UNIVERSITAT POLITÈCNICA DE VALÈNCIA

School of Informatics

Development of Deep Learning models to support Proton Range Verification in Hadron Therapy

End of Degree Project

Bachelor's Degree in Informatics Engineering

AUTHOR: Cano Caravaca, Vicent

Tutor: Gómez Adrian, Jon Ander

External cotutor: LLOSA LLACER, GABRIELA

ACADEMIC YEAR: 2021/2022

Agraïments

En primer lloc vull agrair als meus pares. Ells no només han fet possible econòmicament la meua formació universitària, sinó que també m'han animat sempre a estudiar i han estat disposats a fer el que calguera perquè ho fera.

També he d'agrair al meu tutor Jon Ander Gómez Adrián, qui m'ha orientat en la realització d'aquest TFG. De la mateixa manera, vull agrair a la meua altra tutora, Gabriela Llosá Llácer, i a Marina Borja Lloret. Elles han fet possible la col·laboració amb l'IRIS, el seu grup d'investigació, proveint les dades utilitzades en aquest treball i orientant-me en el camp de la teràpia hadrònica.

Finalment m'agradaria agrair a tots els amics de la universitat, ja que ells, amb la seua ajuda i la felicitat que m'han aportat, han creat durant quatre anys l'entorn perfecte per donar el millor de mi en la universitat. També vull agrair a la meua amiga Sandra, la qual m'ha donat motivació i inspiració durant aquests últims anys.

Resum

El tractament del càncer és un camp en constant evolució, en el qual cada any es proposen i investiguen nous mètodes experimentals. La teràpia Hadrònica és un dels mètodes de tractament del càncer més prometedors que s'estan aplicant i millorant actualment. Encara que va ser proposat per primera vegada en 1946, la tecnologia de l'època va limitar els seus usos significativament. En els últims anys, la teràpia hadrònica està guanyant molta popularitat, amb un increment d'instal·lacions i investigacions científiques centrades en aquest mètode.

Els beneficis de la teràpia hadrònica són destacables, ja que ofereix la possibilitat de tractar tumors de manera més focalitzada, és a dir, tractant de forma més eficient el tumor i fent menys mal als teixits del voltant. No obstant això, aquesta focalització fa que s'hagen d'aplicar marges de seguretat, limitant el potencial de la teràpia hadrònica. Per reduir aquests marges, s'estan investigant tècniques de monitoratge. MACACO II, és una càmera Compton multicapa que està sent desenvolupada al IFIC (València) per dur a terme aquest monitoratge.

Aquest treball explora les possibilitats del Machine Learning per seleccionar els esdeveniments captats per la càmera Compton, amb l'objectiu d'incrementar la proporció de senyal contra soroll abans de la reconstrucció d'imatge.

Paraules clau: teràpia hadrònica, tractament del càncer, xarxes neuronals, deep learning

Resumen

El tratamiento del cáncer es un campo en constante evolución, en el cual cada año se proponen e investigan nuevos métodos experimentales. La terapia Hádrónica es uno de los métodos de tratamiento del cáncer más prometedores que se están aplicando y mejorando actualmente. Aunque fue propuesto por primera vez en 1946, la tecnología de la época limito sus usos significativamente. En los últimos años, la terapia hadrónica está ganando mucha popularidad, con un incremento de instalaciones e investigaciones científicas centradas en este método.

Los beneficios de la terapia hadrónica son destacables, ya que ofrece la posibilidad de tratar tumores de forma más focalizada, es decir, tratando de forma más eficiente el tumor y dañando menos los tejidos de alrededor. No obstante, esta focalización hace necesaria la aplicación de márgenes de seguridad, los cuales limitan el potencial de la terapia hadrónica. Para reducir estos márgenes, se están investigando técnicas de monitorización. MACACO II, es una cámara Compton multicapa que está siendo desarrollada en el IFIC (Valencia) para llevar a cabo esta monitorización.

Este trabajo explora las posibilidades del Machine Learning para seleccionar los eventos captados por la cámara Compton, con el propósito de incrementar el ratio de señal contra ruido antes de la reconstrucción de imagen.

Palabras clave: terapia hadrónica, tratamiento del cancer, redes neuronales, deep learning

Abstract

Cancer treatment is a field in constant development, with new experimental methods being proposed and researched every year. Hadron therapy is one of the most promising cancer treatment methods currently being applied and improved. Despite being first proposed in 1946, the technology of the moment limited its uses significantly. These last years hadron therapy has been gaining much momentum, with an increase of facilities and scientific research focused on this method.

The benefits of hadron therapy are remarkable since it offers the possibility to treat tumors in a more focalized way, that is, damaging less the surrounding tissues and treating more efficiently the tumor. However, this focus makes necessary the application of safety margins, limiting the applicability of hadron therapy. In order to reduce these margins, monitoring techniques are being researched. MACACO II, a multi-layer Compton Camera, is being developed at IFIC (Valencia) for this purpose.

