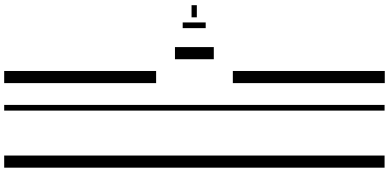


# A Deep Learning-Based 3D Position Reconstruction for XENONnT

Bachelor thesis

Philip Siddhartha Thielges

Westfälische Wilhelms-Universität Münster
Institut für Kernphysik
AG Prof. Dr. C. Weinheimer

First Referee: Prof. Dr. C. Weinheimer
Second Referee: Prof. Dr. C. Klein-Bösing

Münster, October 2022

# Declaration of Academic Integrity

I hereby confirm that this thesis on **A Deep Learning-Based 3D Position Reconstruction for XENONnT** is solely my own work and that I have used no sources or aids other than the ones stated. All passages in my thesis for which other sources, including electronic media, have been used, be it direct quotes or content references, have been acknowledged as such and the sources cited.

Münster, October 5, 2022 _____

I agree to have my thesis checked in order to rule out potential similarities with other works and to have my thesis stored in a database for this purpose.

Münster, October 5, 2022 _____

# Contents

# 1 Introduction

It turns out that only about 5 % of the universe is baryonic matter. The rest is composed of Dark Matter and Dark Energy, which are invisible but dominate the structure and evolution of the universe. Determining the nature of Dark Matter, which accounts for much of the mass of galaxies in the universe, has been and remains one of the greatest puzzles of recent decades. But if there is that much mass represented as dark matter, why has no detector found it yet? The problem is that Dark Matter does hardly interact with any matter known to us. There are still various theories about which Dark Matter candidate is the most appropriate one. Currently, most attention is given to the WIMP-model (Weakly Interacting Massive Particle), which is also the target of the XENONnT-detector situated in the underground laboratory Laboratori Nazionali del Gran Sasso (LNGS) in Italy. XENONnT consists of a dual-phase time projection chamber (TPC) filled with liquid and gaseous xenon. When a particle deposits energy in form of a recoil in the detector, a light and a charge signal are generated, called S1 and S2, respectively. The Dark Matter search analysis relies greatly on the 3D position reconstruction of signals. The pattern of the photomultiplier tubes detecting the ionization signal S2 is typically used to reconstruct the horizontal position of the event. The time difference between the S1 and S2 signals provides information about the depth of the interaction in the detector.

Of course Dark Matter is not the only field in research of high relevance: deep learning, a subset of machine learning, has gained a lot of attention. While the human brain is not naturally capable of working in more than three dimensions, there is no such dimensional limit for computers. A machine can operate in high dimensional spaces, looking for correlations between many variables at once. Thus, it may be that better results are obtained when the computer finds its own way to establish rules instead of prescribing these rules itself. This thesis proposes a newly developed deep learning approach to reconstruct the three-dimensional position based exclusively either on the S1 or S2 signal, depending on the dataset chosen. It employs different neural networks that take the measurements of the photomultiplier tubes at the top and bottom array as input. The training of the networks is based on Monte Carlo (MC) simulations. This method can be used, for example, to analyze events where either the S1 or S2 signal is not available. This is especially the case at low energies where only the

S2 signal is registered. Thus, it might be possible to determine the depth of an event in the detector without the time difference of the two signals.

This thesis is structured as follows: At the start of chapter 2, the current status of the Dark Matter search is surveyed, followed by an introduction to the XENONnT experiment with its detector, the TPC. Chapter 3 first discusses various neural networks and their basic optimization algorithms. Subsequently, the simulated data used and their preprocessing are described, the developed algorithms are introduced in detail. After comparing the selected model architectures with classical reconstruction methods, an outlook is given in the form of a performance analysis for different event depths and varying amplitudes.

# 2 XENONnT: Direct Dark Matter Search

The following chapter first presents basic evidence for the existence of Dark Matter. After that, the various Dark Matter detection channels are reviewed. The chapter also focuses on the experiment itself. A general overview about the used detection method is given followed by a description of the XENONnT experiment.

## 2.1 Indications of the Existence of Dark Matter

The universe is thought to consist of three different types of matter: baryonic matter, Dark Matter, and Dark Energy. Dark Energy is considered to be responsible for the accelerated expansion of the universe. Most of the matter in the universe, according to standard gravitational theory, is in a form that is unknown to us and does not emit enough light to have been detected by current instrumentation. Researchers from around the world are working to finally uncover the properties and possible physical candidates of this Dark Matter. Evidence for Dark Matter exists from the large to the small scale, here two instances will be taken up in more detail.

The stellar mass-luminosity relation can be applied to determine the mass $M$ of a main sequence star as a function of its mere luminosity $L$ via the relation $L \propto M^{\alpha}$ with $\alpha \approx 3$ for large masses and $\alpha \approx 4$ for small masses [10]. However, if one measures the rotational velocity of stars as a function of their distance from the galactic center, the mass-luminosity relation gives a much smaller mass than that obtained by applying Kepler's laws. As an instance, the galaxy NGC 3198, to be seen in figure 2.1, can be mentioned: For large radii $r$, an almost constant value was measured for the rotation velocity $v$, the predicted decay according to Kepler's law, following $v \propto \frac{1}{\sqrt{r}}$, could not be observed [33]. Consequently, there must be mass present in the galaxy, which does not contribute to the radiation emission. If one looks at typical rotation curves of galaxies, this Dark Matter cannot be explained only by a black hole in the center of the galaxies [29]. Rather, it can be concluded that it must be spatially distributed in a galaxy, nearly uniformly. If a Dark Matter halo is introduced, these observations can be physically reproduced. This halo neither absorbs nor emits radiation, it dominates the total mass in the universe.

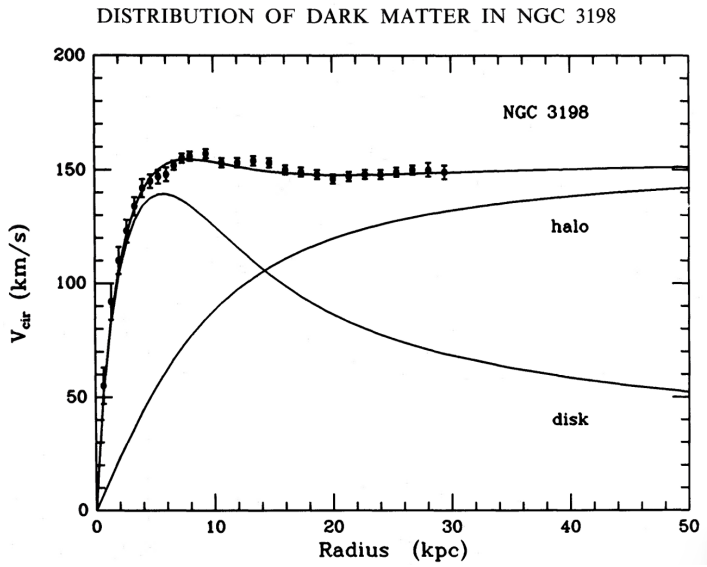


Figure 2.1: Rotational velocity as a function of the radius from the galaxy NGC 3198, the measured velocity (black points) differs from that expected given the distribution of the visible mass (disk). The discrepancy between these two curves can be accounted for by the existence of a Dark Matter halo. Image taken from [33].

The existence of Dark Matter is also supported by data coming from gravitational lensing [8]. This effect, first described by Einstein, results from the change of the four-dimensional continuum with the Lorentzian manifold induced by mass concentrations. If a light beam propagates in the gravitational field of a mass, it is bent under the influence of gravity, following the early predictions of general relativity. The light path of photons deflected by a gravitational lens will be deflected more towards the mass the closer they pass the deviating mass. Gravitational lensing effects can be used as a unique tool to map the matter distribution in the universe. Measurements show that the baryonic matter visible in the light path is smaller than the mass that can be reconstructed using gravitational lensing [23]. The powerful evidences of the existence of Dark Matter have led to the idea of a non-interacting, unknown form of matter, which is almost universally accepted nowadays.

## 2.2 Dark Matter Detection Channels

The Standard Model of particle physics is nowadays a well-established theory. The Weakly Interacting Massive Particle (WIMP), not belonging to this Standard Model, is one of the most attractive Dark Matter candidates. It is assumed to be heavy, electrically uncharged and stable. Although the properties of a WIMP vary greatly depending on the model chosen, its mass should be placed in the GeV/$c^2$ to TeV/$c^2$ range [27]. Moreover, its interactions are constrained to be weak scale. Many WIMP candidates have in common that they are

expected to have been generated as a thermal relic in the early universe when it was hot and dense. This would show an analogy to the particles of the Standard Model according to the big bang theory. As the universe continued to expand and cool more and more, its temperature reached a point below the mass of the WIMPs and a freeze-out of these particles occurred, they could no longer find each other to annihilate. Other Dark Matter candidates, e. g. axions, are not discussed in this thesis.

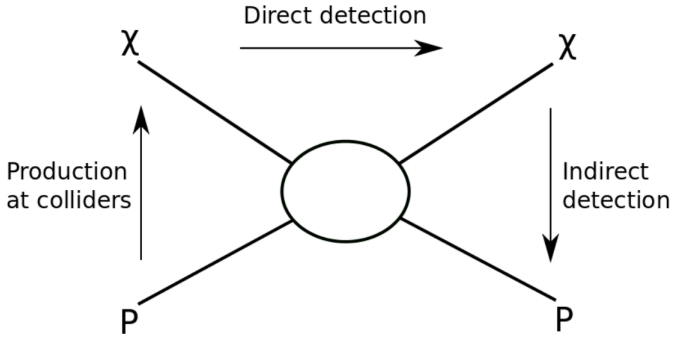


Figure 2.2: Schematic representation of the possible Dark Matter detection channels. Image taken from [32].

Based on these assumptions about the WIMP, three promising detection channels, shown in figure 2.2, come to mind for detecting the potential particles:

1. Production at colliders: WIMPs could be produced by collisions of Standard Model particles in a particle collider. The particles could not be observed directly, but the missing mass method based on energy and momentum conservation can be applied.

2. Indirect detection: It is also possible to detect them indirectly by searching for Dark Matter annihilations. Two WIMPs can annihilate into ordinary matter such as quarks, leptons etc., as they did in the early universe.

3. Direct detection: The last channel, that is also used in XENONnT, is the direct detection. Experiments of this type attempt to detect the recoil energy deposited in a low-background detector when a WIMP scatters elastically from a nucleus within the detector.

Typically, the recoil energy can be deposited in three different ways. The most common approaches are using scintillation, ionization and heat. Numerous experiments around the world act as direct detection experiments using two of the three possible detection paths to differentiate WIMP signals from background signals. The XENON Dark Matter Project takes advantage of the scintillation and ionization channels to search for Dark Matter, setting world leading limits on the scattering cross-section of WIMPs with ordinary matter. Therefore, XENON uses the liquid xenon dual-phase TPC as described in the next section.

## 2.3 Dual-Phase Time Projection Chamber

The heart of the XENONnT experiment is a dual-phase time projection chamber (TPC), which serves as the detector in this experiment. The TPC, a large cylindrical vessel filled with 8.6 t of ultra-pure liquid xenon (LXe), is shown in figure 2.3. In addition to the liquid xenon, there is another layer of gaseous xenon (GXe) on the ceiling of the chamber. Arrays of photomultiplier tubes (PMTs) are attached to the top and bottom of the TPC. These arrays are arranged in a hexagonal structure to allow as little light as possible to escape. When a particle enters the TPC, two different interactions can take place: Either the particle scatters at an atomic nucleus in the detector, such as WIMPs or neutrons, or it is scattered at an electron of the atomic shell, such as gamma or beta radiation. These interactions are referred to as nuclear recoil (NR) and electronic recoil (ER), respectively. [12]

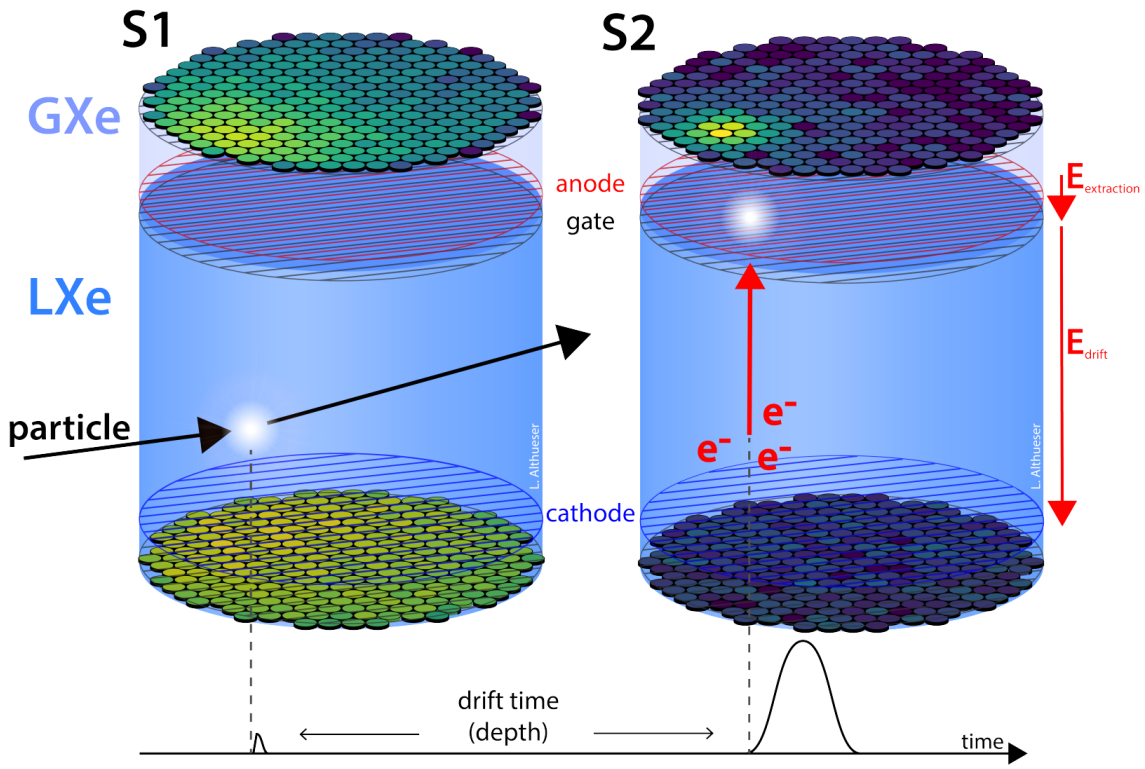


Figure 2.3: Functioning of the TPC and representation of the two signals S1 (left) and S2 (right). When a particle enters the detector, a prompt scintillation light S1 is generated, which can be registered by the photomultiplier tubes on the top and bottom of the TPC. The electrons that do not recombine are first accelerated towards the gate by an electric field $E_{\text{drift}}$ and then extracted into the gas phase by another field $E_{\text{extraction}}$, where a second delayed scintillation signal, S2, is produced by electroluminescence. Image taken from [1].

