energy, polar and azimuthal angles are relative to each other, it was decided to keep $\lambda_\theta = 1$ fixed as a reference.

When using relativistic data, an extra connection is introduced through the Doppler shift, relating the reconstructed energies and polar angles. The standard deviations for different combinations of λ_E and λ_ϕ for the cost function in equation (3.2) with the BF networks is plotted in figure 3.9. It was observed that the mean error in the polar angle is strongly affected by the weighting, following a similar dependency of λ_E and λ_ϕ as the polar angle standard deviation.[3]

[3]The networks were here evaluated with energies up to 10 MeV, with the standard deviation computed up to 15 MeV. This will unfortunately reduce the average standard deviation, but is still a sufficient measurement of the performance for this purpose.

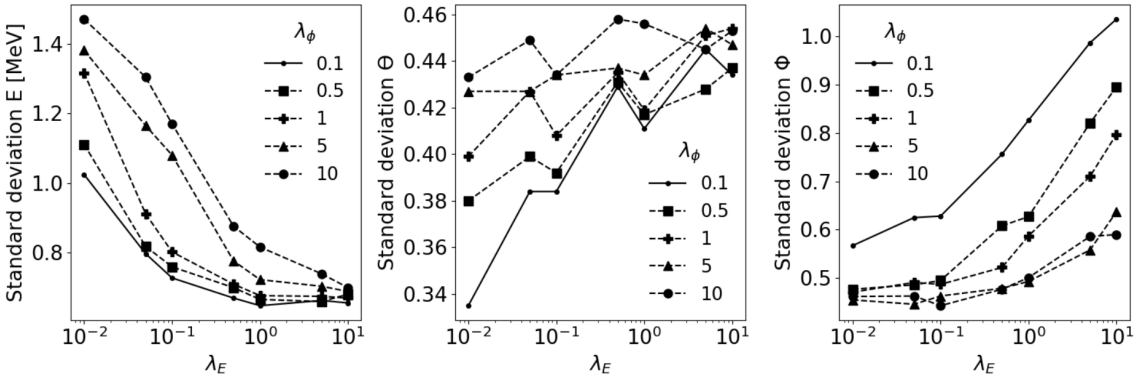


Figure 3.9: The standard deviations of the energy and angles for varying values of λ_E and λ_ϕ with $\lambda_\theta = 1$. The absolute cost function in equation (3.2) was used by the BF network. The favored choice of λ_ϕ is marked with a solid line.

By analysing the standard deviation it was assessed that $\lambda_E = 0.1$ and $\lambda_\phi = 0.1$ were optimal as weighting factors. The data corresponding to the latter is marked with a solid line in figure 3.9. Even if $\lambda_\phi = 0.1$ generates the greatest standard deviation in the azimuthal angle, this was deprioritised in favor of the energy and polar angle reconstruction with the polar angle mean error taken into account.

A corresponding weighting analysis was performed for the RCF network with the cost function in equation (3.3) applying the Doppler shift. The associated standard deviations are presented in figure 3.10. The results are similar to those in figure 3.9 with the major difference found in the polar angle standard deviation for $\lambda_E \gtrsim 1$. Therefore, $\lambda_E = 0.1$ and $\lambda_\phi = 0.1$ were also here assessed as the optimal choices of weighting.

The performance of networks with non-relativistic data for different parameter values is reviewed in appendix C.4. It was concluded that $\lambda_E = \lambda_\theta = \lambda_\phi = 1$ were optimal choices for the standard cost function, see equation (3.2).

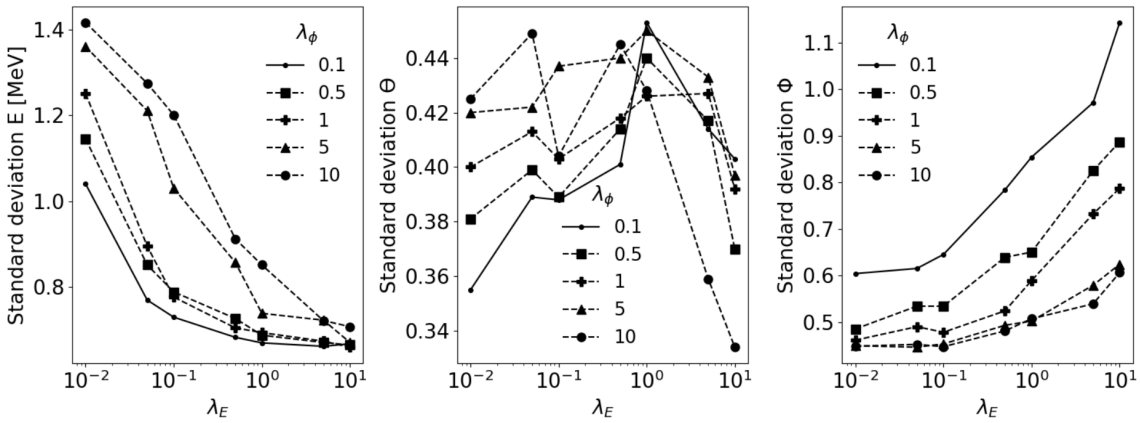


Figure 3.10: The standard deviations of the energy and angles for varying values of λ_E and λ_ϕ with $\lambda_\theta = 1$. The RCF network with the cost function in equation (3.3) applying the Doppler shift was used. The favored choice of λ_ϕ is marked with a solid line.

3.6 Classification of γ-rays

As discussed is section 3.3.1, the output of the network has to be of fixed length, independent of the multiplicity of each individual event. This led to the introduction of so-called zero, or non-existing, γ-rays with energies and angles all set to zero to fill the arrays. The reconstructed energies of these were often low, but never exactly zero, by which they will be mistaken for real γ-rays when using real, unlabelled detector data. Thus the networks have to be able to determine which γ-rays that are non-existing.

3.6.1 The classification node

For the networks to be able to label the γ-rays a classification node μ was added to the output for each photon;

$$y = \Big(E_1,\ \theta_1,\ \phi_1,\ \mu_1,\ ...,\ E_{n_\gamma},\ \theta_{n_\gamma},\ \phi_{n_\gamma},\ \mu_{n_\gamma}\Big)\,. \tag{3.4}$$

The correct label nodes corresponding to μ were set to zero for non-existing γ-rays and one for the non-zero γ-rays. The reconstructed value of the classification node was therefore rounded to one or zero which indicates whether the γ-ray exists or not. Two terms were added to the cost function in equation (3.2); a mean square error for training the classification performance, and a term heavily punishing the network for pairing a real γ-ray with a non-existing one:

$$\begin{aligned} C(\mathbf{y}, \hat{\mathbf{y}}) = \sum_{i=1}^{n_\gamma} \Bigg[& \lambda_E \left(\Delta E_i\right)^2 + \lambda_\theta \left(\Delta\theta_i\right)^2 + \lambda_\phi \left((\Delta\phi_i + \pi) \bmod 2\pi - \pi\right)^2 \\ & + \lambda_\mu \left(\Delta\mu_i\right)^2 + 100 \cdot \mathrm{round}(|\Delta\mu_i|) \Bigg]. \end{aligned} \tag{3.5}$$

The pairing term added a cost much larger than the other terms, resulting in that the classification was prioritised in the pairing process.

By adding the classification node, new measurements of performance were introduced. Naturally, the amount of correctly classified γ-rays was determined. The other measurement was the amount of correctly classified event multiplicities. Networks with the cost function in equation (3.5) were trained with different λ_μ between 1 and 100. Here $\lambda_E = 0.1$, $\lambda_\theta = 1$ and $\lambda_\phi = 0.1$ was used. The results are presented in the tables C.1 and C.2 in the appendix. All of the networks classified more than 92 % of the γ-rays correctly by which the correct event multiplicity was determined in more than 79 % of the cases. No major differences in classification performance was observed from different choices of λ_μ. From the standard deviations of the reconstructions it was clear that the reconstructions got less accurate with rising λ_μ.

