# Localization of interaction points on the $\beta$Plast detector with Machine Learning

Facility for Antiproton and Ion Research in Europe and
GSI Helmholtzzentrum fur Schwerionenforschung

ALESSANDRA MONTALBANO

Internship and Training Project Report
Darmstadt, Germany
15-october-2021

GET INvolved 2021: GI-21441M-IT-STS

**Author(s)**
Alessandra Montalbano
University of Palermo, Department of Physics and Chemistry Emilio Segrè
Viale delle Scienze, Ed. 17, Palermo (PA) - Italy
Email: alessandra.montalbano7@gmail.com

**Project Mentor/Supervisor**
Dr. Helena Albers
Nuclear Spectroscopy
GSI Helmholtzzentrum für Schwerionenforschung GmbH,
Planckstr. 1, 64291 Darmstadt, Germany
Tel: +49 6159 71 xxxx
Email: h.albers@gsi.de

**Program Coordinator**
Dr. Pradeep Ghosh
GSI Helmholtzzentrum für Schwerionenforschung GmbH &
Facility for Antiproton and Ion Research in Europe GmbH
Tel: +49 6159 71 3257, Fax: +49 6159 71 3916
Email: Pradeep.Ghosh@fair-center.eu

**GET Involved 21441M-IT-STS**

Publisher: GSI Helmholtzzentrum für Schwerionenforschung GmbH,
Planckstr. 1, 64291 Darmstadt, Germany
Published: September 2021

## Abstract

This project presents the first application of a multi-output ANN to the reconstruction of interaction positions on a fast-timing plastic scintillator detector on an event-by-event basis. The estimated positions on the fast-timing detector are compared with those computed from the coincidence position-sensitive detector and then compared to those obtained using a classical position reconstruction algorithm. The machine learning algorithm achieved a position resolution of $\sim 5\ mm$ with a standard deviation of $\sim 5\ mm$. Other future developments are also presented. The majority of this work was carried out using the Go4 analysis framework (for data unpacking, calibration and preparation) and Python (for the implementation of the machine learning algorithm).

## Declaration

I hereby declare that the project entitled "**Localization of interaction points on the $\beta$Plast detector with Machine Learning**" is my own work and that I have correctly acknowledged the work of others.

## Acknowledgements

During my time at GSI, I have received a great deal of help and support from the whole DESPEC team.

First and foremost, I would like to thank the DESPEC team head Dr. Jürgen Gerl, for his helpful advice on this project, and my project supervisor Dr. Helena Albers, whose expertise was crucial in developing the research questions and methods. Their insightful feedback encouraged me to do better and improve my work.

Furthermore, I want to thank all the DESPEC team members, especially the PhD students, for patiently explaining what I did not understand and advising me with my work.

I would also like to thank the professors of the Department of Physics and Chemistry at the University of Palermo for their advice and assistance in getting things ready before starting the GET_INVOLVED program.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

In this research project a machine learning algorithm determining the interaction positions of incident gamma rays on the $\beta$Plast detector was developed. The $\beta$Plast is part of the setup of DESPEC (DEcay SPECtroscopy) experiments at GSI and the goal is to achieve an improved position resolution compared to classical reconstruction techniques, in order to enhance the detector's usage during beam experiments.

According to existing research on the application of machine learning for position reconstruction, Artificial Neural Network (ANN) models typically outperform all other machine learning techniques in terms of position reconstruction accuracy [1, 2, 3, 4]. The only study dealing with this problem by means a non-parametric machine learning method is [5], where authors implement a k-nearest neighbours regression. In [3], Bruyndonckx *et al.* also give an accurate description of their CNN model, applied on position reconstruction.

All examined related studies have used a direct method of data acquisition and labelling, i.e., by moving the source on the detector and acquiring data, resulting in a training dataset which may potentially present pitfalls due to the sampling method and generally dealing with small datasets ranging from a few hundred to a few thousand data points.

The ANN presented in this work has the purpose of quantitatively determining interaction positions on an event-by-event basis. Data analysis employing classical reconstruction algorithms produced a position sensitivity of $\sim 1$ *cm* for the $\beta$Plast detector. Such a machine learning model would be especially useful for future uses of the $\beta$Plast detector during in-beam DESPEC experiments.

A major issue with some of the literature is that clear explanations of the reasoning behind machine learning model's selection are not always provided, or they sometimes show inconsistent choice of model parameters, methods, and evaluation metrics, which could also lead to overfitting and, eventually, to very positive (but possibly inconsistent) results. Among the most commonly observed mistakes are:

1. Multicollinearity between variables used in the model, which may result in redundancy. The model might still have the same predicting power, but no conclusion on the individual contribution of the two variables can be drawn;

2. Use of complex model which are clearly prone to overfitting and might memorize the training data to a degree;

3. Testing dataset of small size, plus the fact that training and testing data were not randomized which leads to results being unreliable;

4. Lack of clear statements of error values and experimental results, plus absence of error analysis.

It is common to see machine learning methods being employed as a "black-box solution". Nevertheless, it is crucial to note that not carefully considering a model's underlying assumptions may lead to the model performing well on training data, but having little relevance as a prediction tool when presented with unseen and/or noisy data.