The energy deposition in the detector gives rise to scintillation light emitted from an excited dimer ($Xe_2^*$) that decays to the ground state. This excited dimer is created by two different

processes, which can be seen in equation (2.1) and equation (2.2) [2]. The first process is initiated by an excited xenon atom (Xe*):

$$\begin{aligned} \mathrm{Xe}^* + \mathrm{Xe} + \mathrm{Xe} &\longrightarrow \mathrm{Xe}_2^* + \mathrm{Xe} \\ \mathrm{Xe}_2^* &\longrightarrow 2\,\mathrm{Xe} + \mathrm{h}\nu \end{aligned} \tag{2.1}$$

The second one consists of some processes, starting with an ionized xenon atom ($Xe^+$), from which an ionized xenon dimer is formed:

$$\begin{aligned} \mathrm{Xe}^+ + \mathrm{Xe} &\longrightarrow \mathrm{Xe}_2^+ \\ \mathrm{Xe}_2^+ + \mathrm{e}^- &\longrightarrow \mathrm{Xe}^{**} + \mathrm{Xe} \\ \mathrm{Xe}^{**} &\longrightarrow \mathrm{Xe}^* + \mathrm{heat} \\ \mathrm{Xe}^* + \mathrm{Xe} + \mathrm{Xe} &\longrightarrow \mathrm{Xe}_2^* + \mathrm{Xe} \\ \mathrm{Xe}_2^* &\longrightarrow 2\,\mathrm{Xe} + \mathrm{h}\nu \end{aligned} \tag{2.2}$$

This ionized xenon dimer can recombine with an electron. Finally, the state relaxes to a single excited state, during this process heat is also produced. Again, the emission of scintillation photons is possible, as in equation (2.1). Since the scintillation light is emitted by an excited dimer, other xenon atoms cannot absorb this light. Thus, xenon is transparent to its own scintillation light [2]. The emergence of the scintillation photons, which have a wavelength of 178 nm, allows them to propagate through the detector. Highly reflective polytetrafluoroethylene (PTFE) is attached to the sides of the TPC so that as many photons as possible are reflected and then detected. Due to the difference in the refractive indices of LXe [30] and GXe [5], much of the light is reflected at the liquid-gas interface and thus registered at the lower PMT array. This prompt scintillation light is also referred to as the S1 signal.

In general, LXe is able to produce both scintillation light and electrons by the energy deposition of particles in the detector. If an electric field $E_{\mathrm{drift}}$ is applied between an anode at the liquid-gas interface and a cathode at the bottom of the TPC, depending on the field strength, there are electrons that do not recombine according to equation (2.2). The greater the field strength, the fewer recombinations occur. Light and charge signal are therefore generally anticorrelated. These charges can be driven upwards by the electric field. Above the LXe, an even stronger electric field $E_{\mathrm{extraction}}$ is applied in the gas phase between a gate mesh and an anode grid. Once the electrons reach the liquid-gas interface, they are extracted into

the gaseous phase by the stronger electric field. Electroluminescence produces a second light signal which is detected as the S2 signal. [22]

An essential feature in the Dark Matter search is the principle of background discrimination. Especially for large-scale experiments, the sensitivity of the experiment is limited by the background level. As WIMPs are expected to have an extremely small cross-section, they scatter only once within the detector. Particles that scatter multiple times in the TPC can thus be discarded as possible candidates. Since most backgrounds are electronic recoils, differentiation between NR and ER events is crucial. The ratio of S2 to S1 signal is different for a nuclear and an electronic recoil, allowing these two to be distinguished:

$$\left(\frac{\mathrm{S2}}{\mathrm{S1}}\right)_{\mathrm{NR}} < \left(\frac{\mathrm{S2}}{\mathrm{S1}}\right)_{\mathrm{ER}} \tag{2.3}$$

Xenon is used as the detector material due to its high mass number of A = 131. The spin independent cross-section for interactions of WIMPs with nuclei is proportional to $\mathrm{A}^2$, so a large mass number will increase the number of WIMP-nucleon scatters. There are further several advantages: Being a noble gas, xenon is low chemical reactive and does not react with impurities inside the TPC. It has a high atomic number of Z = 54, which is why the stopping power for $\beta$- and $\gamma$-radiation is very high. This self-shielding of xenon allows a background reduction by fiducialization selecting only events in the innermost part of the detector. Since the radiation from most background sources is stopped in the outer part of the detector, the fiducial volume has a significantly reduced event rate. The presented methods are employed by the experiment XENONnT, which will be described in the following section.

## 2.4 XENONnT

The XENONnT detector that enables the search for the potential WIMPs is installed at the INFN Laboratori Nazionali del Gran Sasso (LNGS) in Italy, the largest underground laboratory in the world [20]. The laboratory is surrounded by 1400 m thick rock, which serves as a natural shield against cosmic radiation. The radiation arriving in the underground halls can thus be reduced by six orders of magnitude [20]. The TPC is placed inside a water tank, which effectively protects against environmental gamma and neutron radiation as an additional shield. The XENON collaboration launched the experiment in 2005, initially running under the name XENON10. In the following years, both the sensitivity of the experiment and the detector size were increased, as can be seen in table 2.1. After continuing the experiment as XENON100 and XENON1T, data are currently being acquired with the

stage XENONnT, utilizing 5.9 t of instrumented liquid xenon as the target mass (8.6 t in total).

Table 2.1: Different stages of the XENON Dark Matter Project: The project XENONnT is currently running. The operation time and the total xenon mass can be seen respectively. Data was taken from [34].

| XENON10 | XENON100 | XENON1T | **XENONnT** |
|---|---|---|---|
| 2006-2007 | 2008-2016 | 2015-2018 | **Since 2020** |
| 25 kg | 160 kg | 3200 kg | **8600 kg** |

An overview of the XENONnT experiment is shown in figure 2.4. The building on the right-hand side houses the systems necessary to run the TPC. For instance, xenon can be stored in the *ReStoX* system, a double-walled stainless steel sphere. The system can store up to 7.6 t (*ReStoX-I*) plus 10 t (*ReStoX-II*) of xenon, either in the liquid, gaseous or solid phase [34]. The TPC on the left-hand side is located inside a water tank that serves as an additional shield against gamma rays and neutrons from natural radioactivity.

Figure 2.4: Setup of XENONnT located in Italy in the underground laboratory in Gran Sasso. The LXE dual-phase TPC is mounted in a water tank on the left side. On the right side stands a service building containing all subsystems necessary for operation, including data processing, xenon purification and xenon storage. Credit: Henning Schulze Eißing.

# 3 Position Reconstruction in XENONnT

This chapter is all about the three-dimensional position reconstruction in the detector. Due to the working principle of the TPC, a full 3D position reconstruction of a recoil event is possible. If the conventional reconstruction is considered, the $x$- and $y$-coordinates can be determined via the intensity distribution of the S2 signal on the top PMT array. The depth of an event, the $z$-coordinate, can be reconstructed using the time difference of the two generated signals, S1 and S2. In this thesis, a new approach built with deep learning techniques is proposed to reconstruct the three-dimensional position of an event in the TPC. In contrast to the classical reconstruction, the goal is to use the newly developed models for events where only information about one of the signals is available (either S1 or S2). This is particularly interesting for small recoil energies where the S1 signal is not registered. With these models, it could be possible to determine the depth of an event without the time difference between S1 and S2. In this chapter, first an overview of the necessary theoretical deep learning techniques is given. Then, the simulation-based data and its training are discussed in more detail, followed by an evaluation of the training by comparing the developed models with classical methods for position reconstruction. At the end of the chapter, an analysis of the model performances is given by examining them for different event depths and amplitudes.

## 3.1 New 3D Position Reconstruction: Development of a Deep Learning Model

In this thesis, different neural networks are applied to the position reconstruction problem and analyzed in detail afterwards. In order to get an overview of the methods used, the necessary background information is given here. The content of this section covers deep learning in general. After the introduction of a multi-layer perceptron, a class of feedforward artificial neural network, the training of a neural network is described. This training can be applied to all architectures outlined in this section. Finally, the models used in this work are presented: a convolutional neural network (CNN), a long short-term memory (LSTM) and a CNN-LSTM.

### 3.1.1 Fundamentals of Deep Learning

The field of artificial intelligence (AI) was born in the 1950s, when computers were used to perform intellectual tasks normally handled by humans. This field of AI gave rise to a subfield, machine learning (ML), which focuses on learning-based approaches. In contrast to classical programming where the output is obtained by human input of explicit rules and data, also known as *symbolic* AI [24], ML takes the data and corresponding desired outputs and learns the underlying rules which can then be applied to new data. In the latest years, a trend evolved where almost every ML approach tends to use deep learning. Deep learning is a subset of ML motivated by the functionality of the human brain. It is currently the state-of-the-art machine learning approach for natural language processing, computer vision and many artificial intelligence tasks in general. The recent development of high-performance graphics processing units (GPUs) enabled the training of more complex models in even shorter times. [6]

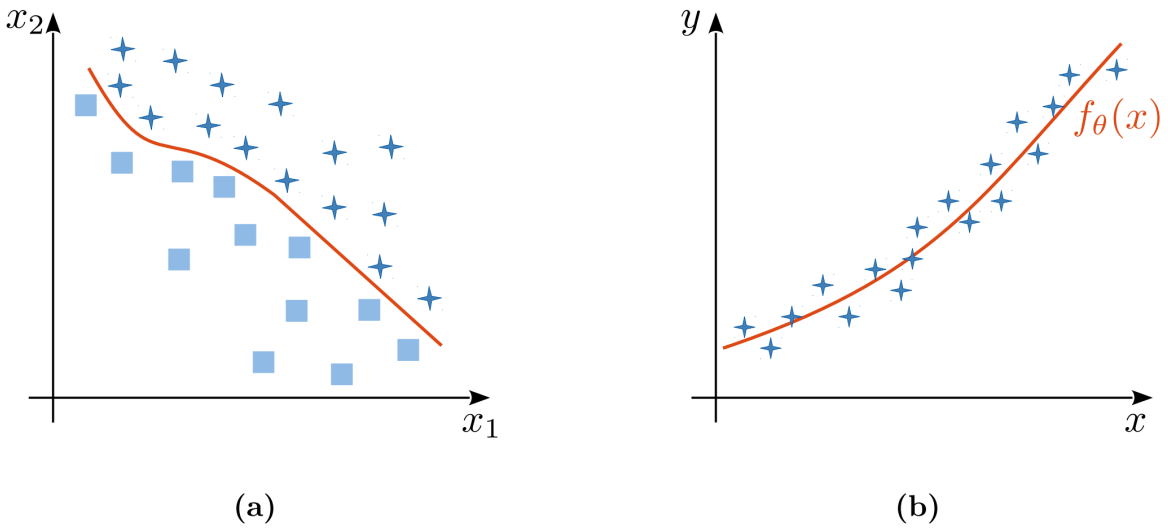


Figure 3.1: Different supervised machine learning classes. (a) Classification: The model is trained based on the parameters ($x_1$, $x_2$) to separate the given samples into two classes. (b) Regression: The model is trained depending on the continuous inputs $x$ and labels $y$. Image taken from [11].

Usually algorithms are classified into four categories based on their training procedure: supervised, unsupervised, self-supervised and reinforcement learning. In supervised machine learning, models are trained using labeled data. The data consists of sample pairs, i. e. inputs $x_i$ and corresponding labels $y_i$. In general, supervised models are trained to approximate a function that describes the underlying relation between inputs $x$ and labels $y$. This principle can be mainly divided into two classes shown in figure 3.1, classification and regression. Classification is a process of finding a function which helps in dividing the dataset into classes

based on different parameters. A model is trained on the training dataset and categorizes the data based on that training. A well-known example from physics is the classification of events as signal or background. On the other hand, regression is a process of finding the correlations between dependent and independent variables. It helps in predicting continuous variables such as prediction of house prices. In this work, supervised learning with a regression task is used by reconstructing the three-dimensional position of an event in the TPC.

The purpose of a ML model is to generalize features from training data to unknown data. Underfitting can occur, where the capacity of the model is too small, it cannot describe the training data well. In deep learning models, however, overfitting is a much larger risk. Overfitting denotes the effect of a model learning statistical fluctuations that occur in the training data. When such models are applied on unseen data, they exhibit large generalization errors. To avoid overfitting, the data available for development can be split into training and validation data. By observing the training and validation loss (see section 3.1.2), overfitting can be detected. Most of the algorithms currently in use are based on neural networks, which are introduced in the following section.

### 3.1.2 Multi-Layer Perceptron

Artificial neural networks (NNs) are a class of machine learning algorithms whose structure is based on nervous systems such as the human brain. The basic architecture of a multi-layer perceptron can be seen in figure 3.2. The circles represent the so-called neurons which are grouped in densely connected layers. The left column is the input layer, which is processed by one or more hidden layers in the middle before it reaches the output layer. By connecting one neuron to each neuron in the next layer, one can assign a specific weight $w$ to each individual connection, where $w_{ij}^{(l)}$ corresponds to the weight from the $i$-th neuron in the $l$-th layer to the $j$-th neuron in the $(l+1)$-th layer. These weight values essentially dictate how important the information being passed on is when looking to obtain the correct, or at least very accurate, predictions. The output of this $j$-th neuron depends on the input of all neurons in the previous layer:

$$x_j^{(l+1)} = f\left(\sum_{i=1}^{N} w_{ij}^{(l)} x_i^{(l)} + b_j^{(l)}\right) , \tag{3.1}$$

where $N$ is the number of neurons in the $l$-th layer and $b$ is the bias vector [18]. The function $f$ is a nonlinear activation function. The nonlinearity ensures the stacking of layers does not yield trivial results. Thus, the derivative of the activation function controls the scale of

the gradient. A common activation function, which is not affected by the vanishing gradient problem [6], is the rectified linear unit (ReLU). It is defined as follows:

$$f(x) = \begin{cases} 0 & \text{for } x \leq 0 \\ x & \text{for } x > 0 \end{cases} \tag{3.2}$$

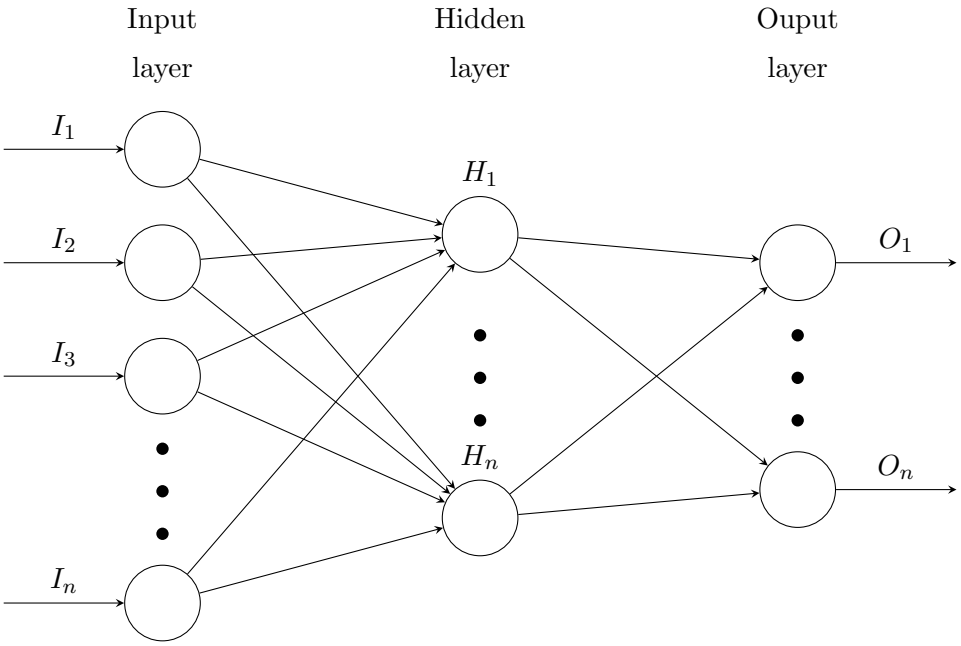


Figure 3.2: Basic architecture of a multi-layer perceptron with $n$ input features. The number of hidden layers can vary according to the selected model architecture. The circles represent the neurons, the arrows connect one neuron to each neuron in the next layer. Depending on the dataset, one has one neuron in the input layer for each input feature. These values are passed through to the middle section of the network, the hidden layers. By linking one neuron to each neuron of the next layer, information is passed from left to right through the network in a process called forward propagation.