A modified cost function was also tested, where the cost from the energy and angles was weighted with the classification node:

$$C(\mathbf{y}, \hat{\mathbf{y}}) = \sum_{i=1}^{n_\gamma} \Bigg[\Big(\lambda_E \left(\Delta E_i\right)^2 + \lambda_\theta \left(\Delta\theta_i\right)^2 + \lambda_\phi \left((\Delta\phi_i + \pi) \bmod 2\pi - \pi \right)^2 \Big) \cdot {\mu_i}^2 + \lambda_\mu \left(\Delta\mu_i\right)^2 + 100 \cdot \mathrm{round}(|\Delta\mu_i|) \Bigg]. \tag{3.6}$$

The networks were then not forced to reconstruct the energy and angles for γ-rays that the network was certain were non-existing. Note that the networks would benefit from tending to classify as many γ-rays as possible as non-existing. The results from usage of cost function in equation (3.6) is presented in the tables C.1 and C.2 in the appendix. With the cost function in equation (3.6), the networks showed tendency to be better at correctly classifying zero γ-rays but worse at the real ones compared to when using the cost function in equation (3.5). The correctness of all of the γ-rays, and therefore also the multiplicity, tended to be slightly worse than with the cost function in equation (3.5).

3.6.2 Training with additional multiplicities

As discussed in section 3.4.4, the neural networks will be encouraged to only reconstruct event properties similar to those that it has seen during the training process. This causes an undesired reconstruction artefact, the dip in energy in the end of the energy interval. Similar effects might appear when training the networks with data of the same event multiplicity as used for the evaluation. One major challenge for the networks is to determine how many γ-rays that were present in each event. The networks have to be as generalisable as possible to varying multiplicities, including when there are no detected γ-rays at all.

In an attempt to improve the classification performance of the networks, events with multiplicity four and zero were added to the training data, where the latter corresponds to events without γ-rays. The classification performance of these networks is presented in table 3.1. A network both trained and evaluated with multiplicity zero to three was also tested. Note that in table 3.1 the amount of zero γ-rays in the evaluation data is varying with the multiplicities used in the training process.

As observed in section 3.6 the networks tend to be better at classifying the zero γ-rays than the existing ones. By comparing the networks trained with multiplicity 1–4 and 0–4, training with zero multiplicity might benefit the classification of zero γ-rays. This is contradicted by networks trained with multiplicity 1–3, which are better at classifying zeros than the 0–3. However, this result might be affected by the training and evaluation on the same multiplicities, as in the case of multiplicity 1–3. Adding multiplicity four to the training data tends to benefit the classification of non-zero γ-rays. Although, adding extra output nodes, as in the case of the networks trained with multiplicity four, is a disadvantage for the multiplicity determination. This is expected, since the networks have to classify more γ-rays correctly to get the right multiplicity. In total it does not seem like there would be a major worsening effect in classification by generalising the networks to greater multiplicities. When also adding zero multiplicity to the evaluation, 0-3*, the network finds it easy to get the zero γ-ray events right, by which both the correctness of the non-existing γ-rays and multiplicity is raised.

Table 3.1: The classification performance of the networks trained with different multiplicities (Trained mult.) and evaluated with multiplicity one to three. The data presented for trained multiplicity 0-3* is evaluated with multiplicities zero to three. The maximum number of γ-rays in the network output is specified in parentheses. As is also the proportion of zero γ-rays in the evaluation data.

Trained mult.	Correct zero γ	Correct real γ	Total correct γ	Correct mult.
1–3 (3)	96.9 % (33 %)	90.8 %	92.8 %	79.0 %
0–3 (3)	96.0 % (33 %)	91.7 %	93.2 %	79.9 %
1–4 (4)	95.8 % (50 %)	93.2 %	94.5 %	78.5 %
0–4 (4)	96.3 % (50 %)	92.7 %	94.5 %	78.5 %
0–3* (3)	98.0 % (50 %)	91.7 %	94.9 %	84.9 %

The mean error and standard deviation of the networks trained with varying multiplicities are presented in table 3.2. The errors for the reconstructed zero γ-rays have not been taken into account. It seems like the networks trained with multiplicity four have the least standard deviations for all three quantities. Adding multiplicity zero instead has a worsening effect. For a review of the uncertainties in the classification and standard deviation precisions, this is presented in section 4.2.

Table 3.2: The mean standard deviation (Std) in the energy and angles reconstructions for the networks trained with different multiplicities (Trained mult.) and evaluated with multiplicity one to three. The data presented for trained multiplicity 0–3* is evaluated with multiplicities zero to three.

Trained mult.	Std E [MeV]	Std θ	Std ϕ
1–3	1.108	0.405	0.701
0–3	1.129	0.408	0.679
1–4	1.055	0.382	0.679
0–4	1.086	0.379	0.682
0–3*	1.129	0.408	0.679

3.7 Hyperparameters of the convolutional neural networks

There are numerous possible parameters to tune in the convolutional architecture of the network. The general structure of the tested networks is visualised in figures 3.2 and 3.11, where each layer C_{ij} has a corresponding kernel size. In the first layer of each block, the kernel size is chosen such that it covers the neighbours of every crystal it convolutes over. This size depends then on whether it is an A or D crystal convolution. The number of layers, the amount of kernels and the kernel size are all dependent on each other and therefore tricky to test individually. Tests were done to try and figure out which setups worked best.

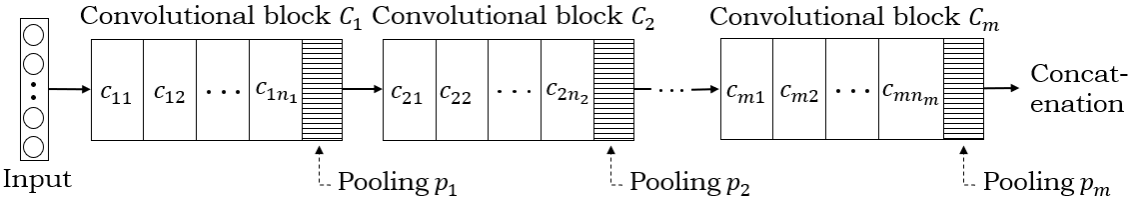


Figure 3.11: A schematic of the general architecture of the convolutional layer structures for either the A or D crystal in figure 3.2. The layers are grouped as blocks C_i with internal convolutional layers c_{ij} where $i = 1, ..., m$ and $j = 1, ..., n$. In the end of each block there is an optional pooling layer, p_i. After the last convolutional block the concatenation follows before the information is passed to the fully connected layers.

Table 3.3 lists the different network structures used in the testing of the hyperparameters. The kernels per layer parameter is the number of convolutional kernels in a network layer and convolutional blocks is the number of serially connected blocks of convolutional layers in the network. Moreover, the table includes the number of fully connected (FC) layers after the last convolution block as well as the number of nodes per FC layers.

Table 3.3: The different CNN structures used in the testing of the network hyperparameters. Each column represents one tested network and the first row is the identification labels of the networks. The different hyperparameters are listed in the first column. The dash indicates the parameter that is being tested.

Network label	**NOK**	**KS**	**PL**	**ND**
Pooling type	max	max	-	off
Pooling filter size	1	1	-	off
Pooling padding	valid	valid	-	off
Kernel size	3	-	3	3
Kernels per layer	-	16, 32, 64	15, 20, 25	64, 64, 64
Layers per block	2	2	3	2
Convolution blocks	3	3	3	-
FC layers	4	3	3–5	4
Nodes per FC layer	3·1024, 256	2·1024, 256	4·1024, 256	4·512

3.7.1 Pooling layers

As explained in section 2.3.5 the two types of pooling layers, max and average, have a size and a padding; same or valid. The network PL in table 3.3 tested all possible combinations of these two parameters for pooling sizes one to seven. Networks with max pooling and valid padding tended to make more accurate reconstructions. With max pooling and valid padding, the mean standard deviation is plotted in figure 3.12 for networks with varying pooling layer size and one network without pooling. From these results it was concluded that pooling did not have any positive effect on the reconstruction.

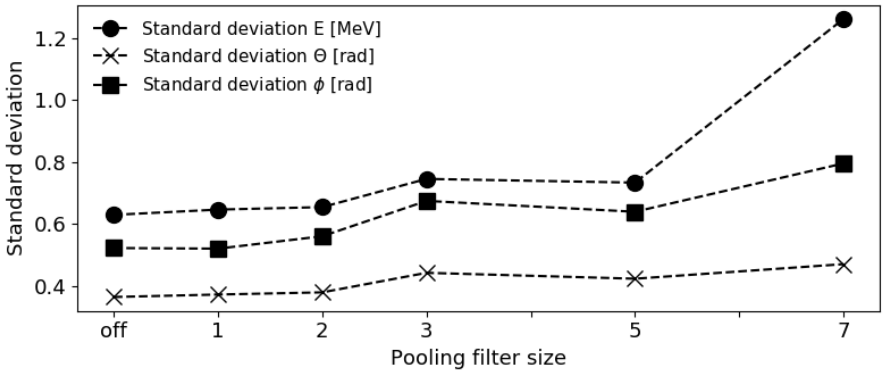


Figure 3.12: The mean standard deviations for networks with pooling layers of sizes between one and seven and when pooling was not applied (off). Not using pooling at all generated the smallest mean standard deviation.