An alternative data acquisition method, consisting of two distinct detectors, one position-sensitive and one time sensitive, was implemented, and a $^{22}$Na positron source was used. The two detectors firing simultaneously allow for photon coincidence measurements. This technique allowed to acquire a considerably large dataset in little time.

This project's goal is to consider all relevant aspects of the machine learning problem and, as a result, to construct a model appropriate to this setup while also doing effective feature engineering, then choosing the most appropriate performance metrics.

# Chapter 2

# Experimental setup

Since the algorithm implemented was a supervised machine learning algorithm, data needed to be labeled. The setup consisted of two detectors firing in coincidence. To estimate label positions, a position-sensitive $\gamma$-ray scintillator detector (PSD) comprising a cylindrical lutetium yttrium oxyorthosilicate (LYSO) crystal, with size of $76mm$ diameter and $3mm$ thick, and an active area of $50mm$ diameter was used. It was optically coupled with a position-sensitive photomultiplier tube (PSPMT) based on a cross-wire anode structure consisting of 16 wires in the x-axis and 16 wires in the y-axis, the Hamamatsu R2486 [6]. Individual multianode readout method (IMAR) was used, meaning that all anodes are read out separately. This method provides a more accurate position resolution and it was first used by A.J.Bird *et al.* in [7], but only dealing with a one-dimensional setup. The spatial resolution of the LYSO detector is of around $\sim 3\ mm$ over an active region of diameter $\sim 50mm$, worsening near the edges. To raise the efficiency of the detector it is wrapped with enhanced specular reflector (ESR) film to reduce the probability that photons escape.

The second detector is the $\beta$Plast, a time sensitive detector, which comprises a sheet of BC400 plastic scintillator plastic, which provides excellent timing resolution, for a total active region of size $80\ mm \times 80\ mm$ and a $3\ mm$ depth, read out by 16 silicon photomultipliers (SiPMs) [1], which are a multipixel semiconductor photodiodes with each pixel being an Avalanche photodiode (APD), situated along the edges. The $\beta$Plast detector was mounted in a light-tight environment to prevent ambient light from interacting with the scintillation material and SiPMs.

The scanner system is configured in such a way that a $^{22}$Na positron source is positioned in the middle of the LYSO and $\beta$Plast detectors. When the radioactive source $^{22}$Na decays, two $\gamma$-ray photons are emitted back-to-

[1] in the analysis we use only 14 SiPMs because two channels were not working.

back, each one with a 511 keV energy. On the $\beta$Plast, when an ionizing radiation hits the scintillator, the scintillating material can emit light in three different ways of interaction: photoelectric absorption, Compton scattering and pair production. The incident energy of photons is absorbed by electrons in the valence band and get elevated to the conduction band. When the electron de-excites it goes back to the valence band and photons are emitted in the visible range, the photons are then collected by the SiPMs situated along the sides of the $\beta$Plast and converted to an electrical signal [8].

Only events where both the LYSO and $\beta$Plast detectors fire in coincidence were recorded in order to minimize the background event rate., in this way the background event rate in the LYSO detector should be minimal, because the time coincidence between the two detectors ensures that only events that are seen on $\beta$Plast are also seen on LYSO. Both the source and the $\beta$Plast detectors are placed at an optimal distance in order to have a maximum solid angle. Here is illustrated the geometry of the scanner setup, the cone represents how events that are seen on the $\beta$Plast detector active area are projected on the LYSO detector:

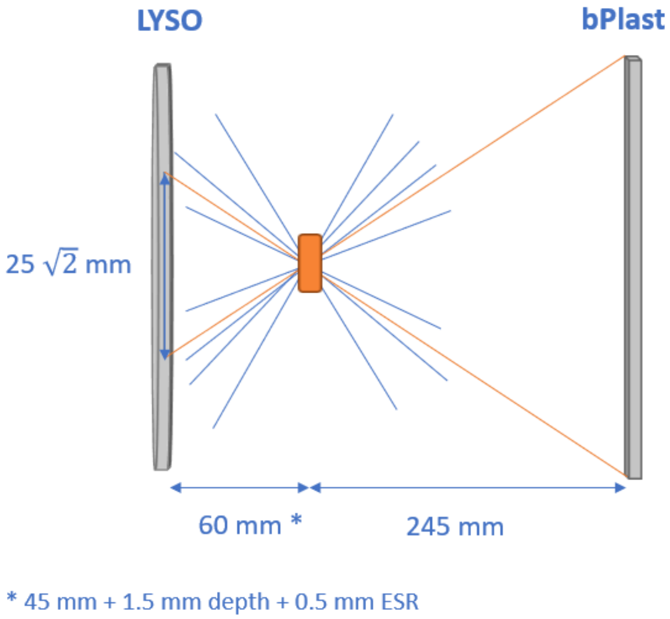

Figure 2.1: Schematic view of the geometry of the scanner set up: seen from the side.