### Training of a Neural Network

A sketch of the general training procedure of NNs is displayed in figure 3.3. The model takes the input data $X$ and outputs predictions $Y'$. These predictions are compared with the targets $Y$, using a so-called loss function $\mathcal{L}$. A loss function is a metric which quantifies the difference between $Y$ and $Y'$. Usually, the *mean squared error* (MSE) is used as the loss function $\mathcal{L}$ for regression, the square of the difference between predictions $Y'$ and targets $Y$:

$$\mathcal{L} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - Y_i')^2 \tag{3.3}$$

In this thesis, $n$ refers to the number of input samples. This metric can be used to increase the performance of the model by updating the individual weights.

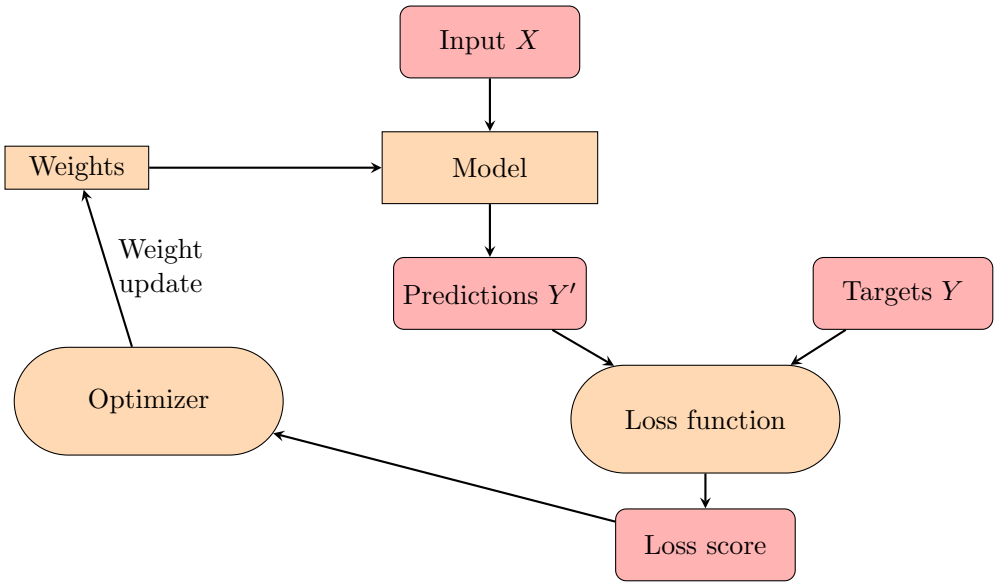


Figure 3.3: Training procedure: The network, composed of the layers, maps the input data to predictions. The loss function then compares these predictions to the targets, producing a loss value: a measure of how well the network's predictions match what was expected. The optimizer uses this loss value to update the network's weights.

However, one also needs an optimizer that determines how the network is updated based on the loss function. It implements a specific variant of *stochastic gradient descent* (SGD). *Gradient descent* in general is an algorithm which is iteratively used to find a local minimum of the loss function. In a figurative sense, the loss is minimized by calculating the gradient with respect to all trainable parameters. Then, the parameters are adjusted in the opposite direction of the gradient within a defined step size, the learning rate. When dealing with networks of many layers, the chain rule has to be used to calculate the loss gradient. This process, called *backpropagation*, starts at the output layer and works itself backwards through the whole network to adjust each trainable parameter. Whereas in *gradient descent*, usually the whole dataset or a single sample was used for a single update, in SGD, a batch of $n$ samples is utilized for a single update. SGD has two important advantages compared to *gradient descent* [26]. On the one hand, it speeds up learning as it enormously reduces the computational costs for a single 'whole-data' update. On the other hand, the use of small batches improves the generalization performance as for each update the gradient averaged over various samples is used. A frequently used optimizer is *Adam* (adaptive moment estimation) [17], which is based on SGD and features adaptive learning rates.

### 3.1.3 Convolutional Neural Networks

Convolutional neural networks (CNNs) [19] are a type of deep learning neural network and are employed particularly in computer vision and image recognition. A key difference between a densely connected layer and a convolution layer is that dense layers learn global patterns, whereas convolution layers learn local patterns. Unlike CNNs, multi-layer perceptrons ignore an essential property of images: Nearby pixels are more strongly correlated with each other than distant pixels. Looking at the structure of CNNs, there are mainly 2 types of layers to be considered:

**Convolutional layer** The objective of the convolution operation is to extract high-level features from the input image. The input image is scanned by small filters referred to as kernels. Each kernel produces a new image called a feature map. The application of the filters, i. e. the convolution of an image $x$ with the kernel, can be divided into two parts. According to where the filter is applied, the weights $W$ of the filter are multiplied by the corresponding pixel $x_k$ and the neighboring pixels $x_j$, depending on the kernel size. As a second step, the result of this multiplication is combined over the neighborhood $N_i$ into a single value by summation. If a nonlinearity $f$ and a bias $b$ are considered, the output of the convolution operation with one filter is calculated for each activation $x'_k$ in the resulting feature map as

$$x'_k = f\left(\sum_{x_j \in N_i} W_j x_j + b\right). \tag{3.4}$$

Figure 3.4 shows an example of the convolution operation with a kernel of size $3 \times 3$. After the entire image is scanned by the filter, a bias and an activation function is applied. Common choices for kernel sizes are $3 \times 3$ or $5 \times 5$, for a 2D convolutional layer respectively.

**Pooling layer** Pooling layers are used to downsample the feature maps in order to limit the required computing power. In addition, this makes the model more robust to variations in the positions of the features of the input image by scanning the image with a filter. Two common pooling methods are max pooling and average pooling. Max pooling returns the maximum value from the portion of the image covered by the kernel. On the other hand, average pooling returns the average of all the values from the portion of the image covered by the kernel. The standard kernel size used for a 2D pooling layer is $2 \times 2$.

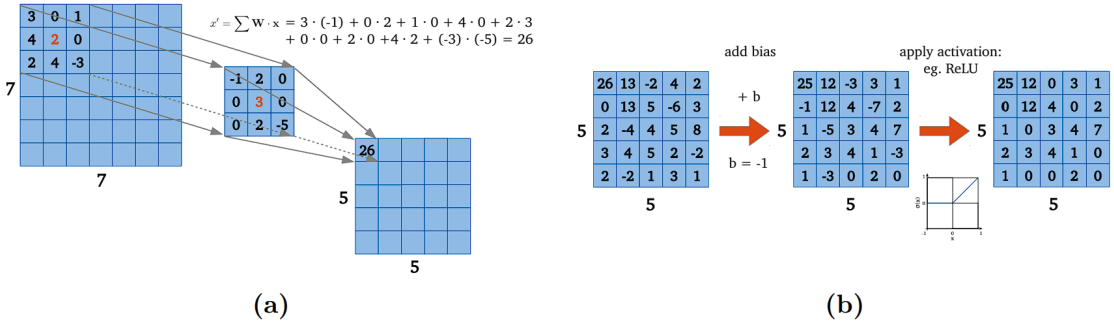


Figure 3.4: The diagram illustrates the convolutional unit of a CNN for a single channel image with a single filter. (a) The filter with a kernel size of $3 \times 3$ scans the input image. The dimension of the image is reduced from $7 \times 7$ to $5 \times 5$. (b) Addition of a bias and an activation function (ReLU). Image taken from [11].

### 3.1.4 Recurrent Neural Networks

Recurrent neural networks (RNNs) are particularly suitable for processing sequential data. A key characteristic of RNNs that distinguishes them from feedforward networks is their memory capacity. In CNNs, each input is processed independently, whereas RNNs maintain states that provide information about what has been processed so far. They take as their input not just the current input, but also what they have perceived previously in time. An RNN is basically a neural network with an internal loop, the current output is included in the new sequence. By default, only the last output for each sequence is returned for a recurrent layer. It is also possible to output one output for each input time step, so that the number of dimensions remains unchanged compared to the previous layer [21].

#### LSTM – Long Short-Term Memory

However, simple RNNs are not suitable for learning long-term dependencies [3]. This is due to the vanishing gradient problem, which occurs especially in deep networks. Applying the backpropagation algorithm, the derivatives from each layer are multiplied together by the chain rule to calculate the derivatives of the initial layer. If the gradients of the layers are less than one, this causes the gradient of the initial layers to decrease exponentially. This ends up leaving the weights of the initial layers essentially unchanged, with no effective training taking place. The theoretical reasons for this effect are explained in more detail in [4].

The LSTM layer [15] solves this problem, it is capable of learning these kind of dependencies. A sketch of the information processing in an LSTM cell is shown below in figure 3.5. It contains internal mechanisms, also called gates, which can regulate the information flow. The essential concept of LSTMs is the cell state and the different gates. The cell state can carry relevant information from earlier to later time steps. The gates are layers that decide to

what extent the information of the respective cell is relevant for the whole training. They use *sigmoid* activation $\sigma$ as their activation function, which is defined so that values are assigned a value between 0 and 1 depending on their relevance to the training:

$$\sigma(x) = \frac{1}{1+e^{-x}} = \frac{e^x}{1+e^x} = \frac{1}{2}\left(1 + \tanh\left(\frac{x}{2}\right)\right) \tag{3.5}$$

While the information is entirely forgotten in the case of the output of a 0, it is completely preserved in the case of a 1. When considering the update of a LSTM cell, four different steps should be distinguished:

- Forget gate

  The first processing step is controlled by the forget gate. This gate takes into account the current input $x^{\langle t \rangle}$ and the information from the previous hidden state $h^{\langle t-1 \rangle}$. With *sigmoid* $\sigma$ as the activation function, the fraction of the memory to be forgotten can be determined.

- Input gate

  In the second step, the input gate determines how much should be learned in the current step. A *tanh* layer creates a vector that contains information about the new input data $x^{\langle t \rangle}$ based on the context of the previous hidden state $h^{\langle t-1 \rangle}$. A *tanh* function is used to allow negative values to be produced, possibly reducing the influence of components in the cell state. Analogous to the input gate, there is a *sigmoid* activated network that is used to decide which values of the vector to keep. The outputs of both layers are multiplied pointwise to be able to regulate the flow of new information.

- Cell state

  Using the previous two steps, the old cell state $c^{\langle t-1 \rangle}$ can be updated into the new cell state $c^{\langle t \rangle}$. The memory update is performed by multiplying the old state $c^{\langle t-1 \rangle}$ with the output of the forget gate to forget the previously determined information. Finally, the output of the input gate is added to obtain the new cell state $c^{\langle t \rangle}$.

- Output gate

  The output, the next hidden state $h^{\langle t \rangle}$, is obtained by applying a *tanh* layer to the updated cell state $c^{\langle t \rangle}$. Again, a *sigmoid* layer is used to decide which parts of this state should be output. The cell state $c^{\langle t \rangle}$ and the new hidden state $h^{\langle t \rangle}$ are passed to the next cell.

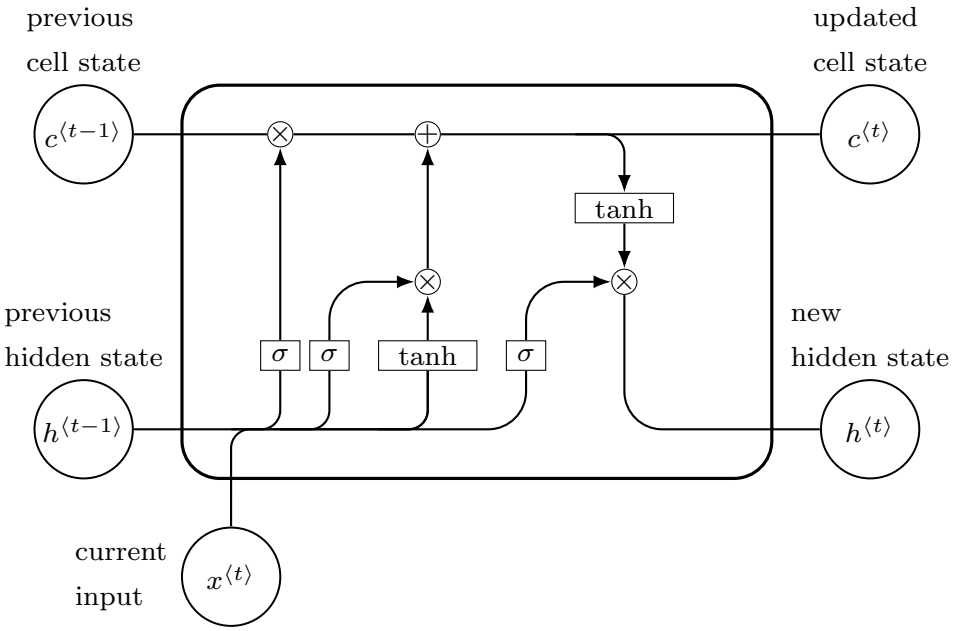


Figure 3.5: LSTM cell and it's operations. The sigmoid function $\sigma$ takes as input in all gates both the previous hidden state $h^{\langle t-1\rangle}$ and the current input $x^{\langle t\rangle}$. Together with the first activation function the forget gate is formed. The second activation function, as well as the *tanh* function, which gets the same inputs, form the input gate, which is needed to update the previous cell state $c^{\langle t-1\rangle}$. The last activation function together with the updated cell state $c^{\langle t\rangle}$, which is given into a *tanh* function, forms the output gate. The new hidden state $h^{\langle t\rangle}$ contains all the essential information that the network needs to incorporate past calculations into the computations for the new cell.

In this thesis, so-called CNN-LSTM architectures are also used. These are a class of models that are well suited for time series prediction problems. The basic architecture combines a deep visual feature extractor, a CNN, with a model that recognizes temporal dynamics and weights them appropriately with respect to their feature importance (RNN). More details can be found in [28].

## 3.2 Simulation-Driven Models

This chapter focuses on the development of the machine learning models and their evaluation. First, the hexagonal binning method is explained, which is used to convert the hexagonal structure of the PMT arrays into rectangular space. After detailing the datasets created using Monte Carlo simulations, the training of the deep learning architectures is analyzed. The different neural networks are compared, and only the best model architectures are used for further analysis. Two classical position reconstruction algorithms are used as comparison to confirm the performance of the newly developed deep learning methods. Finally, the best performing model for each dataset is examined more closely for systematic effects.

### 3.2.1 MC Training Datasets

The PMTs on the top and bottom array in the TPC are arranged in a hexagonal structure. This hexagonal arrangement has the highest packing density so that the light collection efficiency of the PMTs can be maximized [12]. Since the pixels of an image given as input to the networks have a rectangular grid structure, the different layers and operations (kernels etc.) of the previously described neural networks for image-related tasks have been implemented for rectangular space. For this reason, the hexagonal grid must be converted into a rectangular representation. A visualization of this transformation can be seen in figure 3.6. The data is stored in such a way that neighborhood relationships between pixels are preserved. As few padding zeros as possible are added so that additional required computation time is minimized [16].