3.7.2 Kernel size

Network KS in table 3.3 was used to test the kernel size. In the first layer of the entire convolutional neural network the kernel size is always equal to the size of the

neighbour clusters in the input. This makes sure that the neighbour information is preserved when the convolution operation is applied. In any following layers, this is no longer true and the kernel size parameter applies instead.

Different kernel sizes between 1 and 30 was tested. The results showed that varying the kernel sizes between convolution blocks had a much weaker effect than varying the overall kernel size of the network. Thus, for simplicity only the networks with the same kernel size through the whole network are represented in figure 3.13.

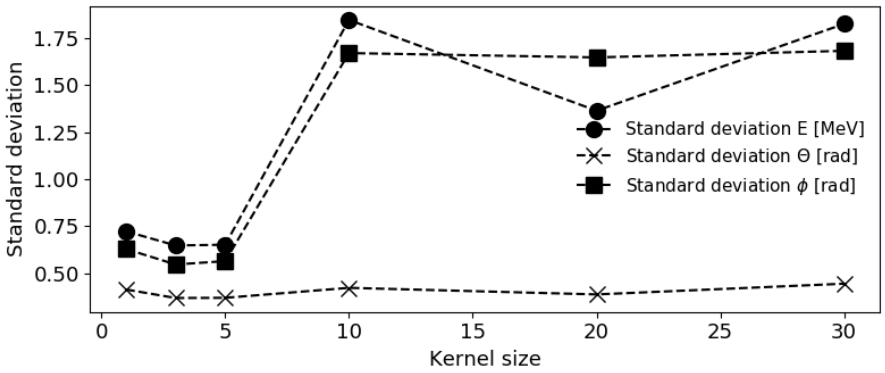


Figure 3.13: The mean standard deviations for six similar networks with different kernel sizes. With kernel sizes of 10 and greater the networks are unable to predict anything useful. The deviations in energy and ϕ are quite similar while the deviations in θ is barely noticeable.

It is clear from figure 3.13 that small kernel sizes perform better. The results indicate that a fixed kernel size of three is favourable. Kernel sizes of ten and greater did not produce any acceptable predictions.

3.7.3 Number of kernels

The total number of kernels was tested on network structure NOK in table 3.3. The effect of the kernel number can be seen in figure 3.14, where the standard deviations for each network is shown as a function of the number of kernels. As the networks grow larger the standard deviations for E and θ increase and reach unacceptable levels while they remain unchanged for θ.

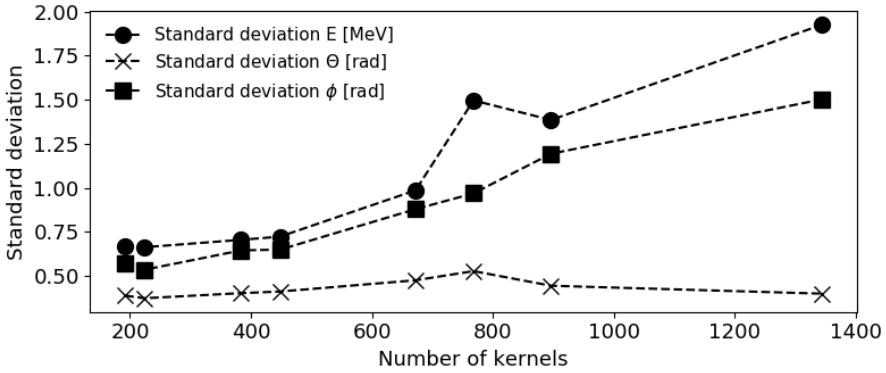


Figure 3.14: The standard deviations for different number of convolution kernels. shallow networks with fewer kernels tend to be more accurate than the wider ones. For large number of kernels the networks fail to predict anything useful. Note that the standard deviation for θ did not change much throughout the test.

3.7.4 Network depth

The network depth is typically defined as the total number of layers. In order to test how the network performance scales with the network depth, a test was designed that varied the number of convolutional blocks. The network specifications are labelled as ND in table 3.3. Figure 3.15 shows how the standard deviations vary for the different network depths. Deeper networks did not improve the results and three convolution blocks was chosen for the majority of the subsequent tests. The number of fully connected layers after the last convolution block was also tested. The results were very similar to those in figure 3.15, i.e. the difference was small and three layers was the best choice.

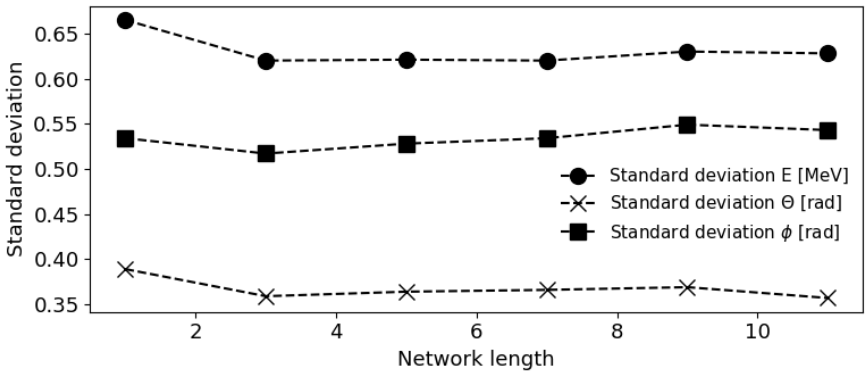


Figure 3.15: The standard deviations for networks with different number of convolutional blocks, each consisting of two convolutional layers per block. The results show that the network depth does not affect the performance to a high degree.

4
Results

The aim of this study is to compare addback with the reconstruction performed by ANN. The method for analysis and comparison is inspired by Olander *et al.* [4].

4.1 Comparison with the addback routine

The addback is the current method for reconstructing the energy and emission angles of the γ-rays. Using more realistic data, four types of neural networks will be compared and evaluated against the neighbour routine, which is explained in section 2.2. The following networks were used:

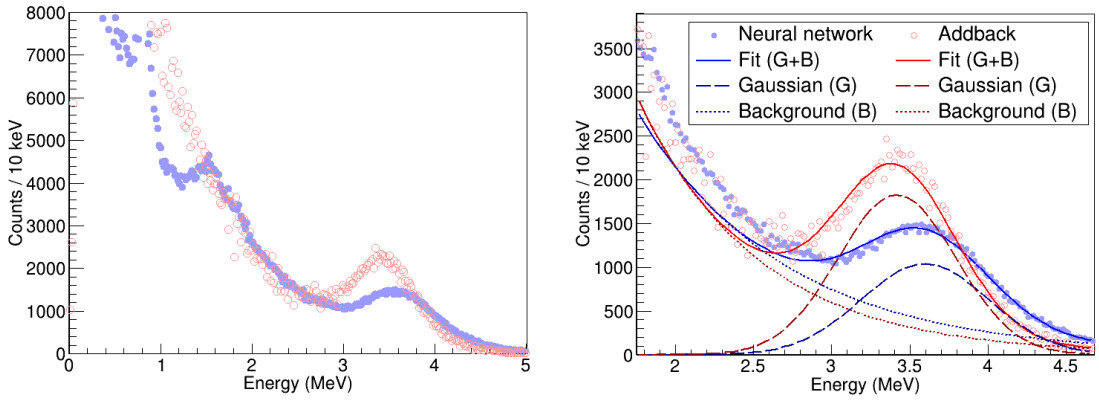
Standard FCN: A fully connected neural network with the standard cost function from equation (3.2). It was trained on non-relativistic data with a multiplicity of one to three with γ-rays of energy up to 15 MeV. The network consists of 10 layers with 128 nodes in each one.

LF Network: The same network as the standard FCN, with the only exception that it is trained on relativistic data with beam frame energies up to 15 MeV.

RCF Network: Same network as the LF network, but it uses the cost function in equation (3.3) which explicitly implements the Doppler-shift correction.

Standard CNN: A convolutional neural network with the same cost function as standard FCN. It consists of a six convolutional layers, where the first and second layer have 16 kernels, the third and fourth have 32 kernels and the last two have 64 kernels. Each kernel has a length of three and the stride size is set to one. After the convolutional layers comes three fully connected layers with 1024, 1024 and 256 nodes, respectively. No pooling is applied.