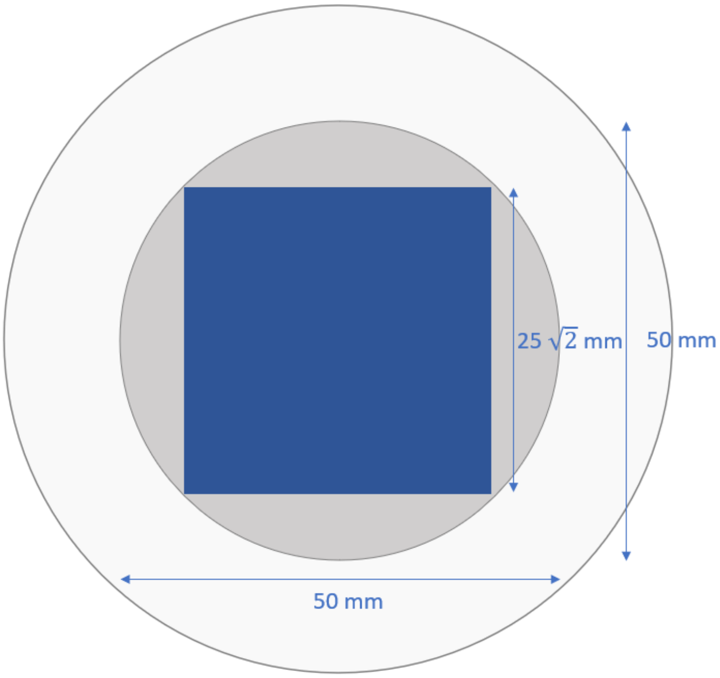

Figure 2.2: Schematic view of the geometry of the scanner set up: $\beta$Plast projected on the LYSO's active region, seen from the front.

During measurements, a dataset of pulse shapes was recorded for all trajectories coming inside the coincidence cone of the LYSO and $\beta$Plast detectors, by extracting a timing signal from each of the two detector which is then processed by the electronics to ensure coincidence.

## 2.1 Spatial resolution and position calibration

The accuracy achieved by the machine learning algorithm depends on the precision with which events positions' are determined on the LYSO detector. In order to get accurate labels and to correctly estimate the error given by the LYSO detector (i.e., LYSO detectors sensitivity), from [9] we can estimate the resolution of the LYSO detector to be around $\sim 1\ mm$ in the center of the detector. The resolution achieved at the edges of the LYSO detector is expected to be lower because of different effects such as light reflections and because of the number and distance of photocathodes.

# Chapter 3

# Data collection

Data has been acquired by means of the Multi Branch System (MBS), that is the standard Data Acquisition System at GSI [10]. Unpacking of data, pulse height spectra and QDC spectra analysis was done by means of the GSI Object Oriented Online Offline System (Go4) [11], which is based on the ROOT data analysis framework developed by CERN [12].
The electronics setup is composed of two Versa Module Europa VME Electronics (for both $\beta$Plast and the LYSO detector).

The dynode and anode signals from the LYSO were both amplified and fed each into an eight-channel discriminator. Then the two outputs were fed into a coincidence unit. An “AND” gate is used for coincidence, where the trigger detector is the $\beta$Plast detector because of the self-activity of the LYSO detector, and the two detectors are connected via a trigger bus.

The signals from the $\beta$Plast's SiPMs and the signals from the LYSO anodes were amplified using fast amplifier boards and then a charge to digital converter (QDC) determines an event's charge by integrating the incoming voltage signal for LYSO.

The gains of the individual anodes of the PMT, are not all equal and this leads to image distortion and low position resolution for the LYSO detector, so the gain of each individual anode was calibrated by means of the individual multianode readout method (IMAR) [13].

The QDC pedestal is the value readout that is measured when there is no input signal, and the input is open. It is mostly attributable to a basic current that is required for the QDC to function properly and is also integrated. The pedestals were therefore subtracted and the resulting charge spectra rescaled such that the high-energy edges of the individual anodes were aligned.

## 3.1 Hit position localization data

Interaction positions were obtained from the LYSO detector. For the LYSO detector the total energy spectrum in LYSO comprises 2,350,383 events.
To restrict the search, the QDC pulse height spectra was summed over all the 32 channels and a gate was placed on the 511 keV peak. This was done to prevent the inclusion of background noise into the dataset, as only at the 511 peak photons are emitted back-to-back. With this gate, there are 853,642 events on the LYSO detector. By further restricting this search on the $\beta$Plast projection on the LYSO (shown in Figure 2.2) there are then 732,568 events.