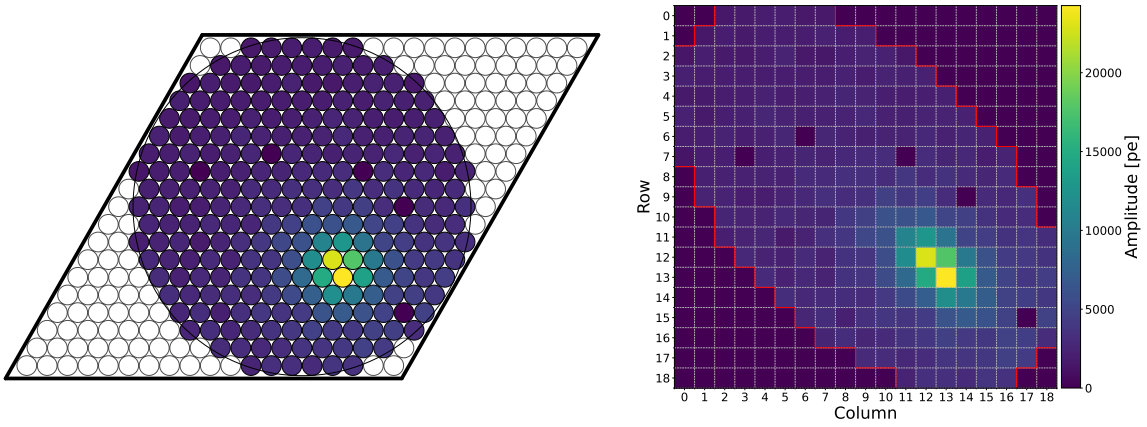


Figure 3.6: Transformation of the hexagonal grid into rectangular space. (left) Padding zeros (empty circles) are added to the original size to fill the parallelogram. (right) By moving the rows, they are aligned to each other. The rectangular representation is obtained, which can be given as input to a neural network. The area outlined in red contains all the information of the PMTs from the structure on the left.

In this thesis three different datasets are used, which were generated by XENONnT Monte Carlo (MC) simulations. The sets each consist of tensors of a fixed size. In the following, the final structure of the datasets is described and the extent to which they differ from one another is discussed:

First of all, the structure will be discussed in general. The datasets generally have the form $(t, \mathrm{PMT_x}, \mathrm{PMT_y}, \mathrm{PMT_{arrays}})$, where the first dimension describes the number of time steps $t$ for a single event. The second and third dimension represent the hexagonal PMT arrangement transformed into a regular matrix. These entries are identical for all records, it being a $20 \times 20$ matrix. The last dimension refers to the number of PMT arrays considered.

1) Dataset 1 (DS1) contains S2 top only signals, i. e. signals registered by the top PMT array in the TPC. It has the shape $(54, 20, 20, 1)$. While the first dimension represents the 54 different time steps, the last dimension indicates that only signals from the top PMT array are registered here.

2) Dataset 2 (DS2) includes S2 top and bottom signals. Here, the signals were detected from either the top or the bottom PMT array. It is of the shape $(136, 20, 20, 2)$. Both arrays are considered in the last dimension, and the number of time steps is increased compared to the other two datasets.

3) Dataset 3 (DS3) contains signals from both the top and bottom PMT arrays. Only the prompt scintillation signal S1 is considered. It takes the shape $(54, 20, 20, 2)$.

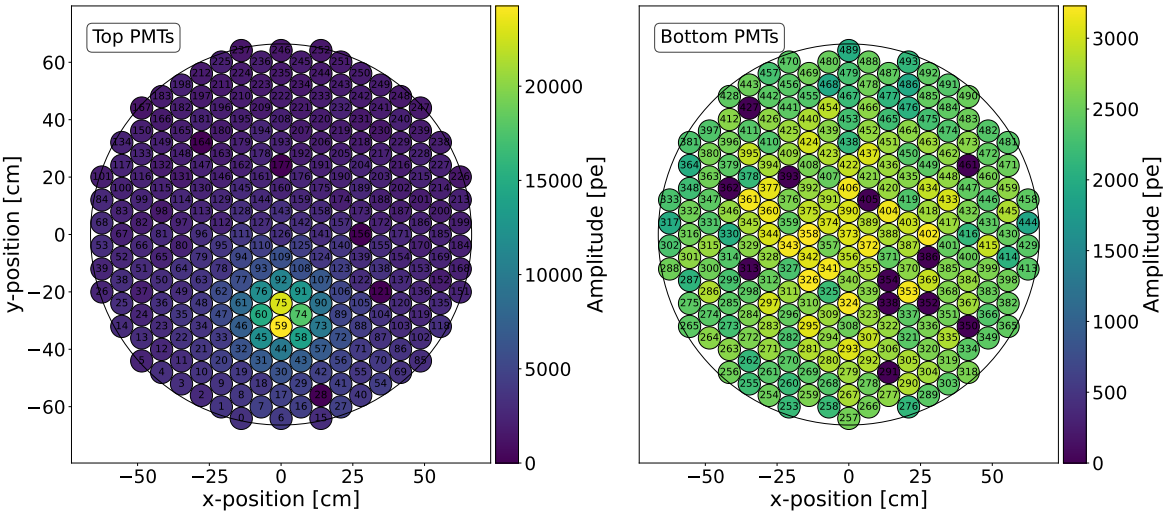


Figure 3.7: Example PMT pattern from DS2, summed over all 136 time steps. (left) Top PMT array with clear amplitude maximum, the $x$-$y$-position can be well determined. (right) Bottom PMT array with scattered amplitude distribution.

Figure 3.7 shows an event from DS2. The amplitudes registered by the 253 (top array) and 241 (bottom array) PMTs were summed over all time steps. The difference between the

pattern on the top PMT array (left) and the bottom PMT array (right) is clearly visible: While the electrons on the former are predominantly detected locally at one location, so that one can well determine the $x$-$y$-position of the event, there is a broad distribution on the bottom PMTs. Photons scatter in the TPC multiple times until they are registered in the bottom array. In addition, they are reflected at the liquid-gas interface. This leads to the wide spread of the scintillation light on the bottom PMT array. These patterns still need to be transformed into rectangular structure using hexagonal binning to feed them as input images to a neural network.

Since the focus in this work is on the 3D position reconstruction, it is essential to mention in which ranges the respective dimensions extend in the simulated data. The $x$- and $y$-dimensions are the same for all of them, the values range from $-49\,\mathrm{cm}$ to $49\,\mathrm{cm}$. While the $z$-value for DS1 varies only from $-99\,\mathrm{cm}$ to $-20\,\mathrm{cm}$, it reaches a maximum negative value of $-139\,\mathrm{cm}$ for DS2 and DS3. By selecting only events in the innermost part of the detector, this study focuses on avoiding the strongest inhomogeneities of the detector. In a further analysis, the other areas could also be examined. The number of training and validation examples for all are listed in table 3.1.

Table 3.1: Overview of the simulated datasets. The training samples are used to train the neural networks, the validation samples to evaluate their performance.

| Datasets | S2 top | S2 top & bottom | S1 top & bottom |
|---|---|---|---|
| Number of training samples | 1,350,000 | 705,000 | 1,920,000 |
| Number of validation samples | 140,000 | 45,000 | 80,000 |

### 3.2.2 Model Selection and Training

In this section, the training of the different neural networks is evaluated. Each dataset is tested with three fundamentally different model architectures and only the best architecture is used for further analysis. A convolutional neural network (CNN), a recurrent neural network (RNN) as a Long Short-Term Memory (LSTM), and a RNN as a CNN-LSTM were used as the model structure in each case. The exact, best performing model structures were determined by testing various hyperparameters. The performance of the models is compared here using the *mean squared error* for the validation sets, as explained earlier in section 3.1.2. All models presented in this section were developed using the open source machine learning library Tensorflow [13], with the Keras [7] interface running on top of Tensorflow. The code written for this thesis has finally been executed on the computer cluster PALMA-II [14] of the University of Münster. PALMA stands for "Paralleles Linux-System für Münsteraner Anwender" ("Parallel Linux System for Münster's Users"). With the use of NVIDIA GeForce

RTX 2080 Ti GPUs, it was possible to keep the required computing time within reasonable limits.

Table 3.2: Layer structure of the CNN used to reconstruct the S2 top only data. In total 292,131 trainable parameters. *None* does not specify the size of the dimension, allowing a variable batch size.

| | Layer (type) | Output shape | Filters | Kernel size | Parameters |
|---|---|---|---|---|---|
| 1 | Input | (None, 54, 20, 20, 1) | - | - | 0 |
| 2 | ZeroPadding3D | (None, 58, 24, 24, 1) | - | - | 0 |
| 3 | Conv3D_1 | (None, 58, 24, 24, 16) | 16 | $3 \times 3 \times 3$ | 448 |
| 4 | MaxPooling3D_1 | (None, 29, 12, 12, 16) | - | $2 \times 2 \times 2$ | 0 |
| 5 | Conv3D_2 | (None, 29, 12, 12, 32) | 32 | $3 \times 3 \times 3$ | 13,856 |
| 6 | MaxPooling3D_2 | (None, 14, 6, 6, 32) | - | $2 \times 2 \times 2$ | 0 |
| 7 | Conv3D_3 | (None, 14, 6, 6, 64) | 64 | $3 \times 3 \times 3$ | 55,360 |
| 8 | MaxPooling3D_3 | (None, 7, 3, 3, 64) | - | $2 \times 2 \times 2$ | 0 |
| 9 | Conv3D_4 | (None, 7, 3, 3, 128) | 128 | $3 \times 3 \times 3$ | 221,312 |
| 10 | MaxPooling3D_4 | (None, 3, 1, 1, 128) | - | $2 \times 2 \times 2$ | 0 |
| 11 | Flatten | (None, 384) | - | - | 0 |
| 12 | Output (Dense) | (None, 3) | - | - | 1155 |
| | | | | Total: | 292,131 |

The evaluation for DS1 is presented in this section, the architectures and analyses of the other datasets are given in appendix A. In this thesis, unless otherwise stated, the performances of the reconstruction of the $y$-position are not presented, as they are analogous to that of the $x$-position. In table 3.2, the architecture of the CNN used for DS1 is summarized. Max pooling was chosen as the pooling method because it consistently produced better results than average pooling. In addition to the convolutional and pooling layers, there is also a ZeroPadding layer at the beginning of the network to avoid shrinking and information loss at the edges of an image. Other layers, such as dropout layers [31], which are an effective regularization method to prevent overfitting and improve the generalization error, have not been added to the architecture. This is justified by several test runs considering architectures both with and without these layers. The best validation loss could not be exceeded by the models with dropout layers. No overfitting occurred even for the architectures that did not consider these layers (see figure 3.9). It is also confirmed by an analysis from XENON1T [9]. The architectures of the LSTM and CNN-LSTM models are shown in table A.1 and table A.2, respectively. The LSTM architecture involves using a LSTM as well as several convolutional

layers for feature extraction on input data. With this combination it is possible to process sequences of input images. Each convolutional layer is wrapped in a TimeDistributed (TD) layer, which allows a layer to be applied to each temporal segment of the input data. Unlike the LSTM model, the CNN-LSTM structure combines a 2D convolution with a LSTM in the ConvLSTM2D layers: A hidden state is created between the steps.

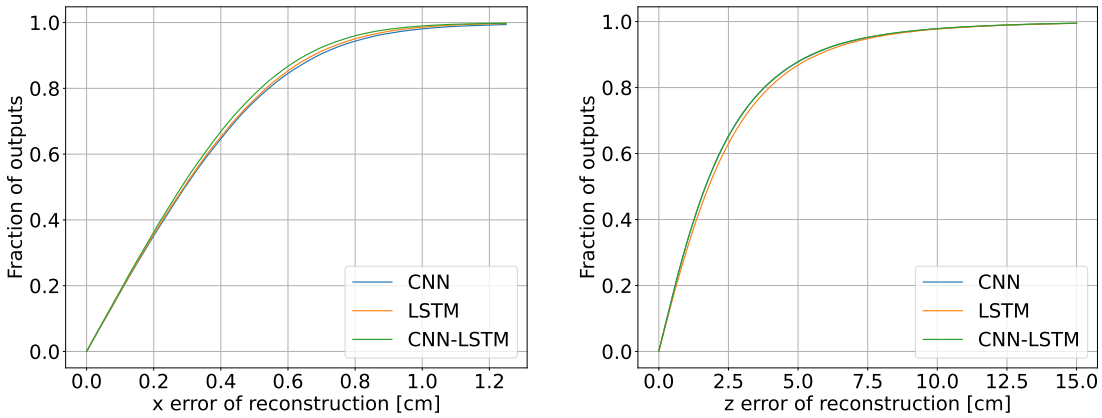


Figure 3.8: The cumulative distribution of reconstruction errors for DS1 is compared for the different architectures. The steeper the curve, the better the model performs.

The reconstruction performance of the different neural networks for the validation samples from DS1 can be seen in figure 3.8. Even though the range of the reconstruction error in the $x$-dimension (left) is significantly smaller than that in the $z$-dimension (right), the reconstruction of the depth of an event is a great success. It can be determined to within a few cm using the models without the time difference between the S1 and S2 signals. Although the performances do not show significant differences, the fraction of outputs from the CNN-LSTM architecture for the continuous range of the reconstruction error is larger than for the other two structures, in each dimension. Despite differing by less than 3 % in the proportion when comparing the models, the CNN-LSTM structure is used for further evaluation. This is justified by the fact that using the CNN or the LSTM model does not yield meaningful differences in the subsequent evaluation.

Table 3.3: Comparison of the best validation losses of the neural networks used for the different simulated datasets: Based on the metric used, MSE, the appropriate architecture is used for further evaluation, marked with a cross.

| Simulated dataset | CNN | LSTM | CNN-LSTM |
|---|---|---|---|
| S2 top (DS1) | 4.40 | 4.67 | 4.37 (×) |
| S2 top & bottom (DS2) | 1.83 | 1.99 | 1.81 (×) |
| S1 top & bottom (DS3) | 23.22 (×) | 28.05 | 803.59 |

This analysis was performed for all three datasets. Table 3.3 shows the best validation losses for each run executed on the PALMA cluster. The losses for each epoch, the number of complete runs through the training set, were calculated and the model was saved at the epoch that had the smallest validation loss. In the table, the model architectures that performed best on each dataset are marked with a cross respectively. Only these are needed for further analysis. In figure 3.9, the training and validation losses of the three best performing models are plotted against the epochs for confirmation. One can see that no overfitting occurs, and the best iterations have been highlighted. When the models were run on the cluster, two additional parameters were considered: Training was stopped if the validation loss did not improve after 30 epochs. This was the case for DS3, where the training ended after 47 epochs. The training for DS1 was completed after 50 epochs, that for DS2 after 2 days due to time constraints on the cluster. An improvement of the MSE loss was no longer expected in either case, as can be seen in figure 3.9. In addition, a learning rate change was applied to the training of DS1 and DS3. The learning rate was reduced by one tenth after 4 epochs if the validation loss did not show improvements. This parameter was considered because models often benefit from a learning rate reduction when training remains stagnant [25].