This work explores the possibilities of Machine Learning to select the events captured by the Compton Camera to increase the signal-to-background ratio before image reconstruction.

Key words: hadron therapy, cancer treatment, neural networks, deep learning

Contents

List of Figures

List of Tables

Acronyms

API Application Programming Interface. 13, 15

CPU Central Processing Unit. 13, 14

CSIC Consejo Superior de Investigaciones Científicas. 58

CUDA Compute Unified Device Architecture. 15

GPU Graphics Processing Unit. 13–15

IARC International Agency for Research on Cancer. 1, 57

IFIC Instituto de Física Corpuscular. v, vi, 4, 5, 9, 11, 17, 19, 41, 42, 58

IRIS Image Reconstruction, Instrumentation and Simulations for medical imaging applications. 4, 5, 7, 11, 18, 35, 36, 38, 58

MACACO Medical Applications Compact Compton Camera. v, vi, ix, x, 4, 5, 7, 9, 11, 21, 23–27, 35, 36, 38, 41, 42

ML Machine Learning. 14, 15

NN Neural Network. ix, 31, 38, 47–50

PET Positron Emission Tomography. 3, 4

PG Prompt Gamma-rays. ix, 4

PMMA Polymethyl methacrylate. ix, 17, 18

RT Ray Tracing. 15

SBDT Stochastic gradient boosted distributed decision trees. 8, 9

SDG Sustainable Development Goals. 6

SOBP Spread-out Bragg Peak. 3

TPU Tensor Processing Unit. 13

VRAM Video Random Access Memory. 15

CHAPTER 1
Introduction

Cancer is one of the leading causes of death worldwide. According to the International Agency for Research on Cancer (IARC) of the World Health Organization, in 2019, more than ten million men and nine million women were diagnosed with cancer, totaling 19,292,789 new cases of cancer. At the same time that year, nearly ten million people died of cancer. Furthermore, these statistics are expected not to improve in the future but to worsen with demographic changes, with a prediction by IARC of 30.2 million new cases in 2040.[1]

This has motivated the scientific community to research and develop cancer treatment, constantly proposing new methods and improving the current ones.

One of the most remarkable new cancer treatment methods is the hadron therapy. This method is not really that new. It was first proposed by Robert R. Wilson in 1946[2], even so, the technological limitations of the moment made its use not feasible. Over the last two decades, it has gained tremendous momentum. Many new centers dedicated to it have been built, and many more are under construction or in the planning stage. [3] At the end of 2020, a hundred particle theory facilities were operating around the world, and more than 290,000 patients have been treated, approximately 250,000 with proton therapy, 39,000 with C-Ions, and 3,500 with other particles, such as He, pions, etc. [4]

To understand the rising interest in hadron therapy it must be taken into account that, in oncology, the main goal is to eliminate cancerous cells without damaging the surrounding healthy tissues or reducing this damage to a safe minimum. Since hadron therapy is a form of radiotherapy, it is essential to know what radiotherapy is and how its kinds approach the goal mentioned above.

Radiotherapy is a cancer treatment based on the fact that the interaction of radiation with cellular DNA destroys this DNA and can cause mutations in the applied cell and even cell death. Nonetheless, cells have the ability to repair themselves if the damage to the DNA is not excessive or complex. The difference in the intensity of this damage is one of the main differences in the effectiveness of different types of radiation.

The energy (Joules) per unit mass (kg) is used to measure the quantity of radiation absorbed, and this magnitude is measured in Gray (Gy).

Treating cancer with radiotherapy without damaging surrounding healthy tissues is impossible because radiation has to go through healthy parts to reach the tumor and because there exist uncertainties when defining the target volume and dose delivery. However, to reduce the damage to this tissue, when planning the radiotherapy treatment, after imaging the patient and from a dosimetric point of view, the body of the patient can be divided into three parts:

- The tumor: will be treated with the prescribed dose (Usually between 50 Gy and 90 Gy)
- The tissues and organs at risk: which usually surround the tissue and typically recieve between 1 Gy and 30 Gy.
- The rest of the body: that will receive shallow doses, usually under 1 Gy.