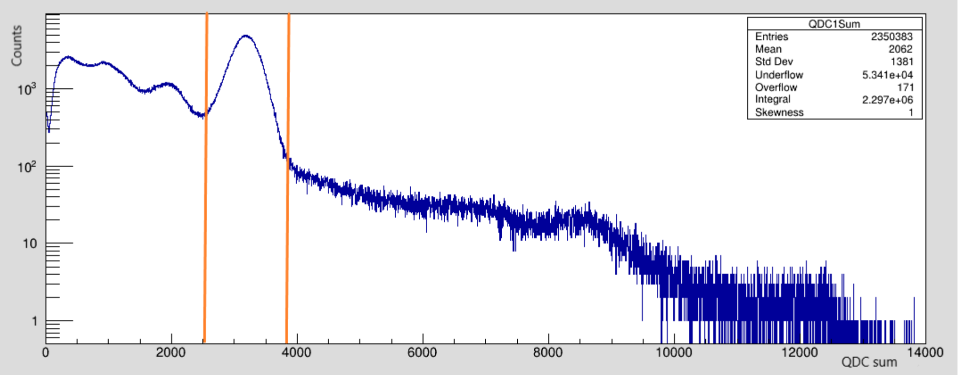

Figure 3.1: Sum of the total QDC across X and Y. The 511 keV peak is clearly noticeable.

The LYSO positions were then computed on an event-by-event basis, by fitting a Gaussian distribution to the measured charge profile. The peak of the fitted Gaussian gives an estimate of the $(x, y)$ coordinates of the $\gamma$-ray interaction, considering the fact that reconstruction on the edges has lower precision [9, 14].

The final result of this procedure is a combined dataset containing the pulse shapes from the SiPMs of the $\beta$Plast detector and the $(x, y, z)$ coordinates from LYSO, with $z$-coordinate fixed to 1.5 $mm$ depth. LYSO positions were then converted to positions on the $\beta$Plast detector using trigonometric formulas.
Here is the geometry of the $^{22}$Na-source and the two detectors:

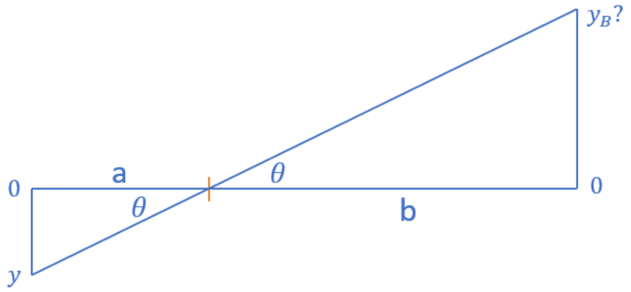

Figure 3.2: Trigonometry of the problem for the y-coordinate.

Assuming that the source is positioned at the point $(0, 0)$, the surface of the LYSO has area $S_A$ and an interaction position on the LYSO detector was defined by coordinates $(x, y)$. Moreover, the surface of the $\beta$Plast detector has area $S_B$ and the simultaneous second $\gamma$-ray's (which is travelling in the opposite direction) interaction position on the $\beta$Plast detector with coordinates $(x_B, y_B)$ was defined. Then, as in the graph above, the following distances were set: $a$ as the distance between the source and the LYSO detector and $b$ as the distance between the $\beta$Plast detector and the source, while $\theta$ is the angle between the particle trajectory and the x-axis.

Considering the setup described above, the following two equations hold:

$$\begin{cases} tg\theta = \frac{y}{a} \\ tg\theta = \frac{-y_B}{b} \end{cases} \tag{3.1}$$

and then the $y_B$ position can be estimated as

$$y_B = -b\frac{y}{a} \tag{3.2}$$

and similarly for the x coordinate.

In the end a formula was employed for “translating” an interaction position on the LYSO detector, with coordinates $(x, y)$ to an interaction position on the $\beta$Plast detector in coordinates $(x_B, y_B)$.

The formula is as follows:

$$(x_B, y_B) = (-b\frac{x}{a}, -b\frac{y}{a}) \tag{3.3}$$

In the examined case, these values were set as a = 65 $mm$ and b = 265 $mm$ as it is shown in Figure 4.2.

## 3.2 Dataset pre-processing and partitioning

| **Variable** | **Description** |
|---|---|
| **Event** | The event number |
| **Position x** | x coordinate of the event of the $\beta$Plast detector |
| **Position y** | y coordinate of the event of the $\beta$Plast detector |
| **QDC $\beta$Plast (1-14)** | A measure of charge of the event |

Table 3.1: Description of the used dataset

In the beginning the original dataset was shuffled to make sure that the ANN can learn in a more robust way, and then it was divided into three parts in order to obtain: a training set, a validation set and a test set, with proportions 70/20/10.
The dataset on which the model is trained is referred to as the training set, while the validation set is the dataset the model has not been trained with and is used during training to optimize hyperparameters. The test set is only utilized at the end to make predictions, once the model has been tuned.