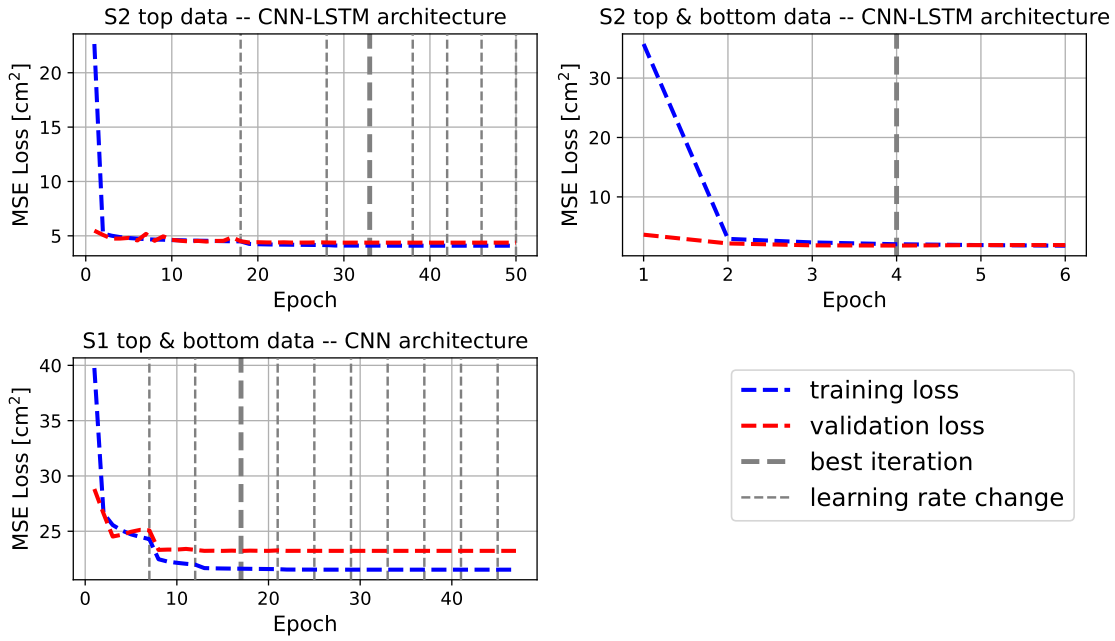


Figure 3.9: Training and validation history of the best performing architectures in this analysis. The epoch-dependent loss can be seen, the models are stored according to the epoch with the lowest validation loss (best iteration). The learning rate is reduced several times shown as dashed lines. A significant drop in validation loss is not visible in the graphs, as this occurs in the first epoch due to the large number of samples.

In light of the results presented in this section, the evaluation can be summarized. Even though the overall validation loss is the smallest for DS2 (S2 top & bottom data), DS1 (S2 top only data) performs better for a reconstruction desired only in the $x$- and $y$- dimensions.

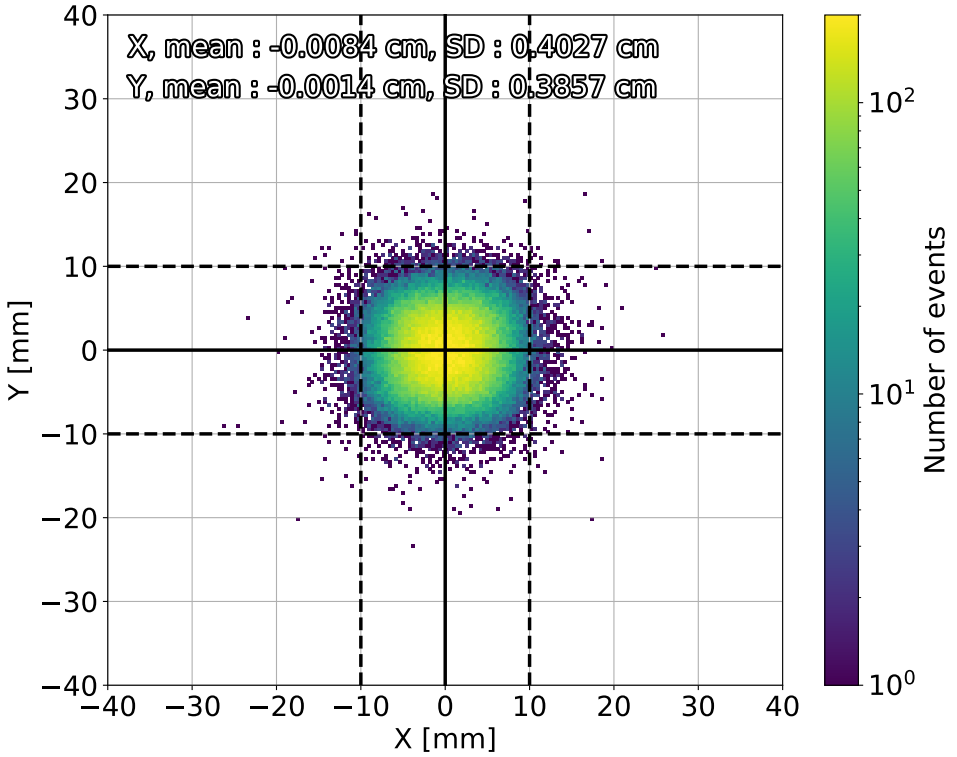


Figure 3.10: Distribution of the localization error in $x$- and $y$-direction of the CNN-LSTM architecture for DS1. The mean error and standard deviation (SD) are given in cm respectively, the $y$-reconstruction has a smaller SD.

Looking at figure 3.10, the $X$ and $Y$ means are very close to 0 cm, there is almost no bias. Furthermore, the distribution is symmetric about the center $(0, 0)$, the reconstruction errors are not correlated with each other. Thus, an accurate reconstruction for the two-dimensional position of an event is possible. Nevertheless, this result was not expected as the events in DS2 include more information than the events in DS1. With the number of input samples $n$, targets $Y$ and the predictions $Y'$, one can define the metric *root mean squared error* (RMSE):

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(Y_i - Y_i')^2} \tag{3.6}$$

Using this metric the predicted data points are off by about 0.40 cm (DS1) or 0.51 cm (DS2) in the $x$-, and 0.39 cm (DS1) or 0.68 cm (DS2) in the $y$-direction. The larger RMSE for DS2 compared to DS1 in the $(x, y)$-dimension can be explained by the fact that the performance for the depth of an event is significantly better. While the RMSE is 3.58 cm for DS1, it

equals 2.17 cm for DS2. Since the algorithm was not given any restrictions, such as first determining the two-dimensional position of an event as accurately as possible before the depth is predicted as the third value, it learns by itself (supervised learning) and only tries to minimize the overall validation loss. However, this is not a disadvantage for this evaluation. On the contrary, with the DS2 it is possible to forecast the depth of an event to within a few cm. Over 80 % of all validation samples have a reconstruction error in depth of 2 cm or less. Thus, with this CNN-LSTM architecture, the depth of an event can be projected for these simulated data without relying on the time difference between the S1 and S2 signals. Only registered S2 signals are used here. This result is confirmed by the analyses presented in figure A.8 and figure A.9. The performance of DS3, the S1 signals, is significantly worse than that of DS1 and DS2, especially in the planar reconstruction of events. However, this was to be expected since the S1 signal does not distribute the light as localized as the S2 signal, see section 2.3. Therefore, the $(x, y)$-position of the interaction point in XENONnT is inferred by localized PMT hit patterns from S2 signals on the top PMT array [12].

## 3.3 Conventional Methods for Position Reconstruction – Comparison

In this section, rather simple, conventional methods for position reconstruction are used. They are limited to the planar reconstruction of an event, and use only signals detected by the top PMT array. The goal is to compare the reconstruction accuracies of the classical approaches with those of the deep learning models developed in section 3.2.2. Only DS2 is considered, the tensors are summed up with respect to the dimension of the time steps, so that an event has the PMT hit pattern over all 136 steps. This data is compared with 2 different methods, which will be presented in the following:

**Maximum value** After summing up the amplitudes over the time dimension, the $(x, y)$-position of the PMT with the largest amplitude is assumed to be the position of the event. An example of how the reconstruction looks like with this method is shown on the left side in figure 3.11.

**Weighted mean** As with the *maximum value* method, the position of the PMT that registers the largest amplitude is determined first. Finally, a weighted average is calculated using the position of this PMT and the positions of the 18 closest PMTs by weighting each position with the registered amplitude. This PMT selection covers the region where most of the S2 light is detected. This can be seen on the right side in figure 3.11. The number of PMTs considered was limited to 18 instead of using the full PMT array to avoid a potential bias from the finite PMT array size. For example, if an event takes place further to the right on the PMT array ($x > 0$), the positions of all PMTs further to the left ($x < 0$) are weighted almost equally, since they register approximately the same number of photoelectrons. The $x$-position would be incorrectly assigned a smaller value during event reconstruction due to the limited size of the PMT array on the right.

Figure 3.12 shows the reconstruction for a single random event from DS2. The actual position is marked with a red cross. The deep learning-based architecture deviates only slightly from the correct position in $x$ and $y$, giving a good position reconstruction. While the *maximum value* method shows a larger deviation especially in the $x$-coordinate, the *weighted mean* method performs marginally better.

Figure 3.13 shows a quantitative comparison between the 3 methods applied to DS2. Using the same notation as for the RMSE, the metric *mean absolute error* (MAE) can be defined as

$$\mathrm{MAE} = \frac{1}{n} \sum_{i=1}^{n} \left| Y_i - Y_i' \right| . \tag{3.7}$$

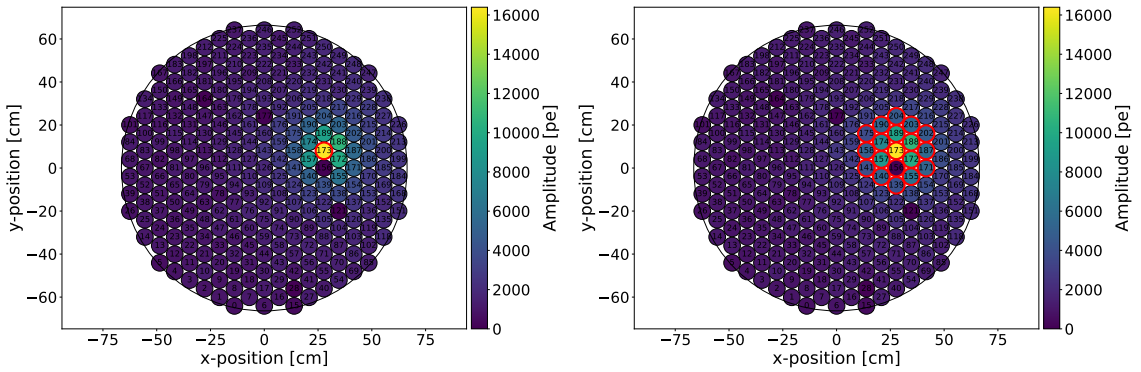


Figure 3.11: Illustration of the two classical techniques used for reconstruction. (left) *Maximum value* method: Only the PMT that registers the largest amplitude is included in the position calculation. (right) *Weighted mean* method: Both the PMT with the largest detected amplitude and the 18 closest PMTs have an influence on the reconstructed position.

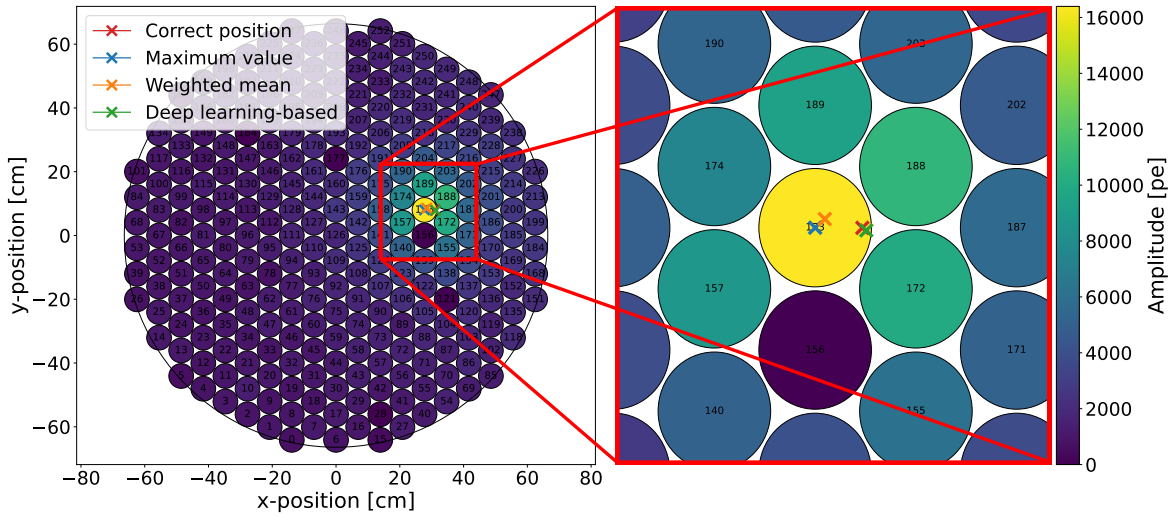


Figure 3.12: Comparison of the 3 reconstructions for a single event from DS2. The *maximum value* method (blue cross) reconstructs the position exactly at the position of the PMT with the largest registered amplitude. While the *weighted mean* method (orange cross) deviates slightly from this, the CNN-LSTM structure (green cross) comes closest to the actual position (red cross).

With a MAE of 0.41 cm, the deep learning-based model performs best and should be preferred to the classical approaches. The *weighted mean* method with a MAE of 1.67 cm still performs better than the *maximum value* method, which has a MAE of 2.29 cm. The small error of the deep learning model can thus be confirmed, it outperforms the other two approaches. In a further investigation, it would be of great interest to compare the CNN-LSTM architecture with existing position reconstruction methods for XENONnT. It should be mentioned that these methods are primarily intended for reconstruction in the $(x, y)$-dimension. However, this is beyond the scope of this thesis.