There are two main methods to obtain the minimum level to eradicate the tumor while getting the dose irradiated to healthy organs to the minimum. The first one, known as brachytherapy, is the use of radioactive isotopes in the form of solid seeds. However, since this method requires insertion into the patient, it can only be performed in cases where the tumor is located in accessible cavities or operable sites. This fact causes that the primary method used is the external generation of radiation.[5]

The relation between the probability of tumor control and the likelihood of normal tissue damage is reflected in the therapeutic ratio.[6] The growing interest in hadron therapy is caused by the fact that it not only improves the therapeutic ratio, it also considerably reduces the radiation applied to the nearby healthy structures, and this causes a significant reduction of the integral dose absorbed by the patient.[5] When using hadron therapy instead of traditional photon therapy, the radiation dose applied to structures surrounding the tumor, according to the International Commission on Radiation Units and Measurements, is reduced by a factor of 2 to 3. [7]

This improvement is due to the difference in the physical characteristics between photon therapy and hadron therapy. Specifically, the difference in the depth-dose curves in these two kinds of cancer treatment is hugely relevant. This curve represents the depth in the patient body (on the X-axis) versus the quantity of radiation deposited by each kind of treatment (on the Y-axis).

On the one hand, in photon therapy, this curve raises extremely quickly, getting to the maximum very early, near the surface of the patient body. From this peak, the dose decreases exponentially. This can be seen in Figure 1.1.

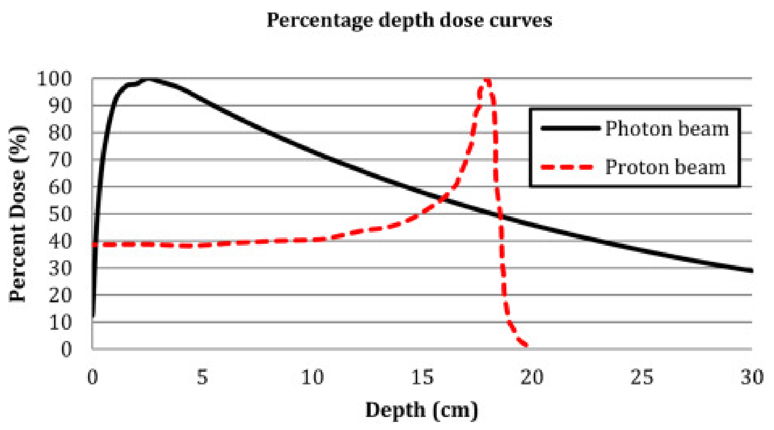


Figure 1.1: Dose deposition for a photon and a proton beam.[8]

On the other hand, protons deposit their energy mainly due to electromagnetic interactions with the electrons in the material, and the transferred energy is inversely proportional to the proton's velocity. This fact causes that when entering the patients, the protons deposit a relatively low quantity of radiation increasing rapidly as it looses speed and generating a very high peak called the Bragg Peak. After this peak, the deposited energy falls abruptly to nearly 0. Again, this shape is shown in Figure 1.1.

Having the Bragg Peak is extremely interesting for radiotherapy in order to reduce the radiation deposited in healthy zones. When entering the body, the radiation deposited before arriving at the target is significantly lower than the dose absorbed in photon therapy and extremely lower than the dose deposited in the target zone. Moreover, due to the abrupt fall in energy deposited after the Bragg Peak, there is no exit dose, defined as the dose deposited after the target tumor.

It has to be considered that, usually, the tumor surface is broader than the surface covered by the Bragg Peak. To solve this, the depth-dose curve has to be spread out, creating what is known as a Spread-out Bragg Peak (SOBP) to achieve a homogeneous dose over the target. An SOBP is created by combining different Bragg Peaks. This can be seen in Figure 1.2.

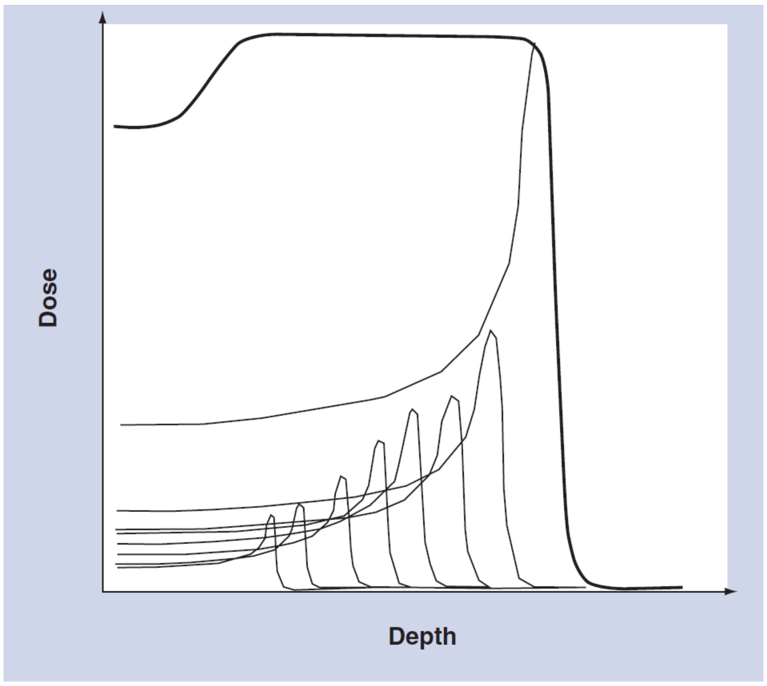


Figure 1.2: Spread-out Bragg peak.[5]