When training an ANN, it is important to pre-process data in best way possible depending on the dataset's characteristics and features. The datasets for the algorithm training, testing and validation of the ANN were pre-processed in the following way:

1. Standardization can be useful to eliminate the influence of one channel over another, and will influence the objective function to have a more concentric circle contour, making training easier. Reasons for standardization:

    - Since the channels are measured on different scales and the loss function used in the machine learning developed for this project is scale-sensitive loss function, this might cause the contribution of one channel to the distance to be much bigger than other columns. So, it is important to rescale the channels so that their variability reflects their importance, in this specific case all the channels should have equal importance, so they were rescaled on the same range. However, in the case of an ANN, as inputs are combined linearly this might not be strictly necessary, but during hyperparameter and parameter tuning this procedure showed to improve the outcome of the machine learning model.
    - It should make the training of the algorithm faster and might reduce the chances of getting stuck in local optima when using gradient descent methods.

- It should avoid saturation in the output activation function, as the model is going to have a bounded function between -1 and 1 on the output layer.

For the considered ANN all features are going to be rescaled from -1 to 1, meaning that every datapoint $x_{ij}$ belonging to the $i^{th}$ row and the $j^{th}$ column will be rescaled normalization such as:

$$x_{ij,stand} = \frac{x_{ij} - x_{min,j}}{x_{max,j} - x_{min,j}} \tag{3.4}$$

this is the case because the distributions of charges for each channel are skewed. On the other hand, the label columns do not need to be applied this procedure, it is still useful to get them rescaled in a [-1,1] range and this is done according to the following formula:

$$x_{ij,stand} = \frac{x_{ij}}{x_{max,j}} \tag{3.5}$$

2. After that, a kind of "normalization" per row is applied, which means dividing by the norm of the entire row, and in this specific case, to make the Euclidean length of the row equal to one. The reason for doing this is because in this case a small part of events has really small or really big values, transforming them means that they would have equal "weight" on the algorithm by making sure that the sum over all channels is exactly one. So $x_{ij,stand}$ belonging to the $i^{th}$ row and the $j^{th}$ column will be rescaled as:

$$x_{ij,stand,final} = \frac{x_{ij,stand}}{\sum\limits_{j} x_{i,j}} \tag{3.6}$$

3. Missing data is handled as follows: if any channel from the sides or up and down SiPMs are missing, the event is discarded. In theory, it could be also assumed that a datapoint is missing because a SiPM did not fire when the event occurred too far away from it.

4. A workaround was used since the output range is bounded: the labels were scaled in the range [-1,1], in this way they can be treated as "probabilities" and an hard sigmoid activation function can be used on the output layer. To scale them, the used formula is the same as in Equation (3.5).

# Chapter 4

# Multitarget Artificial Neural Network Regression Model

From a mathematical perspective, the position reconstruction problem, since it involves the prediction of continuous variables, i.e., coordinates x and y, can be modelled as a regression problem.

The Python programming language and the TensorFlow machine learning package [15] were used to develop the machine learning algorithm. TensorFlow is a low-level library based on the concept of a computational graph, and its individual units, which are called tensors, are essentially the so-called neurons in machine learning models.

When a $\gamma$-ray hits $\beta$Plast detector, each SiPM sees a signal with an amplitude and delay which give info about the hit position: closer SiPMs will see a larger signal.

The chosen machine learning algorithm was a Multitarget ANN regression model, which is a specific type of ANN for non-linear regression models, allowing to predict a continuous vector output.

## 4.1 Model definition

The training set $D$, made of $N$ events: $D = \{(C^{(i)}, P^{(i)})\}$ with $i = 1, ..., N$, where $P$ and $C$ are two matrices of sizes, respectively, $N \times 2$ and $N \times 14$. The task is to learn a multitarget regression model from the training dataset $D$ consisting weights $W_i$ for the matrix $C^{(i)}$ a label $P^{(i)}$.

A Feedforward ANN consisting of three layers (input layer, one hidden layer and output layer) was implemented. The input layer is composed of 14 input nodes. To each node different weights are associated and all neurons in adjacent layers are fully connected to one another, and each neuron in the hidden and output layer has a weight associated with it.

The following is the ANN's architecture:

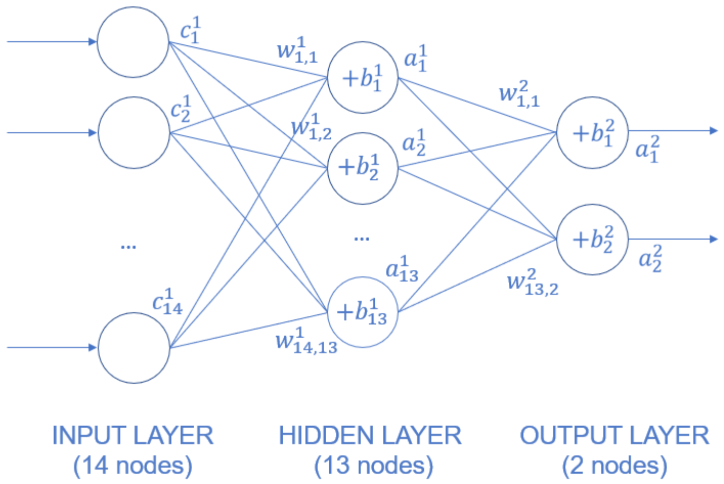

Figure 4.1: ANN architecture

In a ANN, each non-input node aggregates values that are fed into it by the previous layer's nodes and produces a single value to which a bias is added, the result, called net input, is then fed into the nodes of the subsequent layer. Then the nodes of the hidden and output layers convert their net input using an activation function, which determines the node's output.