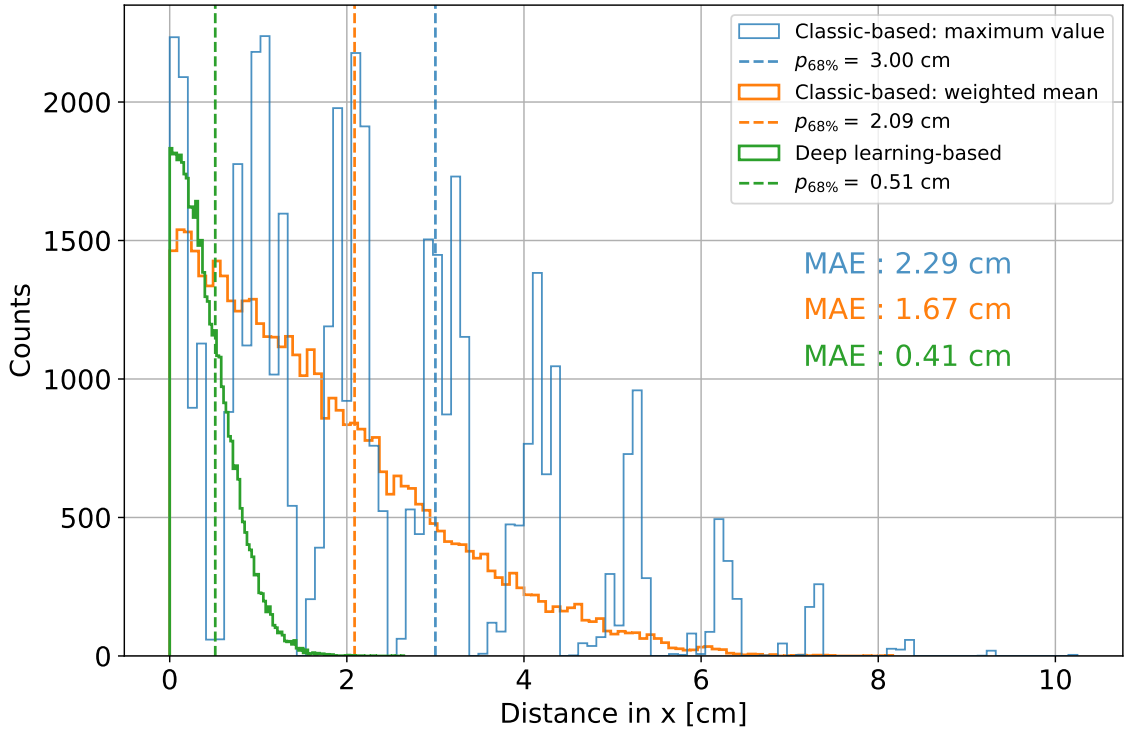


Figure 3.13: Comparison of the classical methods (blue and orange) with the CNN-LSTM structure (green) from DS2. The dashed vertical lines delimit the area in which 68 % of all validation samples lie, the *mean absolute error* is additionally shown. Peaks occur with the *maximum value* method, since only discrete values can be assumed here.

## 3.4 Performances for different Depths and Amplitudes

Examining the performance of the simulation-driven models in different zones in the detector is fundamental to identifying systematic effects. Moreover, non-uniformities can be detected for the position reconstruction of the events in the TPC. In this section, the error in the reconstruction will be analyzed especially for different event depths and amplitudes.

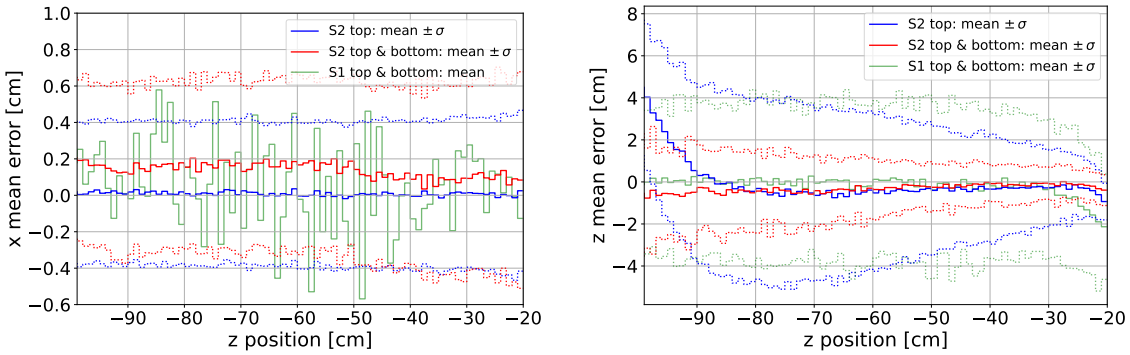


Figure 3.14: Performances of the trained models with respect to the $x$-position (left) and $z$-position (right) for DS1 (blue), DS2 (red), and DS3 (green) for varying $z$-positions of the simulated events. The standard deviation is also marked with dotted lines, for DS3 on the left side it is not visible due to a too large deviation from the other errors.

In figure 3.14, the left side shows the $x$ mean error calculated as difference between the true and reconstructed $x$-position plotted against the $z$-position of the events. It is noticeable here that the mean error of DS2 is shifted to the positive. The $x$-position predicted with the CNN-LSTM architecture thus always has a larger average value than the actual $x$-position. In the same figure, the $z$ mean error is plotted against the $z$-position of the events on the right side. The CNN-LSTM architecture of DS1 has a particularly high reconstruction error for a small $z$-position, $z < -85\,\text{cm}$, which becomes maximum for the maximum value of $z = -99\,\text{cm}$. This can be accounted for by considering only the top array in DS1. A subtle tendency in this direction can also be seen for DS2, where the error in the range $-50\,\text{cm} < z < -30\,\text{cm}$ is smaller than for depths $z < -50\,\text{cm}$.

Furthermore, the dependence of the $z$ mean error on the amplitude for DS2 is investigated. Figure 3.15 shows that a smaller reconstruction error in $z$ can be achieved for events with larger simulated amplitudes. For small amplitudes with $0.1 \times 10^6$ or less photoelectrons, the reconstruction error into the negative drastically increases, the predicted event depth is on average smaller than the correct depth of the events.

The previously described research leads to the conclusion that it would be useful to expand the datasets in specific areas. For example, more events with a depth of $z < -85\,\text{cm}$ could be

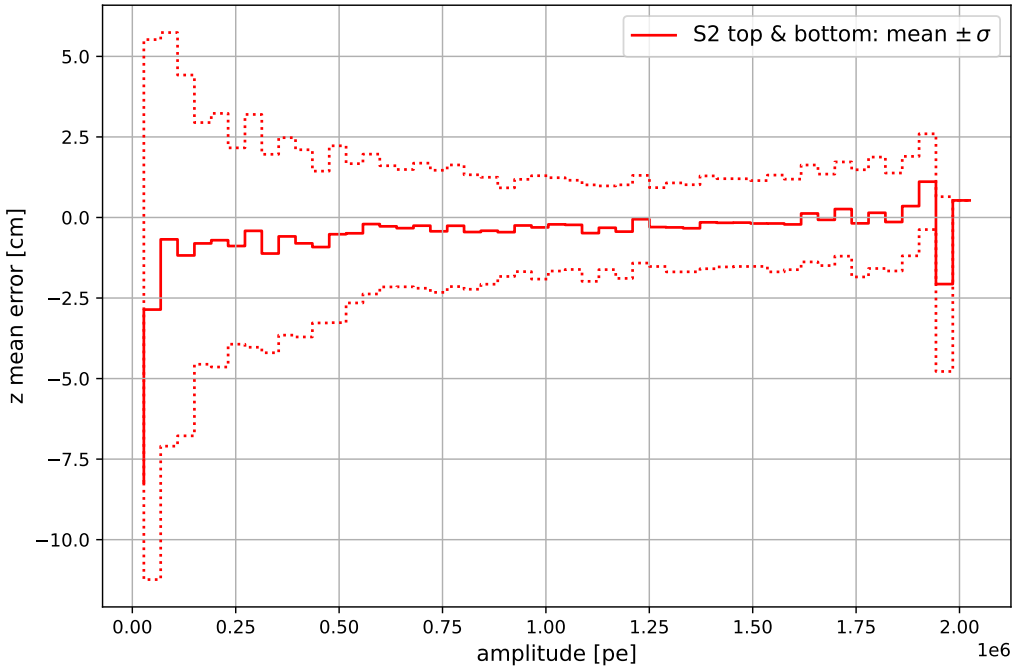


Figure 3.15: Plotted is the average $z$-error compared to different signal amplitudes. In particular, a large reconstruction error can be observed at very small amplitudes.

included in DS2 allowing to observe how the reconstruction error of the model changes in this range. Possibly, an adjustment of the deep learning models for this range is useful and also necessary to minimize the error. Events with an amplitude of $0.1 \times 10^6$ or less photoelectrons should also be considered in a separate training. As mentioned in section 3.2.1, the simulated data avoid the strongest inhomogeneities in the detector. In a further investigation, the training of the developed architectures should additionally consider these areas. In particular, a verification should be performed whether the models provide similar results here.

# 4 Conclusion and Outlook

In this thesis, a three-dimensional position reconstruction for XENONnT data using deep learning methods was investigated. It was shown that the depth of an event can be reconstructed to a reasonable accuracy using a single S1 or S2 signal, in contrast to the classical method relying on the time difference between S1 and S2. In addition, a position reconstruction for S1 signals was presented.

It was outlined in chapter two that there is plenty of evidence for the existence of Dark Matter. Up to now, there is no direct detection yet. One of the experiments aiming at the direct detection of WIMP Dark Matter is the XENONnT experiment. The experiment along with its dual-phase time projection chamber was presented in this chapter. In XENONnT, background reduction by fiducialization takes place by considering only events in the innermost part of the detector. Therefore, the position of an event in the detector must be determined as accurately as possible. It was reconstructed in this thesis using deep learning techniques introduced in chapter three. Three different models were used: a convolutional neural network, a long short-term memory and a combination of both. While in CNNs each input is processed independently, LSTM layers have a cell state that can incorporate information from previous time steps into ongoing computations. The application of three different datasets allowed the differentiation between S2 top only (DS1), S2 top & bottom (DS2), and S1 top & bottom (DS3) signals. As a result, the depth of an event from DS2 could be predicted within a few cm, over 80 % of the validation data has a reconstruction error of 2 cm or less. This enables the depth to be determined without the help of the time difference between S1 and S2. The 3D position was also reconstructed by using only S1 signals. In another section, the architecture for DS2 was compared with two classical methods, *maximum value* and *weighted mean*. It was confirmed that the deep learning-based model outperforms these approaches.

The developed deep learning models can potentially be used in the future for a three-dimensional position reconstruction of either S1 or S2 signals only. In a further analysis, improvements and comparisons can be made in several fields. For example, training could be performed using only events from DS2 with an amplitude of $0.1 \times 10^6$ or less photoelectrons to potentially make a change to the deep learning architectures in this area. Data more affected by inhomogeneities in the detector could be examined.

# A Appendix

## Model Architectures and Evaluation of the Neural Networks

Table A.1: Layer structure of the LSTM used to reconstruct the S2 top only data. In total 5,835,651 trainable parameters. *None* does not specify the size of the dimension, allowing a variable batch size. In the LSTM layer, the last output is returned in the output sequence instead of the full sequence.

| | Layer (type) | Output shape | Filters | Kernel size | Parameters |
|---|---|---|---|---|---|
| 1 | Input | (None, 54, 20, 20, 1) | - | - | 0 |
| 2 | ZeroPadding3D | (None, 58, 24, 24, 1) | - | - | 0 |
| 3 | TD(Conv2D_1) | (None, 58, 24, 24, 16) | 16 | $7 \times 7$ | 800 |
| 4 | MaxPooling3D_1 | (None, 29, 12, 12, 16) | - | $2 \times 2 \times 2$ | 0 |
| 5 | TD(Conv2D_2) | (None, 29, 12, 12, 32) | 32 | $5 \times 5$ | 12,832 |
| 6 | MaxPooling3D_2 | (None, 14, 6, 6, 32) | - | $2 \times 2 \times 2$ | 0 |
| 7 | TD(Conv2D_3) | (None, 14, 6, 6, 64) | 64 | $5 \times 5$ | 51,264 |
| 8 | TD(Flatten) | (None, 14, 2304) | - | - | 0 |
| 9 | LSTM_1 | (None, 512) | - | $2 \times 2 \times 2$ | 5,769,216 |
| 10 | Output (Dense) | (None, 3) | - | - | 1539 |
| | | | | Total: | 5,835,651 |

Table A.2: Layer structure of the CNN-LSTM used to reconstruct the S2 top only data. In total 439,875 trainable parameters. *None* does not specify the size of the dimension, allowing a variable batch size. In the ConvLSTM2D layers, the last output is returned in the full sequence instead of the output sequence.

| | Layer (type) | Output shape | Filters | Kernel size | Parameters |
|---|---|---|---|---|---|
| 1 | Input | (None, 54, 20, 20, 1) | - | - | 0 |
| 2 | ZeroPadding3D | (None, 58, 24, 24, 1) | - | - | 0 |
| 3 | ConvLSTM2D_1 | (None, 58, 24, 24, 8) | 8 | $7 \times 7$ | 14,144 |
| 4 | MaxPooling3D_1 | (None, 29, 12, 12, 8) | - | $2 \times 2 \times 2$ | 0 |
| 5 | ConvLSTM2D_2 | (None, 29, 12, 12, 16) | 16 | $5 \times 5$ | 38,464 |
| 6 | MaxPooling3D_2 | (None, 14, 6, 6, 16) | - | $2 \times 2 \times 2$ | 0 |
| 7 | ConvLSTM2D_3 | (None, 14, 6, 6, 32) | 32 | $5 \times 5$ | 153,728 |
| 8 | MaxPooling3D_3 | (None, 7, 3, 3, 32) | - | $2 \times 2 \times 2$ | 0 |
| 9 | ConvLSTM2D_4 | (None, 7, 3, 3, 64) | 64 | $3 \times 3$ | 221,440 |
| 10 | Flatten | (None, 4032) | - | - | 0 |
| 11 | Output (Dense) | (None, 3) | - | - | 12,099 |
| | | | | Total: | 439,875 |

Table A.3: Layer structure of the CNN used to reconstruct the S2 top & bottom data. In total 294,483 trainable parameters. *None* does not specify the size of the dimension, allowing a variable batch size.

| | Layer (type) | Output shape | Filters | Kernel size | Parameters |
|---|---|---|---|---|---|
| 1 | Input | (None, 136, 20, 20, 2) | - | - | 0 |
| 2 | ZeroPadding3D | (None, 140, 24, 24, 2) | - | - | 0 |
| 3 | Conv3D_1 | (None, 140, 24, 24, 16) | 16 | $3 \times 3 \times 3$ | 880 |
| 4 | MaxPooling3D_1 | (None, 70, 12, 12, 16) | - | $2 \times 2 \times 2$ | 0 |
| 5 | Conv3D_2 | (None, 70, 12, 12, 32) | 32 | $3 \times 3 \times 3$ | 13,856 |
| 6 | MaxPooling3D_2 | (None, 35, 6, 6, 32) | - | $2 \times 2 \times 2$ | 0 |
| 7 | Conv3D_3 | (None, 35, 6, 6, 64) | 64 | $3 \times 3 \times 3$ | 55,360 |
| 8 | MaxPooling3D_3 | (None, 17, 3, 3, 64) | - | $2 \times 2 \times 2$ | 0 |
| 9 | Conv3D_4 | (None, 17, 3, 3, 128) | 128 | $3 \times 3 \times 3$ | 221,312 |
| 10 | MaxPooling3D_4 | (None, 8, 1, 1, 128) | - | $2 \times 2 \times 2$ | 0 |
| 11 | Flatten | (None, 1024) | - | - | 0 |
| 12 | Output (Dense) | (None, 3) | - | - | 3075 |
| | | | | Total: | 294,483 |

Table A.4: Layer structure of the LSTM used to reconstruct the S2 top & bottom data. In total 5,836,435 trainable parameters. *None* does not specify the size of the dimension, allowing a variable batch size. In the LSTM layer, the last output is returned in the output sequence instead of the full sequence.