The realistic data used for the final evaluation consists of events with up to multiplicity three. There is a certain probability to have one Doppler-shifted γ-ray with a fixed energy and up to two isotropic background γ-rays with different energies distributed between 0 and 2 MeV, which are not Doppler-shifted. The probabilities of being emitted are 10 % and 50 % for the Doppler-shifted γ-ray and the background γ-rays, respectively. Furthermore, the relativistic speed of the Doppler-shift was $\beta = 0.7$. The evaluation is based on how accurately the addback and the neural network can reconstruct the Doppler-shifted γ-ray.



(a) **(b)**

Figure 4.1: The beam frame energy spectrum reconstructed by addback and the LF network from events with Doppler-shifted γ-rays at 3.5 MeV and the background γ-rays. Figure (b) zooms in at 3.5 MeV from figure (a) and shows the fitted Gaussian and descending exponential curve.

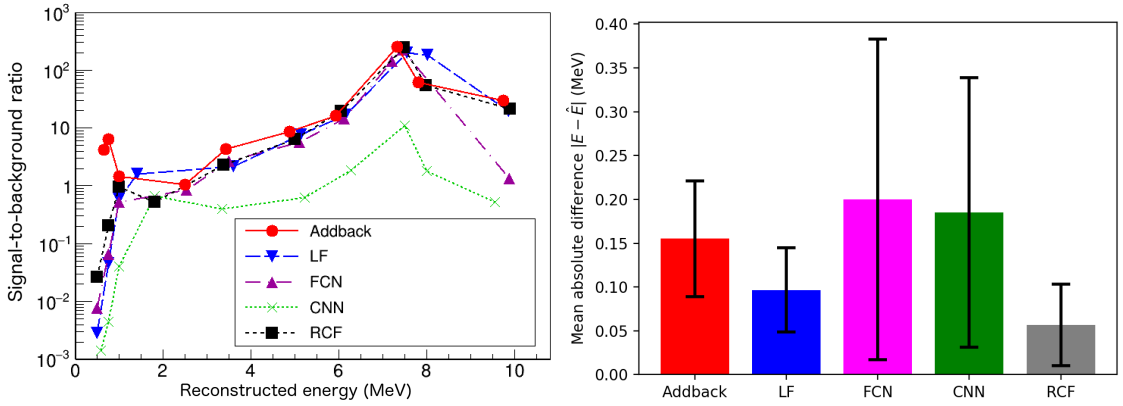
Figure 4.1 shows the addback comparison, here an evaluation of both the addback and the LF network where the energy of the Doppler-shifted γ-ray is 3.5 MeV. The background is separated from the signal by fitting a Gaussian distribution f_{Gaussian} corresponding to the signal and a descending exponential curve $f_{\text{exponential}}$ to the background, as illustrated in (b). Observe that the background energy interferes and makes it hard to reconstruct low energies. Note the hump between 1 to 2 MeV in figure 4.1a, it emerges as an artefact from how the background γ-rays were generated but could be falsely interpreted as a γ-ray of interest.

To decide whether the Doppler-shifted γ-rays actually have been reconstructed among the background γ-rays, a signal-to-background ratio R can be used. The ratio is defined as

$$R = \frac{\int_{\mu-\sigma}^{\mu+\sigma} f_{\text{Gaussian}}(E)dE}{\int_{\mu-\sigma}^{\mu+\sigma} f_{\text{exponential}}(E)dE}. \tag{4.1}$$

where μ and σ are the mean and standard deviation of the Gaussian distribution. A high ratio would imply that the reconstruction is successful and not mistaken for a false detection.

Two comparisons are made between the addback and the neural networks. The first is to use the signal-to-background ratio as a comparison between the addback and the neural network. This has been done with the Doppler-shifted γ-ray having fixed energies between 1 and 10 MeV in the beam frame. The second comparison is done by observing the absolute difference between the reconstructed energy E and the actual energy $\hat{E}$ of the Doppler-shifted γ-ray. The reconstructed energy E is chosen as the mean μ of the fitted Gaussian peak, which in our analysis works well for energies equal to and above 3.5 MeV. Both comparisons are illustrated in figure 4.2.



(a) **(b)**

Figure 4.2: Figure (a) shows the signal-to-background ratio for evaluation of energies between 1 to 10 MeV in beam frame. Figure (b) illustrates the mean and standard deviation of the absolute difference between the reconstructed and actual energy, taken from evalutation of the energies 3.5 to 10.0 MeV.

The signal-to-background ratio suggests that the networks have more difficulty to reconstruct lower energies since the background γ-rays dominate in the range 0 to 2 MeV. This is however not the case for the addback. For energies between 2 and 10 MeV, addback seems most often to have a higher ratio, but the neural networks do not perform significantly worse. The only exception is the convolutional network which appear mediocre with a low ratio for all energies.

The mean absolute difference in figure 4.2b shows the accuracy of the reconstructions. The differences are taken from evaluation of 3.5 to 10 MeV in the beam frame. Any lower energies were not taken into account since the neural networks failed to reconstruct them. Both the LF and RCF networks gives a lower mean than the addback while all of their standard deviations are similar. Meanwhile, the standard FCN and CNN are not as accurate as the others.

The best performance is shown by the RCF and LF networks which were trained on relativistic data. They had a mean absolute difference of $5.6 \cdot 10^{-2}$ and $9.6 \cdot 10^{-2}$, whereas the addback had a mean of $15.4 \cdot 10^{-2}$. All of these had standard deviations of approximately $\sim 5 \cdot 10^{-2}$. This makes it hard to select a clear winner but the results from the neural networks are promising. However, there are still problems with reconstructing low energies where the addback performs better. Finally, comparing FCN and CNN gives the impression that convolutional network structures need significant improvements to achieve a good performance when looking at the signal-to-background ratio which implies that the reconstructions might not generalise to different detector data.

4.2 Uncertainty analysis of training and evaluation

To assess the consistency of the performance, two different networks have been trained ten times each with fixed parameter settings. The first network was the BF network with the standard cost function, weighted with $\lambda_E = 0.1$, $\lambda_\theta = 1$ and $\lambda_\phi = 0.1$. The gathered data of the measurements of performance for this network are presented in table 4.1. The conclusion is that for the standard deviations, the maximum and minimum deviation from the average values are approximately 5 %.

Table 4.1: The variation in standard deviations (std) for ten identically trained networks with relativistic data without the relativistic cost function. The maximum and minimum values are also presented relative to the averages in parentheses.

	Energy std [MeV]	Polar angle std	Azimuthal angle std
Average	1.080	0.397	0.661
Maximum	1.124 (104 %)	0.408 (103 %)	0.699 (106 %)
Minimum	1.061 (98 %)	0.379 (95 %)	0.631 (95 %)
Std. dev.	0.021	0.011	0.024

The second network was also using the classification node. Corresponding data for the accuracy of the classification node is presented in table 4.2 for the network trained on relativistic data with the standard cost function 3.2. The classification of the zero and non-zero γ-rays had a maximum error from the average of about 0.6 percentage points and 0.2 percentage points for all of the γ-rays. The maximum error for the multiplicity was instead 0.6 percentage points.

Table 4.2: The variation in the classification for ten identical networks with the classification node.

	Correct zero γ	Correct non-zero γ	Total correct γ	Correct mult.
Average	95.9 %	91.9 %	93.2 %	80.3 %
Maximum	96.5 %	92.4 %	93.4 %	80.7 %
Minimum	95.3 %	91.4 %	93.0 %	79.7 %
Std. dev.	0.40 %	0.35 %	0.14 %	0.34 %

5
Discussion

The results of the comparison between the neural networks and the addback showed that the addback still might have better performance with an overall better signal-to-background ratio and a reasonable accuracy. Meanwhile, the neural networks were showing better accuracy at higher energies but had a worse ratio. However, issues remain at reconstructing low energies where the neural networks could not show any improvement at all when compared to the addback.

5.1 Classification

It is of great importance to be able to determine the amount of detected γ-rays in an event, as reviewed in section 3.6. By introducing the classification node, existing and non-existing γ-rays were classified with an accuracy of more than 90 % for multiplicity one to three. In addition, the same networks were also capable of reconstructing the energy and emission angles of the γ-rays. Since the training time increased substantially with event multiplicity because of the permutation cost function, tests with greater multiplicities were not investigated. To determine the multiplicity alone, the approach of Olander *et al.* [4] was instead to use classification networks, without utilising a permutation cost function. This type of network was not limited by the event multiplicities as those with the classification node. However, the ability to classify each individual γ-ray is desirable and the classification can possibly be used to improve the reconstruction of the energy and angles.