It turned out that a elu activation function [16] for the hidden layer was the most effective to solve this problem. Furthermore, since the goal is to predict continuous values in a bounded range, a hard sigmoid activation function was used on the output layer, and since the output of the model will be in the x-range $[-40, 40]$ and y-range $[-40, 40]$, the final output was reshaped to get values in those ranges in the final step.

The following is the output of the first hidden layer of the ANN in Figure 4.1:

$$elu(\underset{1\times 14}{C} \cdot \underset{14\times 13}{W^{(1)}} + \underset{1\times 13}{B^{(1)}}) \tag{4.1}$$

**Problems with tanh activation function:**

At first glance, this appears to be a viable method because the machine learning algorithms always provide outputs in the [-1,1] range, which can later be rescaled.

However, it did not work as expected as the rescaled positions are in the range $[-1, 1]$, but the tanh function $\in (-1, 1)$, it suffers from vanishing gradient problems and almost never reaches those boundary values. Many alternatives were tried, but results were not satisfactory yet.
Consequently, by taking inspiration from Bounded Leaky Relu [17] and Hard Sigmoid [18] activation functions, the most convenient solution seemed to build a custom bounded linear activation function whose properties and performance could be further assessed. Here is the function:

$$f(x) = \begin{cases} -0.001x - c & \text{if } x \leq -1 \\ x & \text{if } -1 < x \leq A \\ 0.001x + c & \text{if } x > A \end{cases} \tag{4.2}$$

where A = 1 and c = 0.999 A. Using this function eliminates the problem of the vanishing gradient and it's less computationally expensive.

One of the best known and most effective methods for training an ANN is the so-called error Backpropagation (BP) algorithm [19], which systematically updates the weights of the connections between the nodes, so that the output of the network gets closer and closer to the label.

To speed up the training batch normalization [20] is implemented in the ANN. When the input distribution changes during the training, it experiences covariate shifts [21], to avoid this problem a good practice is to normalize each training mini-batch, before it enters the activation function of each hidden layer. In order to further speed up learning but mainly to prevent overfitting, a stochastic regularization technique called dropout [22] is also used. Basically, percentage of the units is dropped out from the network (those units are chosen at random), together with all its connections to the previous and next layers. And this happens for each mini batch that is being trained. The optimal amount of dropped out units turned out to be 10%.

## 4.2 Hyperparameter selection

Hyperparameters are all the training variables that must be manually set to a predetermined value before the training begins. The choice of hyperparameters is critical to the model's predicting performance. The number of epochs, learning rate, batch size, and number of nodes in each layer are some of the most important hyperparameters.

The learning rate is very important because it determines the step size of the learning procedure: if it is too small the model does not converge (or it does very slowly making the model impossible to train), while if it is too big it can cause the model to overfit. The best learning rate for this ANN was determined to be 0.147.

To determine the number of layers, as well as other hyperparameters, Bergstra and Bengio's concluded that grid search is inefficient [23], and so a search in the hyperparameter space using a randomized search was performed, while also considering the minimization of running time of the machine learning algorithm. Simpler models were therefore preferred to more complex ones.

According to Glorot and Bengio [24], the biases should be initialized to 0 while the weights $w_{ij}$ should be initialized at each layer according to the following uniform distribution:

$$w_{ij} \sim U[-\frac{1}{\sqrt{N}}, \frac{1}{\sqrt{N}}] \tag{4.3}$$

where $N$ is the size of the previous layer. This initialization gave better results than the more commonly used initialization to random values distributed according to a Gaussian distribution.

## 4.3 Learning algorithm

The used learning algorithm was adaptive moment estimation (Adam), a learning rule that includes momentum, known to speed up learning and to help not getting stuck in local minima (because SGD always goes straight downhill). According to Kingma *et al.*, [25] the Adam learning algorithm "is straightforward to implement, is computationally efficient, has little memory requirements, is invariant to diagonal rescaling of the gradients, and is well suited for problems that are large in terms of data and/or parameters".

It essentially combines classical momentum, in that it computes a decaying mean (with decay constant $\beta_1$) of the past gradient into a momentum vector $m$ and uses that instead of the actual gradient, together with RMSProp, computing an exponentially decaying mean of the past gradient with decay costant $\beta_2$.
This enables both faster learning along dimensions where gradients remain stable throughout training steps and slower learning across turbulent dimensions where the gradient fluctuates, and the same time it has the RMSProp characteristics of facilitating training in unbalanced datasets. This is the algorithm:

$$\begin{cases} g_t = \nabla_{\theta_{t-1}} f(\theta_{t-1}) \\ m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t \\ \hat{m}_t = \frac{m_t}{1-\beta_1^t} \\ n_t = \beta_2 n_{t-1} + (1-\beta_2) g_t^2 \\ \hat{m}_t = \frac{n_t}{1-\beta_2^t} \\ \theta_t = \theta_{t-1} - \beta_2 \frac{\hat{m}_t}{\sqrt{\hat{n}_t} + \epsilon} \end{cases} \tag{4.4}$$

## 4.4 Loss function

A loss function must be defined with aim to minimize it so that the supervised model will learn from the training data.