| | Layer (type) | Output shape | Filters | Kernel size | Parameters |
|---|---|---|---|---|---|
| 1 | Input | (None, 136, 20, 20, 2) | - | - | 0 |
| 2 | ZeroPadding3D | (None, 140, 24, 24, 2) | - | - | 0 |
| 3 | TD(Conv2D_1) | (None, 140, 24, 24, 16) | 16 | $7 \times 7$ | 1584 |
| 4 | MaxPooling3D_1 | (None, 70, 12, 12, 16) | - | $2 \times 2 \times 2$ | 0 |
| 5 | TD(Conv2D_2) | (None, 70, 12, 12, 32) | 32 | $5 \times 5$ | 12,832 |
| 6 | MaxPooling3D_2 | (None, 35, 6, 6, 32) | - | $2 \times 2 \times 2$ | 0 |
| 7 | TD(Conv2D_3) | (None, 35, 6, 6, 64) | 64 | $5 \times 5$ | 51,264 |
| 8 | TD(Flatten) | (None, 35, 2304) | - | - | 0 |
| 9 | LSTM_1 | (None, 512) | - | $2 \times 2 \times 2$ | 5,769,216 |
| 10 | Output (Dense) | (None, 3) | - | - | 1539 |
| | | | | Total: | 5,836,435 |

Table A.5: Layer structure of the CNN-LSTM used to reconstruct the S2 top & bottom data. In total 458,723 trainable parameters. *None* does not specify the size of the dimension, allowing a variable batch size. In the ConvLSTM2D layers, the last output is returned in the full sequence instead of the output sequence.

| | Layer (type) | Output shape | Filters | Kernel size | Parameters |
|---|---|---|---|---|---|
| 1 | Input | (None, 136, 20, 20, 2) | - | - | 0 |
| 2 | ZeroPadding3D | (None, 140, 24, 24, 2) | - | - | 0 |
| 3 | ConvLSTM2D_1 | (None, 140, 24, 24, 8) | 8 | $7 \times 7$ | 15,712 |
| 4 | MaxPooling3D_1 | (None, 70, 12, 12, 8) | - | $2 \times 2 \times 2$ | 0 |
| 5 | ConvLSTM2D_2 | (None, 70, 12, 12, 16) | 16 | $5 \times 5$ | 38,464 |
| 6 | MaxPooling3D_2 | (None, 35, 6, 6, 16) | - | $2 \times 2 \times 2$ | 0 |
| 7 | ConvLSTM2D_3 | (None, 35, 6, 6, 32) | 32 | $5 \times 5$ | 153,728 |
| 8 | MaxPooling3D_3 | (None, 17, 3, 3, 32) | - | $2 \times 2 \times 2$ | 0 |
| 9 | ConvLSTM2D_4 | (None, 17, 3, 3, 64) | 64 | $3 \times 3$ | 221,440 |
| 10 | Flatten | (None, 9792) | - | - | 0 |
| 11 | Output (Dense) | (None, 3) | - | - | 29,379 |
| | | | | Total: | 458,723 |

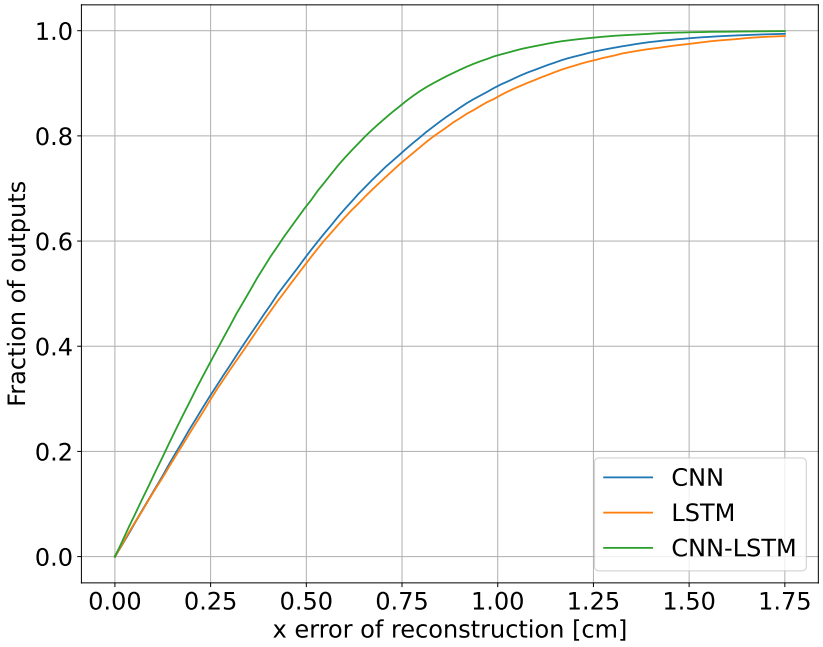


Figure A.1: The cumulative distribution of reconstruction errors in the $x$-direction for DS2 is compared for the different architectures. The steeper the curve, the better the model performs. The CNN-LSTM model, whose underlying architecture is shown in table A.5, performs best.

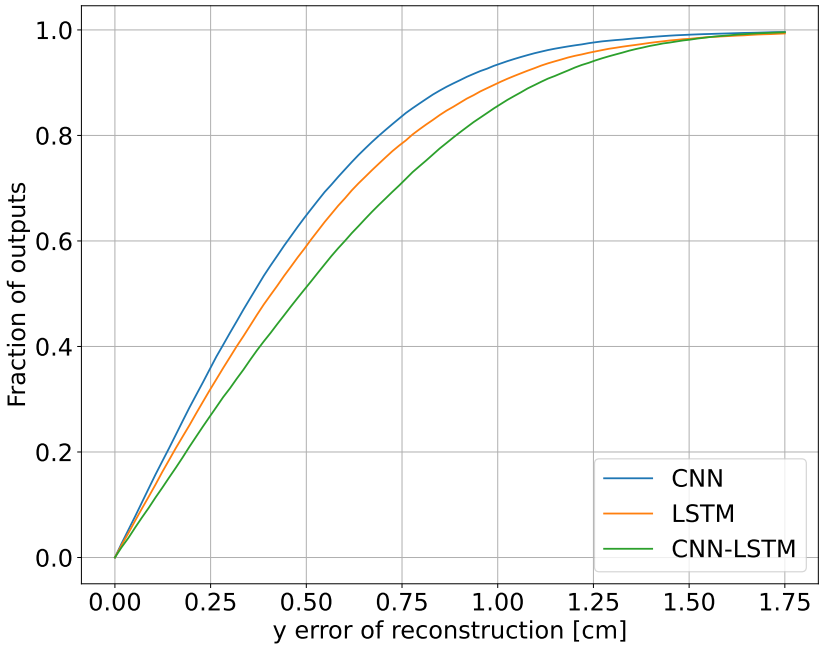


Figure A.2: The cumulative distribution of reconstruction errors in the $y$-direction for DS2 is compared for the different architectures. The steeper the curve, the better the model performs. The CNN model, whose underlying architecture is shown in table A.3, performs best. In contrast to the other results, the performance deviates here, compared to the $x$-direction.

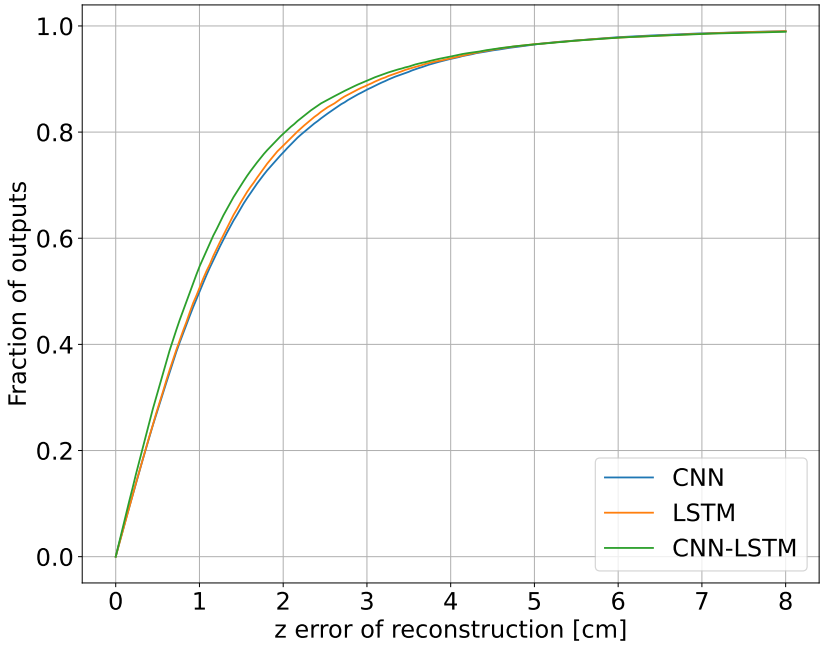


Figure A.3: The cumulative distribution of reconstruction errors in the $z$-direction for DS2 is compared for the different architectures. The steeper the curve, the better the model performs. The CNN-LSTM model, whose underlying architecture is shown in table A.5, performs best.

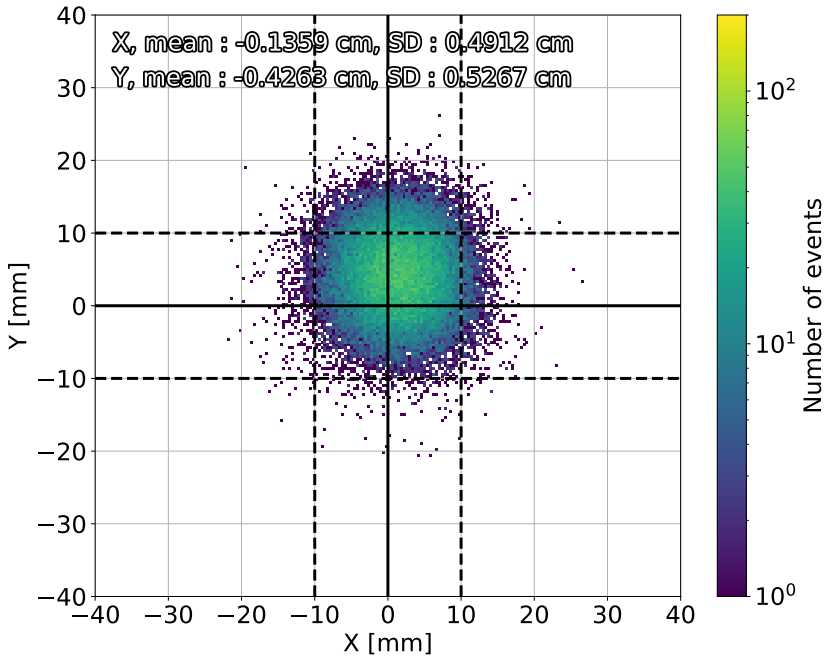


Figure A.4: Distribution of the localization error in $x$- and $y$-direction of the CNN-LSTM architecture for DS2. The mean error and standard deviation (SD) are given in cm respectively, the $x$-reconstruction has a smaller SD.

Table A.6: Layer structure of the CNN used to reconstruct the S1 top & bottom data. In total 292,563 trainable parameters. *None* does not specify the size of the dimension, allowing a variable batch size.

| | Layer (type) | Output shape | Filters | Kernel size | Parameters |
|---|---|---|---|---|---|
| 1 | Input | (None, 54, 20, 20, 2) | - | - | 0 |
| 2 | ZeroPadding3D | (None, 58, 24, 24, 2) | - | - | 0 |
| 3 | Conv3D_1 | (None, 58, 24, 24, 16) | 16 | $3 \times 3 \times 3$ | 880 |
| 4 | MaxPooling3D_1 | (None, 29, 12, 12, 16) | - | $2 \times 2 \times 2$ | 0 |
| 5 | Conv3D_2 | (None, 29, 12, 12, 32) | 32 | $3 \times 3 \times 3$ | 13,856 |
| 6 | MaxPooling3D_2 | (None, 14, 6, 6, 32) | - | $2 \times 2 \times 2$ | 0 |
| 7 | Conv3D_3 | (None, 14, 6, 6, 64) | 64 | $3 \times 3 \times 3$ | 55,360 |
| 8 | MaxPooling3D_3 | (None, 7, 3, 3, 64) | - | $2 \times 2 \times 2$ | 0 |
| 9 | Conv3D_4 | (None, 7, 3, 3, 128) | 128 | $3 \times 3 \times 3$ | 221,312 |
| 10 | MaxPooling3D_4 | (None, 3, 1, 1, 128) | - | $2 \times 2 \times 2$ | 0 |
| 11 | Flatten | (None, 384) | - | - | 0 |
| 12 | Output (Dense) | (None, 3) | - | - | 1155 |
| | | | | Total: | 292,563 |

Table A.7: Layer structure of the LSTM used to reconstruct the S1 top & bottom data. In total 5,836,435 trainable parameters. *None* does not specify the size of the dimension, allowing a variable batch size. In the LSTM layer, the last output is returned in the output sequence instead of the full sequence.

| | Layer (type) | Output shape | Filters | Kernel size | Parameters |
|---|---|---|---|---|---|
| 1 | Input | (None, 54, 20, 20, 2) | - | - | 0 |
| 2 | ZeroPadding3D | (None, 58, 24, 24, 2) | - | - | 0 |
| 3 | TD(Conv2D_1) | (None, 58, 24, 24, 16) | 16 | $7 \times 7$ | 1584 |
| 4 | MaxPooling3D_1 | (None, 29, 12, 12, 16) | - | $2 \times 2 \times 2$ | 0 |
| 5 | TD(Conv2D_2) | (None, 29, 12, 12, 32) | 32 | $5 \times 5$ | 12,832 |
| 6 | MaxPooling3D_2 | (None, 14, 6, 6, 32) | - | $2 \times 2 \times 2$ | 0 |
| 7 | TD(Conv2D_3) | (None, 14, 6, 6, 64) | 64 | $5 \times 5$ | 51,264 |
| 8 | TD(Flatten) | (None, 14, 2304) | - | - | 0 |
| 9 | LSTM_1 | (None, 512) | - | $2 \times 2 \times 2$ | 5,769,216 |
| 10 | Output (Dense) | (None, 3) | - | - | 1539 |
| | | | | Total: | 5,836,435 |

Table A.8: Layer structure of the CNN-LSTM used to reconstruct the S1 top & bottom data. In total 441,443 trainable parameters. *None* does not specify the size of the dimension, allowing a variable batch size. In the ConvLSTM2D layers, the last output is returned in the full sequence instead of the output sequence.

| | Layer (type) | Output shape | Filters | Kernel size | Parameters |
|---|---|---|---|---|---|
| 1 | Input | (None, 54, 20, 20, 2) | - | - | 0 |
| 2 | ZeroPadding3D | (None, 58, 24, 24, 2) | - | - | 0 |
| 3 | ConvLSTM2D_1 | (None, 58, 24, 24, 8) | 8 | $7 \times 7$ | 15,712 |
| 4 | MaxPooling3D_1 | (None, 29, 12, 12, 8) | - | $2 \times 2 \times 2$ | 0 |
| 5 | ConvLSTM2D_2 | (None, 29, 12, 12, 16) | 16 | $5 \times 5$ | 38,464 |
| 6 | MaxPooling3D_2 | (None, 14, 6, 6, 16) | - | $2 \times 2 \times 2$ | 0 |
| 7 | ConvLSTM2D_3 | (None, 14, 6, 6, 32) | 32 | $5 \times 5$ | 153,728 |
| 8 | MaxPooling3D_3 | (None, 7, 3, 3, 32) | - | $2 \times 2 \times 2$ | 0 |
| 9 | ConvLSTM2D_4 | (None, 7, 3, 3, 64) | 64 | $3 \times 3$ | 221,440 |
| 10 | Flatten | (None, 4032) | - | - | 0 |
| 11 | Output (Dense) | (None, 3) | - | - | 12,099 |
| | | | | Total: | 441,443 |

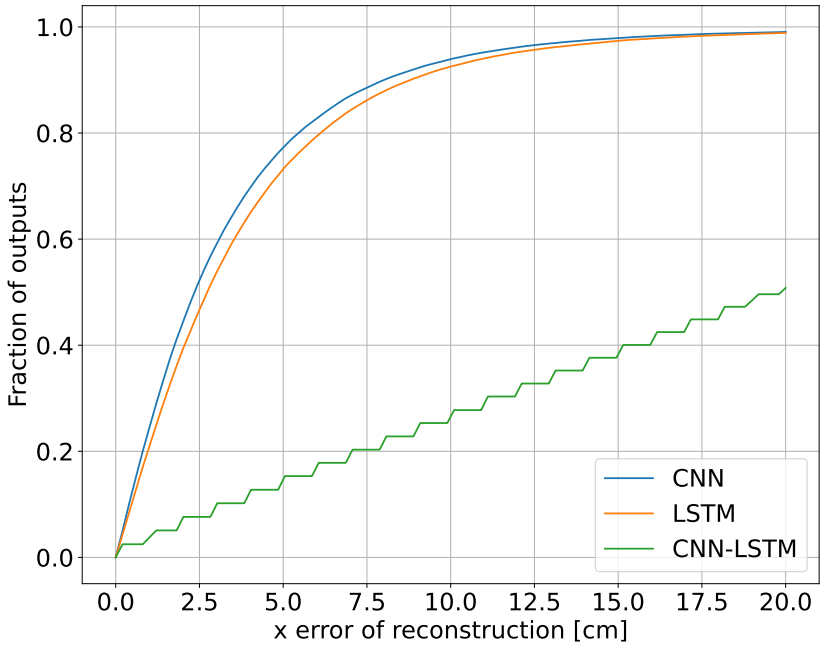


Figure A.5: The cumulative distribution of reconstruction errors in the $x$-direction for DS3 is compared for the different architectures. The steeper the curve, the better the model performs. The CNN model, whose underlying architecture is shown in table A.6, performs best.