5.2 Fully connected and convolutional neural networks

One important question to discuss is why the fully connected neural networks perform better than the convolutional neural networks. In theory convolutional neural networks look promising since the event reconstruction problem is similar to image recognition, a problem that convolutional neural networks are commonly applied to. However, the convolutional neural networks will rely heavily on associating the structure of the network with the geometry of the detector since the convolutional and pooling kernels operate locally.

Most of our tests on CNN show that the performance of the networks increases up to a certain level of network complexity after which it starts to decrease. The results from investigating the depth of CNN show that networks can be expanded in length without losing performance. When studying number of kernels, kernel size

and effects of pooling, results showed that increasing these parameters would have a negative effect on the reconstruction. The cause of this could be that information is not propagating correctly through such networks. It could be explained by the way the convolutions are done in the intermediate layers. In contrast to the first layer which has a matching kernel size and stride as the size of the neighbour clusters, the intermediate layers may distort this by a poor choice of kernel parameters.

These problems do not arise for the fully connected networks. This could depend on the fact that in practice it is easier to implement a fully connected neural network for event reconstruction as these do not rely on the same geometrical interpretation when designing the structure of the network. This might explain the difference between the convolutional and fully connected networks in figure 4.2a.

5.3 Conclusion

In the thesis, different approaches have been investigated to design artificial neural networks that are able to identify and reconstruct the deposited energies of γ-rays. The following is concluded:

- The constructed artificial neural networks show promising results by possibly obtaining an accuracy that is equal to or better than addback in the range of 3.5 to 10 MeV. However, problems remain with reconstructing γ-rays at lower energies.
- Convolutional neural networks currently perform worse with reconstruction in comparison to fully connected neural networks.
- By introducing a classification node it was possible to distinguish existing γ-rays from non-existing ones.

Beyond above-mentioned conclusions, we would like to emphasise some minor results. Firstly, using training data with γ-rays of greater energies than the networks were evaluated with was shown to be necessary and should be used in the future. Moreover, the permutation cost function is so far the best alternative when reconstructing the energies and emission angles. Finally, it is possible to reconstruct the azimuthal angle with a negligible deterioration in reconstruction of the energy and polar angle.

In summary, the outcome of this thesis shows potential, but further investigation of its possibilities is needed to reach the goal of improving event reconstruction.

5.4 Further investigations

A suggestion is to investigate how to, as efficiently as possible, use the information of the classification node to improve the reconstruction of the energies and angles. For example, the weighting term in the cost function in equation (3.5) was used to prioritise the classification node in the pairing process. Thus, the information

contained in the reconstruction of the energy and the angles could potentially be improved by fine-tuning the weighting term. Using the classification node for CNN could possibly also improve the performance. This may be a successful approach since the strength of CNN are to detect local features.

As investigated by Olander *et al.* [4], a classification network can be used to determine the multiplicity, where the output is a probability distribution of the possible multiplicities. This could be implemented as a neural network connected in series by first letting a network specialise on determining the event multiplicity. The output could then be combined with the original detector data as input to a network reconstructing the energies, angles and the γ-ray classification. Then, it would be possible to use the major qualities of CNN to detect γ-rays and at the same time relieve the capacity of the latter network.

An important aspect of building a convolutional neural network is the clustering procedure of the detector crystals when creating the input to the network. It would be of further interest to develop this concept by studying alternatives and eventually finding new ways of handling the detector geometry. For example, it could be possible to include different number of neighbours in the clusters or investigate if the CNN may recognise similar events which express rotational and/or mirror symmetries.

To obtain further optimised networks, more testing on how the network structures affect the performance should be considered. Furthermore, it would also be of interest to investigate physically motivated cost functions such as the relativistic cost function in equation (3.3). For example, it would be meaningful to weight the azimuthal cost term by $\sin\theta$ since the azimuthal angle is of less importance when the polar angle is close to 0 or π. There is also the unsolved problem of why some polar and azimuthal angles are reconstructed too low, which is the origin of the vertical line in the azimuth reconstruction.

Bibliography

[1] T. Aumann et al., *Technical Report for the Design, Construction and Commissioning of The CALIFA Barrel: The R3B CALorimeter for In Flight detection of γ rays and high energy charged pArticles*. 2011.

[2] S. Lindberg. *Optimised Use of Detector Systems for Relativistic Radioactive Beams*. M.S. thesis, Dept. Physics, CTH, Sweden, 2013.

[3] I. Goodfellow, Y. Bengio and A. Courville. *Deep Learning*. MIT Press, 2016. [Ebook] Availabe: `http://www.deeplearningbook.org`. [Accessed: 2019-05-12].

[4] J. Olander, M. Skarin, P. Svensson and J. Wadman. *Rekonstruktion från detektordata med hjälp av neurala nätverk: En studie i tvärsnittet mellan kärnfysik och maskininlärning*. B.S. thesis, Dept. Physics, CTH, Sweden, 2018.

[5] R. S. Simon. The Darmstadt-Heidelberg Crystal Ball. Journal de Physique Colloques, 1980, 41 (C10), [Online] Available: `https://hal.archives-ouvertes.fr/jpa-00220650/document`. [Accessed: 2019-05-12].

[6] B. R. Martin. *Nuclear and Particle Physic*, John Wiley & Sons, 2006.

[7] W. R. Leo. *Techniques for nuclear and particle physics experiments: a how-to approach*. 2:nd ed., Springer, 1987.

[8] *Experimental Setup*, GSI. [Online] Available: `https://www.gsi.de/work/forschung/nustarenna/nustarenna_divisions/kernreaktionen/research_program/reactions_with_exotic_nuclei/experimental_setup.htm`. [Accessed: 2019-05-12].

[9] R. Thies. *Prototype tests and pilot experiments for the R3B scintillator-based detection systems*. M.S. thesis, Dept. Physics, CTH, Sweden, 2011.

[10] H. T. Johansson & A. Heinz, *K8 - Coincidence Measurements*. [Online] Available: `https://pingpong.chalmers.se/courseId/10640/node.do?id=5227077&ts=1552134259502&u=-1479464541`. [Accessed: 2019-05-12].

[11] W. Rindler, *Relativity: Special, General and Cosmological*, 2:nd ed., New York, NY, Oxford University Press Inc., 2006.

[12] S. Agostinelli et al. *Geant4—a simulation toolkit*. Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment. vol 506, pp. 250-303,

July 2003. [Online] https://www.sciencedirect.com/science/article/pii/S0168900203013688. [Accessed: 2019-05-12].

[13] H. T. Johansson, *The ggland command-line simulation wrapper - Long write-up — documentation and manual*. 2013.

[14] *ROOT Data Analysis Framework - User's Guide* May, 2018. [Online] Available: https://root.cern.ch/root/htmldoc/guides/users-guide/ROOTUsersGuide.html#introduction. [Accessed: 2019-05-12].

[15] I. Kiral-Kornek et al. *Epileptic Seizure Prediction Using Big Data and Deep Learning: Toward a Mobile System*. EBioMedicine, vol. 27, pp. 103-111, January 2018. [Online] Available: https://www.ebiomedicine.com/article/S2352-3964(17)30470-X/fulltext. [Accessed: 2019-05-12].

[16] S. Theodoridis, *Machine Learning - A Bayesian and Optimization Perspective*, Ofxford, Academic Press, 2015.

[17] A. L. Maas, A. Y. Hannun and A. Y. Ng, *Rectifier Nonlinearities Improve Neural Network Acoustic Models*, 2013. [Online] Available: https://ai.stanford.edu/~amaas/papers/relu_hybrid_icml2013_final.pdf. [Accessed: 2019-05-12].

[18] S. Ruder, *An overview of gradient descent optimization algorithms*, 2016. [Online] Available: https://arxiv.org/abs/1609.04747 [Accessed 2019-05-15]

[19] A. Krizhevsky, I. Sutskever, G. Hinton. *ImageNet Classification with Deep Convolutional Neural Networks*. Curran Associates, Inc., 2012. [Online] Available: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf. [Accessed: 2019-05-14].

[20] *Kebnekaise*, High Performance Computing Center North website, May 10, 2019, [Online] Available: https://www.hpc2n.umu.se/resources/hardware/kebnekaise. [Accessed: 2019-05-12].