The goal is to find an estimate $\hat{P}$ of the labels $P$ that minimizes the distance of the distributions of real and predicted positions. The MSE loss function was used:

$$L(P, \hat{P}) = \frac{1}{N}[(P - \hat{P})^T (P - \hat{P})] \tag{4.5}$$

where $\hat{P}$ is a $1 \times 2$ matrix containing the estimate for the label $P$, which is also a $1 \times 2$ matrix. Then the ANN is trained to minimize the average MSE value over the training set.

Since training dataset was quite large size, network was trained by means of mini batches, as it is widely known that training a model with larger batches causes a significant degradation of the model [26]. The batch size defines the number of samples that will be propagated through the network. Since in the considered dataset there are 422556 training samples, batch size was setted up to 422, for a total of 1000 steps. This is not excessively small, as it's important to take into consideration also the training time of the algorithm. Using this method, the algorithm at each step takes one batch and trains the network each time, this procedure is done until all samples have been propagated through of the network.

Regarding the number of epochs, by random search it turned out that training the machine learning algorithm for 200 epochs was enough, the algorithm was trained to convergence and then the weights with the lowest validation error were selected. This procedure, called "early stopping" or "stopped training" [27], makes sense when assuming that the validation error is a good estimate of the generalization error.

## 4.5 Performance

To estimate the resolution of the machine learning method both systematic errors (due to the experimental setup) and statistical errors were considered.

The precision estimation of the ML algorithm is computed as the mean absolute error, which is computed as the difference between real and observed position:

$$\bar{\delta}_{ML} = \frac{|P - \hat{P}|}{N} \tag{4.6}$$

The final position resolution would then be computed as the sum in quadrature of the precision resolution of the ML algorithm and the position resolutions of the labels, that is the LYSO labels, being them independent:

$$\delta_{tot} = \sqrt{\bar{\delta}^2_{ML} + \delta^2_{LYSO}} \tag{4.7}$$

The estimated position resolution of the LYSO detector is estimated to be of $\sim 1\ mm$.

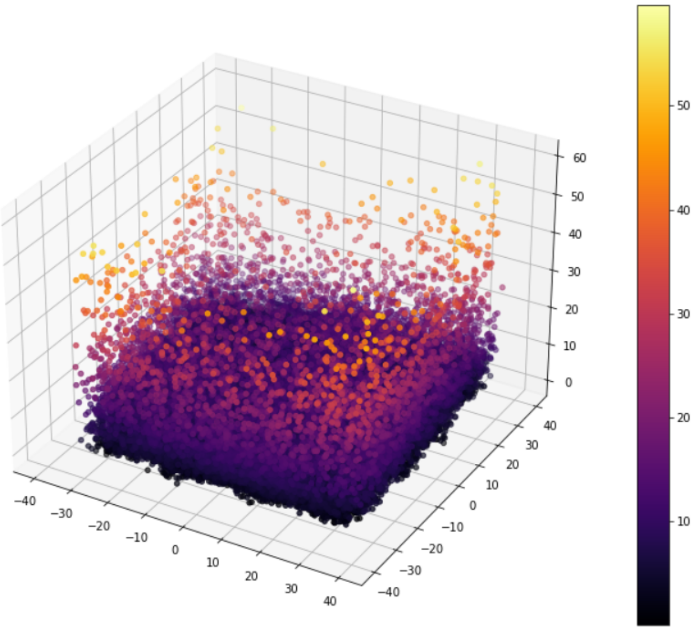

Figure 4.2: 3D histogram of the errors given position.

The average position resolution achieved by the ANN of 5.7 $mm$ on the whole surface of the detector and of 4.9 $mm$ in the central region ($-20 \leq x \leq 20, -20 \leq y \leq 20$), with standard deviation of, respectively, 6.4 $mm$ and 4.6 $mm$.

# Chapter 5

# Conclusion

The ANN achieved an average position resolution of 5.7 $mm$ on the whole detector surface and 4.9 $mm$ in the central region ($-20 \leq x \leq 20, -20 \leq y \leq 20$), with standard deviations of 6.4 $mm$ and 4.6 $mm$, respectively. The $\beta$Plast detector's resolution was definitely improved, considering that classical algorithms such as the "Anger Method" provides a position resolution of $\sim 1$ $cm$, but there is still space for improvement.
Future developments:

1. Training data used in this project was affected by background noise. Good events are typically accompanied by background events. Noise can originate from a range of sources, among which SiPMs failures, and so on. This problem might be addressed in future projects, for example, by attempting to cluster various types of events (good events vs noise) using unsupervised machine learning techniques. These complex perturbations have an impact on the ML model training since they might reduce the quantity (and quality) of "knowledge" that can be extracted from the training dataset.