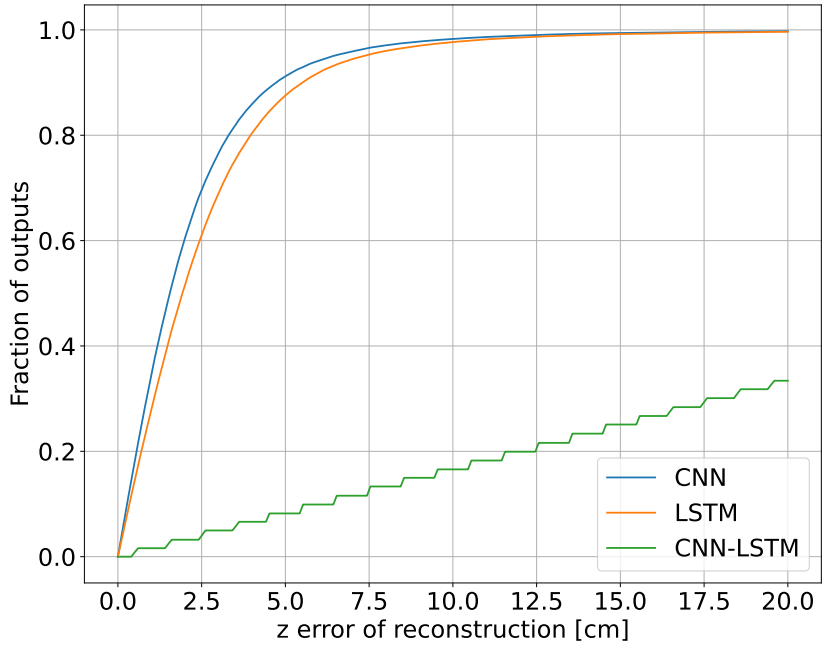


Figure A.6: The cumulative distribution of reconstruction errors in the $z$-direction for DS3 is compared for the different architectures. The steeper the curve, the better the model performs. The CNN model, whose underlying architecture is shown in table A.6, performs best.

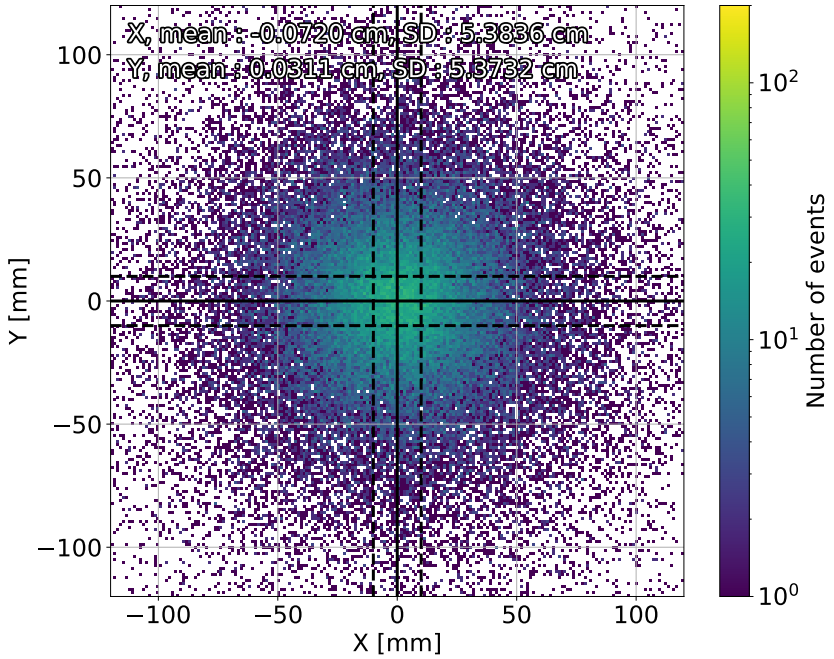


Figure A.7: Distribution of the localization error in $x$- and $y$-direction of the CNN architecture for DS3. The mean error and standard deviation (SD) are given in cm respectively, the $y$-reconstruction has a smaller SD.

## Cumulative Distributions of Reconstruction Errors for all Datasets

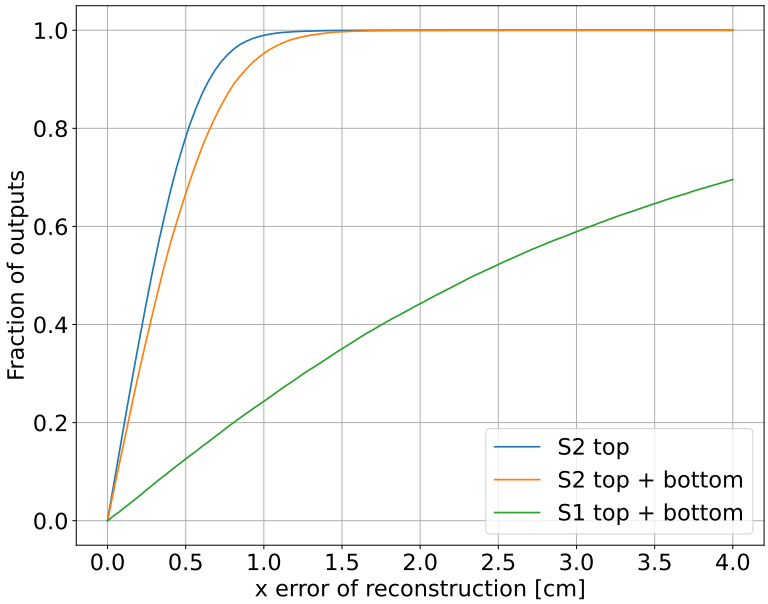


Figure A.8: The cumulative distribution of reconstruction errors in the $x$-direction is compared for the best performing models of the three datasets. The steeper the curve, the better the model performs. DS1 trained with the CNN-LSTM architecture performs best.

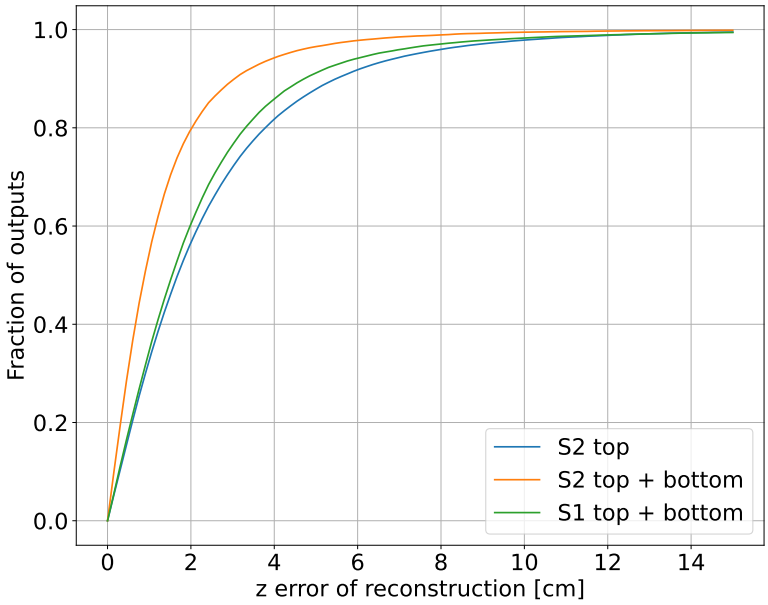


Figure A.9: The cumulative distribution of reconstruction errors in the $z$-direction is compared for the best performing models of the three datasets. The steeper the curve, the better the model performs. DS2 trained with the CNN-LSTM architecture performs best.

# Bibliography

[1] Lutz Althüser. Light Collection Efficiency simulations of the XENON1T experiment and comparison to data. Master's thesis, Westfälische Wilhelms-Universität Münster, 2017. 6

[2] E. Aprile and T. Doke. Liquid Xenon Detectors for Particle Physics and Astrophysics. *Reviews of Modern Physics*, 82(3):2053–2097, Jul 2010. 7

[3] Y. Bengio, Paolo Frasconi, and Patrice Simard. Problem of learning long-term dependencies in recurrent networks. *1993 IEEE International Conference on Neural Networks*, pages 1183 – 1188 vol.3, February 1993. 17

[4] Y. Bengio, Patrice Simard, and Paolo Frasconi. Learning Long-Term Dependencies with Gradient Descent is Difficult. *IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council*, 5:157–66, 02 1994. 17

[5] A. Bideau-Mehu, Y. Guern, R. Abjean, and A. Johannin-Gilles. Measurement of refractive indices of neon, argon, krypton and xenon in the 253.7–140.4 nm wavelength range. Dispersion relations and estimated oscillator strengths of the resonance lines. *Journal of Quantitative Spectroscopy and Radiative Transfer*, 25(5):395–402, 1981. 7

[6] François Chollet. *Deep Learning with Python*. Manning Publications Co., USA, 1st edition, 2018. 12, 14

[7] François Chollet et al. Keras. `https://keras.io`, [Online; last accessed October 3, 2022]. 22

[8] Douglas Clowe, Maruš a Bradač, Anthony H. Gonzalez, Maxim Markevitch, Scott W. Randall, Christine Jones, and Dennis Zaritsky. A Direct Empirical Proof of the Existence of Dark Matter. *The Astrophysical Journal*, 648(2):L109–L113, August 2006. 4

[9] Lucas de Vries. Deep Neural Networks for Position Reconstruction in XENON1T. Master's thesis, University of Amsterdam & Nikhef, 2020. 23

[10] Wolfgang Demtröder. *Experimentalphysik 4: Kern-, Teilchen- und Astrophysik*. Springer Spektrum Berlin, Heidelberg, 5th edition, 2017. 3

[11] Martin Erdmann, Jonas Glombitza, Gregor Kasieczka, and Uwe Klemradt. *Deep Learning for Physics Research*. WORLD SCIENTIFIC, 2021. 12, 17

[12] E. Aprile et al. Projected WIMP Sensitivity of the XENONnT Dark Matter Experiment. *Journal of Cosmology and Astroparticle Physics*, 2020(11):031–031, November 2020. 6, 20, 27

[13] Martín Abadi et al. TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. *CoRR*, 2016. Software available from tensorflow.org. 22

[14] High Performance Computing. `https://confluence.uni-muenster.de/display/HPC`, [Online; last accessed September 23, 2022]. 22

[15] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-term Memory. *Neural computation*, 9:1735–80, December 1997. 17

[16] Emiel Hoogeboom, Jorn W. T. Peters, Taco S. Cohen, and Max Welling. HexaConv. *CoRR*, abs/1803.02108, 2018. 20

[17] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. 2014. 15

[18] Ben Kröse, B. Krose, Patrick van der Smagt, and Patrick Smagt. An introduction to neural networks. *J Comput Sci*, 48, January 1993. 13

[19] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, November 1998. 16

[20] LNGS Overview. `https://www.lngs.infn.it/en/lngs-overview`, [Online; last accessed September 17, 2022]. 8

[21] LSTM layer. `https://keras.io/api/layers/recurrent_layers/lstm/`, [Online; last accessed September 21, 2022]. 17

[22] Aaron G. Manalaysay. Studies of the Scintillation and Ionization Properties of Liquid Xenon for Dark Matter Detection. February 2006. 8

[23] Richard Massey, Thomas Kitching, and Johan Richard. The dark matter of gravitational lensing. *Reports on Progress in Physics*, 73(8):086901, July 2010. 4

[24] M. Mitchell. *Artificial Intelligence: A Guide for Thinking Humans*. Farrar, Straus and Giroux, 2019. 12

[25] ReduceLROnPlateau. `https://keras.io/api/callbacks/reduce_lr_on_plateau/`, [Online; last accessed September 21, 2022]. 25

[26] Sebastian Ruder. An overview of gradient descent optimization algorithms. *CoRR*, abs/1609.04747, 2016. 15

[27] Marc Schumann. Direct Detection of WIMP Dark Matter: Concepts and Status. *Journal of Physics G: Nuclear and Particle Physics*, 46(10):103003, August 2019. 4

[28] Xingjian Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo. Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting. *CoRR*, abs/1506.04214, 2015. 19

[29] C. Sivaram and Venkata Manohara Reddy. A. Dark Matter, Dark Energy and Rotation Curves. 2007. 3

[30] V.N Solovov, V Chepel, M.I Lopes, A Hitachi, R Ferreira Marques, and A.J.P.L Policarpo. Measurement of the Refractive Index and Attenuation Length of Liquid Xenon for its Scintillation Light. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 516(2):462–474, 2004. 7

[31] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15:1929–1958, June 2014. 23

[32] Teresa Marrodá n Undagoitia and Ludwig Rauch. Dark matter direct-detection experiments. *Journal of Physics G: Nuclear and Particle Physics*, 43(1):013001, December 2015. 5

[33] T. S. van Albada, J. N. Bahcall, K. Begeman, and R. Sancisi. Distribution of dark matter in the spiral galaxy NGC 3198. *The Astrophysical Journal*, 295:305–313, August 1985. 3, 4

[34] XENON Dark Matter Project – Direct Search for Dark Matter with Liquid Xenon Deep Underground at the INFN Laboratori Nazionali del Gran Sasso, Italy. `http://xenonexperiment.org/`, [Online; last accessed September 29, 2022]. Credit: XENON Collaboration. 9