[21] Overview, TensorFlow, [Online] Available: https://www.tensorflow.org/overview/. [Accessed: 2019-05-13].

[22] D. P. Kingma and J. L. Ba, *Adam: a method for stochastic optimization* at International Conference on Learning Representations (ICLR), 2015. [Online] Available: https://arxiv.org/pdf/1412.6980.pdf [Accessed: 2019-05-15]

[23] H. T. Johansson *Command line simulations*, 2012. [Online] Available: http://fy.chalmers.se/~f96hajo/shows/htj_dec2012_cmdline_sim.pdf. [Accessed: 2019-05-13].

[24] M. A. Nielsen, *Neural Networks and Deep Learning*, Determination Press, 2015, [Online] Available: http://neuralnetworksanddeeplearning.com/ [Accessed: 2019-05-14].

A
Derivation of back propagation

The following derivation and explanation of the back propagation is inspired by [24].

Observe z_i as defined in equation (2.2). To keep track of the layers it will be needed to add another index l such that z_i^l where $l = 1, ..., m$ is the index of the current layer. The expression for z_i^l will then be

$$z_i^l = \sum_{j=0}^{n} w_{ij}^l y_j^{l-1} + b_i^l \tag{A.1}$$

where $y_j^{l-1} = g(z_j^{l-1})$ and g is the activation function of the network. Then, the cost related to of one output node for an event k is denoted as $C_i^k(y_i^m, \hat{y}_i)$ where $\hat{y}_i$ is the correct value that the network should reconstruct. The total cost is $C^k = \sum_i C_i^k$.

To minimise the cost function its gradient first needs to be computed with respect to every weight and bias in the network,

$$\frac{\partial C^k}{\partial w_{ij}^l} \quad \text{and} \quad \frac{\partial C^k}{\partial b_i^l}. \tag{A.2}$$

This will be done by utilising the chain rule and that other partial derivatives are known. Starting with the last layer m, it is possible to write

$$\frac{\partial C^k}{\partial w_{ij}^m} = \frac{\partial z_i^m}{\partial w_{ij}^m} \frac{\partial y_i^m}{\partial z_i^m} \frac{\partial C^k}{\partial y_i^m}. \tag{A.3}$$

It can immediately be seen that $\partial z_i^m / \partial w_{ij}^m = y_i^{m-1}$ from equation (A.1). Moreover, $\partial y_i^m / \partial z_i^m$ is known since the activation function g is also known which means that $\partial y_i^m / \partial z_i^m = g'(z_i^m)$. This gives the expression

$$\frac{\partial C^k}{\partial w_{ij}^m} = y_i^{m-1} g'(z_i^m) \frac{\partial C^k}{\partial y_i^m}. \tag{A.4}$$

Furthermore, observe $\partial C^k / \partial y_i^{m-1}$ which can be written in a similar way as

$$\frac{\partial C^k}{\partial y_i^{m-1}} = \frac{\partial z_i^m}{\partial y_i^{m-1}} \frac{\partial y_i^m}{\partial z_i^m} \frac{\partial C^k}{\partial y_i^m}. \tag{A.5}$$

which is equal to

$$\frac{\partial C^k}{\partial y_i^{m-1}} = w_{ij}^m g'(z_i^m) \frac{\partial C^k}{\partial y_i^m}. \tag{A.6}$$

It is now possible to see a recursive relationship between $\partial C^k/\partial y_i^{l-1}$ and $\partial C^k/\partial y_i^l$. Since the cost function is known it is possible to calculate

$$\frac{\partial C^k}{\partial y_i^m} = \frac{\partial C(y_i^m, \hat{y})}{\partial y_i^m}. \tag{A.7}$$

Finally, recursively calculating $\partial C^k/\partial y_i^l$ for $l = 1, ..., m-1$ can accordingly be used to calculate $\partial C^k/\partial w_{ij}^l$ with equation (A.4).

In a similar manner it can be shown how to compute $\partial C^k/\partial b_i^l$ using the chain rule.

$$\frac{\partial C^k}{\partial b_i^l} = \frac{\partial z_i^l}{\partial b_i^l}\frac{\partial y_i^l}{\partial z_i^l}\frac{\partial C^k}{\partial y_i^l} = g'(z_i^m)\frac{\partial C^k}{\partial y_i^m}. \tag{A.8}$$

In practice is the training usually performed in epochs with a batch of events. This is done to decrease the computational load and increase training speed. A relatively small training batch of size n is used when equations (A.2) are evaluated. An average of the partial derivatives is computed as

$$\frac{\partial C}{\partial w_{ij}^l} = \frac{1}{n}\sum_k \frac{\partial C^k}{\partial w_{ij}^l}. \tag{A.9}$$

Then, to adjust a specific parameter it is needed to multiply the gradient in respect to the parameter with a learning rate η, written as

$$\Delta w_{ij}^l = -\eta \frac{\partial C}{\partial w_{ij}^l} \tag{A.10}$$

and add Δw_{ij}^l to the old w_{ij}^l [16]. The minus sign comes from the fact that the goal is to find the direction of descent in the cost function. The same is done for the partial derivative in respect to the bias. This corresponds to one iteration in the training process where all the weights and biases get updated in this manner, this process is repeated until the networks reaches a satisfactory performance.

B
Distribution of the error

Introduced in section 3.4.3, the used measurement of performance is based on the standard deviation and mean of the quantity errors. The standard deviations are plotted in figure B.1 for the intervals [4.95, 5.15] MeV, [1.57, 1.63] radians and [0.79, 0.82] radians for the BF network. The normal distribution corresponding to the mean error and mean standard deviation is also plotted. For both the energy and angles, it is clear that the errors decrease faster than the normal distribution. As investigated by Olander *et al.* [4], the Student's t-distribution might have a better fit to the errors. However, as long as the standard deviation is used only for comparison of different networks with similar error distribution, the ordinary standard deviation would still be a sufficient measurement of performance.

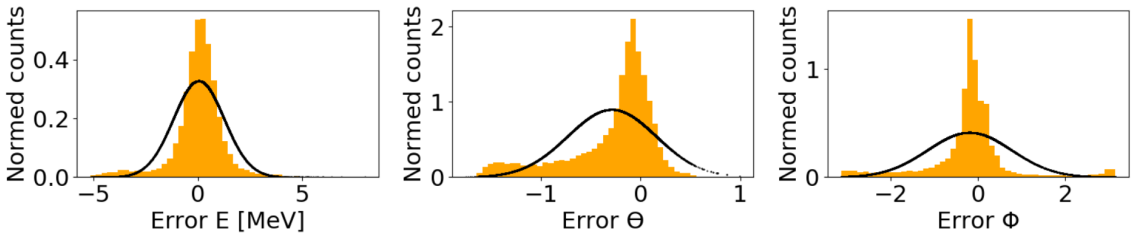


Figure B.1: The mean error and standard deviation as of the energy and angles in the intervals [4.95, 5.15] MeV, [1.57, 1.63] radians and [0.79, 0.82] radians respectively for the the network outputting the energy in the beam frame with the standard cost function in equation (3.2).

C
Additional tests

C.1 Network training time

As investigated in section C.5, the performance of the fully connected networks depended on the width and depth of the hidden layers. When training a vast amount of networks the time it takes to train each network is a major delimitation when setting the networks hyperparameters. It was concluded that the size of the fully connected layers greatly contributed to the training time of the network. The total training time is plotted in figure C.1 as of the width and depth of the hidden fully connected layers. The networks were trained using resources on Kebnekaise. Using the standard network size with ten layers and 128 nodes in each layer took over one and a half hour to train, in comparison with the longest training time of almost twenty hours.