   Most importantly, there would be two kinds of noise in the data: noise from the labels (i.e., noise in the LYSO) and noise from the features (i.e., noise in channels).

   However, given the nature of this research project and the fact that the machine learning algorithm developed for the $\beta$Plast detector will need to be retrained eventually when employed for the beam experiments, it may be too much effort for the time being.

2. Pulse shape discrimination: having the shape of the charge spectra for each event would be useful to distinguish between different types of incident particles (i.e. $\alpha$, $\beta$, $\gamma$, ion).

# Bibliography

[1] R. Gholipour Peyvandi and S.z. Islami Rad. Precise prediction of radiation interaction position in plastic rod scintillators using a fast and simple technique: Artificial neural network. *Nuclear Engineering and Technology*, 50(7):1154 – 1159, 2018.

[2] Valerio Tabacchini, Giacomo Borghi, and Dennis R Schaart. Time-based position estimation in monolithic scintillator detectors. *Physics in Medicine and Biology*, 60(14):5513 – 5525, 2015.

[3] P. Bruyndonckx, S. Leonard, S. Tavernier, C. Lemaitre, O. Devroede, Yibao Wu, and M. Krieguer. Neural network-based position estimators for pet detectors using monolithic lso blocks. *IEEE Transactions on Nuclear Science*, 51(5):2520 – 2525, 2004.

[4] S. Delorme, R. Frei, C. Joseph, J.-F. Loude, and C. Morel. Use of a neural network to exploit light division in a triangular scintillating crystal. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 373(1):111 – 118, 1996.

[5] H. T. Van Dam, S. Seifert, R. Vinke, P. Dendooven, H. Lohner, F. J. Beekman, and D. R. Schaart. Improved nearest neighbor methods for gamma photon interaction position determination in monolithic scintillator pet detectors. *IEEE Transactions on Nuclear Science*, 58(5):2139 – 2147, 2011.

[6] HAMAMATSU. Position-sensitive photomultiplier tubes with crossed wire anodes, r2486 series.

[7] A.J. Bird, Z. He, and D. Ramsden. Multi-channel readout of crossed-wire anode photomultipliers. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 348(2):668–672, 1994.

[8] Soo Hyun Byun. Med phys 4r06/6r03 radioisotopes and radiation methodology, 2018.

[9] C. Domingo-Pardo, N. Goel, T. Engert, J. Gerl, M. Isaka, I. Kojouharov, and H. Schaffner. A position sensitive $\gamma$-ray scintillator detector with enhanced spatial resolution, linearity, and field of view. *IEEE Transactions on Medical Imaging*, 28(12):2007 – 2014, 2009.

[10] S.Augustin D.Bertini H.G.Essel S.Linev J.Adamczewski-Musch, M.Al-Turany. The go4 analysis framework reference manual v6.1. *GSI Helmholtzzentrum für Schwerionenforschung GmbH*, May 2021.

[11] S.Linev J.Adamczewski-Musch, N.Kurz. Gsi data acquisition system mbs release notes v6.3. *GSI Helmholtzzentrum fÃ¼r Schwerionenforschung GmbH*, November 2017.

[12] Rene Brun and Fons Rademakers. Root – an object oriented data analysis framework. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 389(1):81–86, 1997. New Computing Techniques in Physics Research V.

[13] N. Goel, C. Domingo-Pardo, T. Engert, J. Gerl, I. Kojouharov, and H. Schaffner. Spatial calibration via imaging techniques of a novel scanning system for the pulse shape characterisation of position sensitive hpge detectors. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 652:591–594, 2011.

[14] G. Charpak and F. Sauli. High-accuracy, two-dimensional read-out in multiwire proportional chambers. *Nuclear Instruments and Methods*, 113(3):381–385, 1973.

[15] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[16] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus), 2016.

[17] Shan Sung Liew, Mohamed Khalil-Hani, and Rabia Bakhteri. Bounded activation functions for enhanced training stability of deep neural networks on visual pattern recognition problems. *Neurocomputing*, 216:718–734, 2016.

[18] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. Binaryconnect: Training deep neural networks with binary weights during propagations. *CoRR*, abs/1511.00363, 2015.

[19] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.

[20] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015.

[21] Hidetoshi Shimodaira. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of Statistical Planning and Inference*, 90(2):227–244, October 2000.

[22] 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(56):1929–1958, 2014.

[23] J. Bergstra and Y. Bangio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13:281 – 305, 2012.

[24] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In Yee Whye Teh and D. Mike Titterington, editors, *AISTATS*, volume 9 of *JMLR Proceedings*, pages 249–256. JMLR.org, 2010.

[25] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[26] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. *CoRR*, abs/1609.04836, 2016.

[27] Warren S. Sarle. Stopped training and other remedies for overfitting. In *Proceedings of the 27th Symposium on the Interface of Computing Science and Statistics*, pages 352–360, 1995.