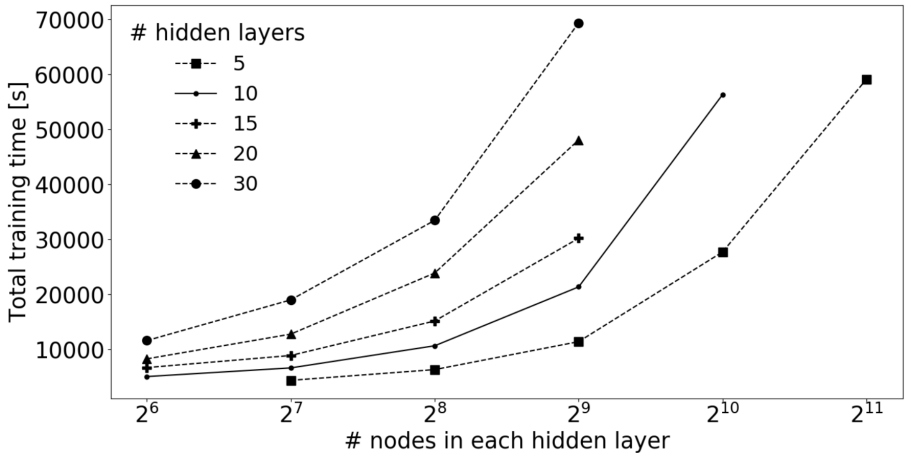


Figure C.1: The total training time dependence of the width and depth of the fully connected networks, varying from roughly one to twenty hours.

In figure C.2 the training time per each hidden node, total training time divided by the total amount of hidden nodes, is plotted. This can be considered as a measurement of the effectiveness in training time. It is seen that the training gets more effective from adding more nodes in each hidden layer until there are 512 nodes in each layer. After that, the training time per node increases. These effects might depend on the capacity of the computer used.

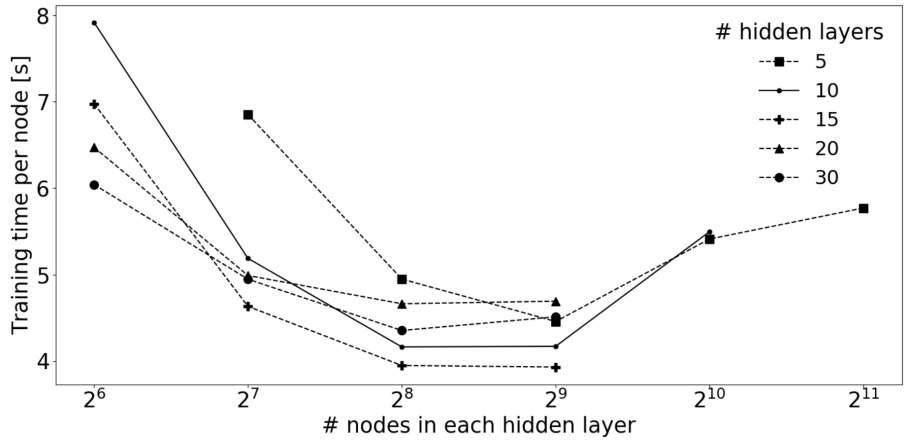


Figure C.2: The training time per hidden node.

C.2 Relative weighting of the classification node

In table C.1 the classification correctness is presented for two classification networks with different values of λ_μ. The λ_μ marked with an asterisk correspond to networks with the cost function in equation (3.6). Otherwise the cost function in equation (3.5) was used. No major tendencies of the classification are observed for varying λ_μ. Although, when using the cost function (3.6), the networks tend to be better at classifying zero γ-rays. This is expected since the network cost from the energy and angle terms is reduced for the γ-rays classified as non-existing. In table C.2, the mean standard deviations and errors of the energy and angles are presented for the same networks. The errors from the zero γ-rays have not been taken into account. In addition, the same measurement of performance is presented for the network without the classification node. This one has a slightly more accurate reconstruction than the networks utilising the classification node. For the classification networks, the standard deviations seem to increase for greater λ_μ. Since no major classification effect was observed for any choice of λ_μ, $\lambda_\mu = 1$ or 5 seems to be sufficient choices.

Table C.1: The classification performance of the networks with the cost functions in equation (3.5) and (3.6) for different values of λ_μ. The performance of the latter cost function is marked with an asterisk.

λ_μ	Correct zero γ	Correct non zero γ	Total correct γ	Correct multiplicity
1	96.3 %	91.1 %	92.8 %	79.1 %
5	95.8 %	92.1 %	93.3 %	80.4 %
10	96.1 %	91.9 %	93.3 %	80.3 %
50	95.1 %	92.7 %	93.5 %	81.0 %
100	95.8 %	92.0 %	93.2 %	80.2%
1*	97.8 %	88.4 %	91.5 %	75.8 %
5*	97.1 %	90.5 %	92.7 %	78.8 %
10*	96.0 %	91.9 %	93.3 %	80.4 %
50*	96.4 %	91.4 %	93.0 %	79.7 %
100*	95.4%	92.4%	93.4%	80.7%

Table C.2: The mean error (Error) and mean standard deviation (Std) in the energy and angle reconstructions for the networks trained with the cost functions in equation (3.5) and (3.6) for different values of λ_μ. The performance of the latter cost function is marked with an asterisk.

λ_μ	Std E [MeV]	Std θ	Std ϕ	Error E [MeV]	Error θ	Error ϕ
-	1.074	0.379	0.644	0.048	-0.191	0.013
1	1.128	0.391	0.680	0.027	-0.197	-0.003
5	1.092	0.391	0.662	0.074	-0.206	-0.028
10	1.132	0.401	0.665	0.023	-0.206	-0.015
50	1.189	0.400	0.769	-0.012	-0.179	0.023
100	1.275	0.407	0.890	-0.014	-0.258	-0.010
1*	1.086	0.378	0.673	-0.024	-0.250	-0.008
5*	1.114	0.405	0.657	-0.023	-0.231	-0.020
10*	1.096	0.380	0.698	0.056	-0.206	-0.019
50*	1.258	0.400	0.809	0.050	-0.250	-0.021
100*	1.294	0.408	0.893	0.102	-0.256	-0.001

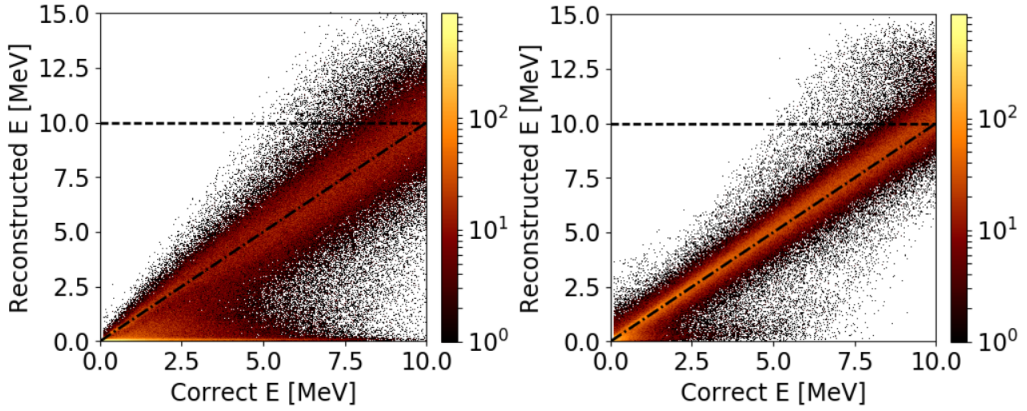
C.3 Relative cost function

The standard cost function in equation (3.2), is based on punishing the network by the mean square error. This implies that for a specific error, the network will be punished by the same amount, independent of the value of the correct observable. This is desirable for the two spatial angles since, for example an error of $\pi/2$ is equally bad if the correct angle is zero or π, respectively. However, an energy error of 0.4 MeV would be inacceptable for a correct energy of 0.1 MeV but tolerable if the correct energy was 10 MeV. This reconstruction trait can be enhanced by using the relative cost function;

$$C(\mathbf{y}, \hat{\mathbf{y}}) = \sum_{i=1}^{n_\gamma} \left[\lambda_E \left(\frac{\Delta E}{\hat{E}_i + E_{\text{offset}}} \right)^2 + \lambda_\theta \left(\Delta\theta_i \right)^2 + \lambda_\phi \left((\Delta\phi_i + \pi) \bmod 2\pi - \pi \right)^2 \right], \tag{C.1}$$

where $\hat{E}$ denotes the correct energies. E_{offset} is a parameter used to avoid dividing by zero for the non-existing γ-rays with zero energy.

For low $\hat{E}_i$ the cost of an energy error will be large and vice versa. In total, the network gains by reducing the absolute error for low energies. The use of the relative cost function can be seen in figure C.3 for $\lambda_E = 10$ and $E_{\text{offset}} = 0.5$ MeV. With the relative cost, the reconstruction is more accurate than with the standard cost function for energies lower than 1 MeV. Also notably is the accumulation of reconstructed energies above the abscissa. It seems as if the network tends to reconstruct too many γ-rays with low energies. This behaviour is altered by the two parameters λ_E and E_{offset}, further discussed in appendix C.4. However, the low-energy accumulation did not vanish completely for any choice of parameters without at the same time loosing the desired accuracy at low energies.



(a) **(b)**

Figure C.3: Reconstruction of γ-ray energies with the relative and absolute cost functions in (a) and (b), respectively. For the relative cost function, $\lambda_E = 10$ and $E_{\text{offset}} = 0.5$ MeV were used.

The classification output node was also implemented with the relative cost function in equation (C.1) to try solve the issue with the low-energy accumulation. However, no major improvement was observed.

C.4 Relative weighting of cost function terms with non-relativistic data

With non Doppler-shifted data, the performance of the network with the cost function in equation (3.2) is presented in figure C.4 for a variety of values of λ_E and λ_ϕ. The networks were trained and evaluated with γ-ray energies up to 10 MeV. It can be seen that with $\lambda_\phi = 5$ or 10 the standard deviation in the energy and polar angle reconstruction is inadequate. The same applies to $\lambda_\phi = 0.1$ for the azimuthal angle. Left is the two values of λ_ϕ by which $\lambda_\phi = 1$ with $\lambda_E = 1$ was assessed to be an sufficient choice of weighting to minimise the standard deviations. The data corresponding to $\lambda_\phi = 1$ is marked with a solid line in figure C.4.

For the relative cost function in equation (C.1) it was assumed that the relative weighting among the two angles could be as for the standard cost function, with $\lambda_\theta = 1$ and $\lambda_\phi = 1$. Instead of varying λ_ϕ the energy offset, E_{offset}, is introduced and the standard deviation depending on the weighting is presented in figure C.5. It is clear that $\lambda_\phi = 0.5$ yields the least standard deviation for both the energy and the angles. This is since a greater E_{offset} leads to a smaller low-energy accumulation but also less effect from the desirable features of the relative cost function. Therefore it is not necessary that that $E_{\text{offset}} = 0.5$ is a good choice.

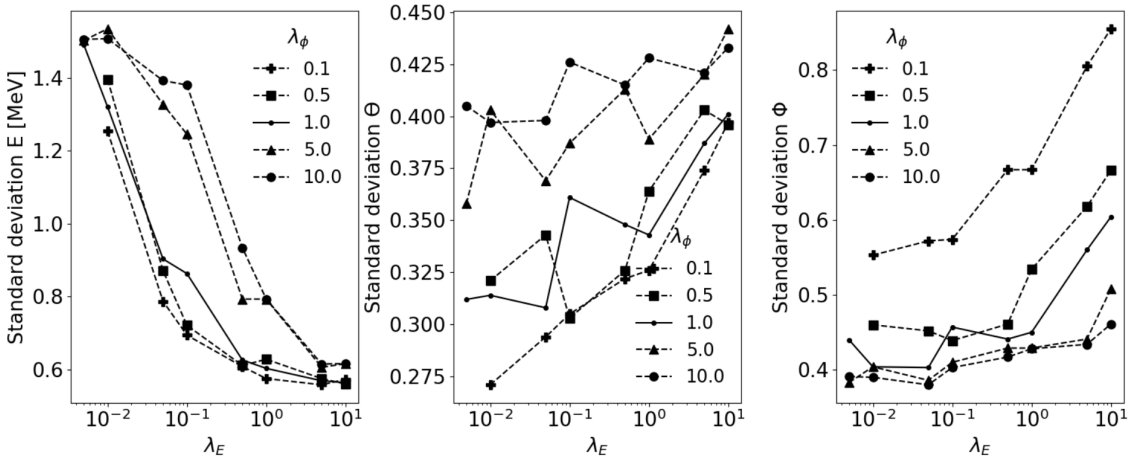


Figure C.4: The standard deviation of the energy and angles for varying values of λ_E and λ_ϕ with $\lambda_\theta = 1$. Here the standard cost function in equation 3.2 was used with non Doppler-shifted data. The favored choice of λ_ϕ is marked with a solid line.

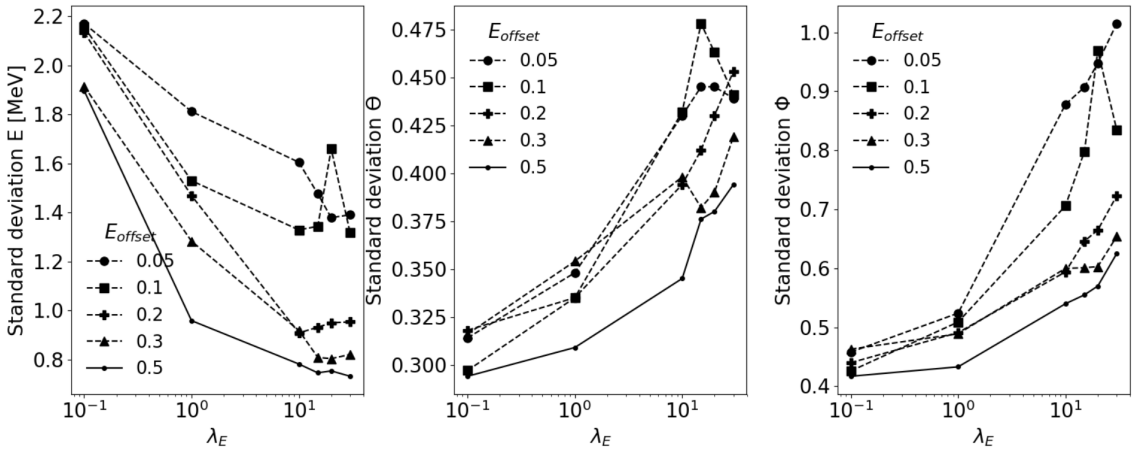


Figure C.5: The standard deviation of the energy and angles for varying values of λ_E and E_{offset} with $\lambda_\theta = 1$ and $\lambda_\phi = 1$. Here the relative cost function was used with non Doppler-shifted data. The favored choice of E_{offset} is marked with a solid line.

C.5 Structures of the fully connected layers

Two major hyperparameters of the fully connected networks are the width and depth of the fully connected layers. How these influence the performance of the reconstruction was tested with the BF network of depths varying from 5 to 30 fully connected layers. For each of the trained networks, the width was constant through the layers. The standard deviations of these reconstructions are plotted in figure C.6.

Observe that the network with the standard size of 10 layers with 128 nodes in each layer was one of the most accurate networks. All of the networks were trained with equally many iterations. The larger a network is, the more training iterations it would need before it becomes fully trained. This might probably have effected the performance of the presented networks. Since none of the shallower networks show any tendencies to out perform the others with increasing width, the conclusion was that no considerable improvement was to be expected for bigger FCN.

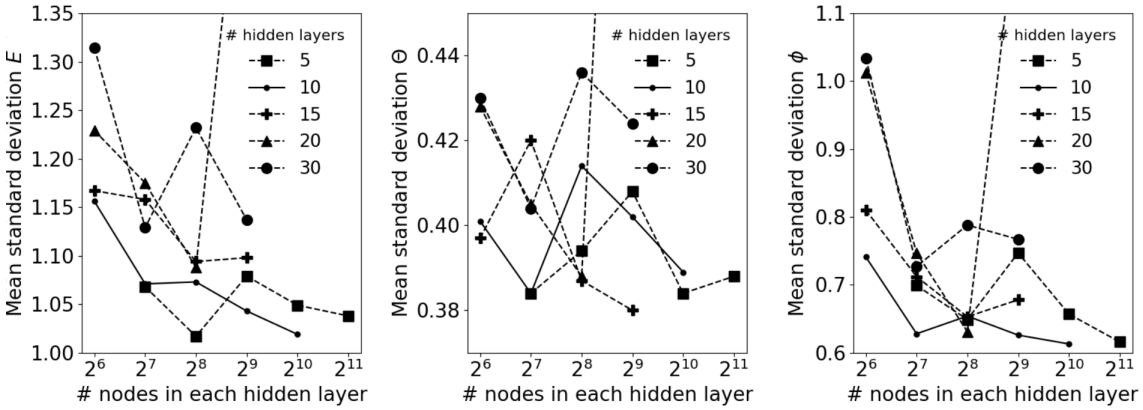


Figure C.6: The performance of the networks, measured as the mean standard deviations for different widths and depths of the fully connected networks. The network with 10 hidden layers, used as the standard network with 128 nodes in each layer, is marked with a solid line. The network with 20 layers and 512 nodes in each layer seems to not have been trained properly.