While the shape of the protons' depth-dose curve has all the aforementioned advantages, it can also cause problems and disadvantages. The fact that the radiation is highly concentrated in one spot makes different types of uncertainties (caused, for example, by organ motion, anatomical variations or dose calculation approximations) extraordinarily relevant and able to cause fatal errors in cancer treatment. If the Bragg Peak hits healthy parts, it will cause severe damage due to the high concentration. To solve this, safety margins are always applied. Nonetheless, due to the abrupt fall of the depth-dose curve of particles in hadron therapy, this can also be dangerous because unexpected variations from the treatment plan to the dose delivery can cause a part of the tumor to be left without any dose applied.[9]

However, the application of these safety margins limits notably the applicability of hadron therapy, especially when the tumor is close to high-risk organs. One of the main ways to reduce these safety margins is by achieving reliable online motorization techniques.

There are different online motorization techniques. One of the most researched ways to do this is detecting secondary radiation along the beam path, using Positron Emission Tomography (PET) to monitor $\beta+$ emitters generated during irradiation. Despite being a good technique, the usage of PET carries some disadvantages, such as the fact that the current PET devices must be adapted for in-beam utilization (integrating the scanner with the beam), the high background of prompt emissions, biological washout, and delayed signal respect to irradiation due to high half-lives of the majority of $\beta+$ emitters.[10, 11]

Another solution to perform online validation is to use the prompt emission of high-energy photons or Prompt Gamma-rays (PG). This solution is not affected by the drawbacks that the use of PET to detect secondary radiation has.[10]

The use of PG is auspicious. Not only because they can be quickly detected but also because the longitudinal distribution of the prompt radiation is highly correlated to the depth dose curve, that is, to the applied dose of radiation in every point.[12, 13] Especially because an intense peak of PG is emitted near the end of the range, that is, near the Bragg Peak. This can be seen in Figure.1.3

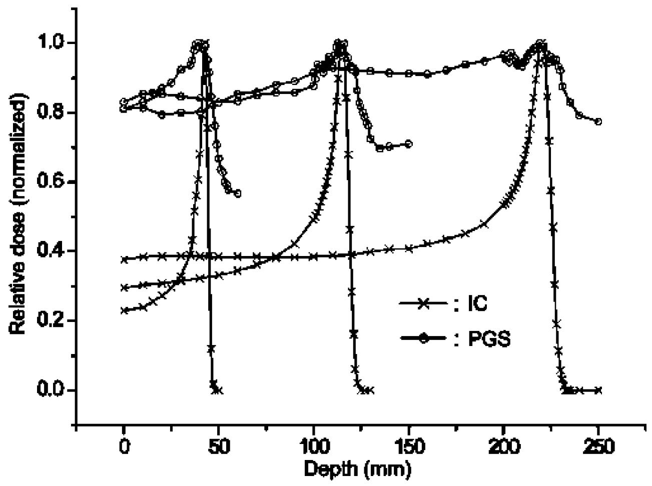


Figure 1.3: Comparisons of the depth-dose distributions with the PGS measurements at three different proton energies of 100, 150, and 200MeV.[12]

Multiple online monitoring techniques based on PG are being researched, for instance, mechanical collimation, with collimated gamma cameras, based on the PG time of flight, etc. For the interested reader, a review of PG monitoring can be found in *Prompt-gamma monitoring in hadrontherapy: a review*.[14]

One of these techniques is the use of Compton Cameras, which do not require mechanical collimation and can detect protons in the range of prompt gamma-rays. Compton imaging can be achieved by detecting three-interaction events, allowing reconstruction of the initial photon energy. Imaging with Compton Cameras can also be achieved with only two-interaction events. However, in this case, this has to be done by selecting well-known PG emission lines, requiring full energy absorption in the second interaction, or using spectral reconstruction methods.[10]

When using Compton Cameras, in every field, from astronomy to medical imaging, an event selection is crucial to separate the true signal from background noise events, reducing this noise as much as possible before image reconstruction. For this purpose, neural networks are being researched and applied.

The IRIS group of IFIC (Valencia) has developed MACACO II, a Compton Camera to achieve hadron therapy online monitoring. A basic neural network has also been used by the IRIS group to perform event selection.

This work aims to use the data and information obtained by this group in their research to create new deep learning and other state-of-the-art classification methods to improve the event selection, hence, improving image reconstruction and online monitoring of hadron therapy.

1.1 Presentation of the problem

Even though hadron therapy is one of the most promising emerging treatments of cancer, the physical characteristics of the treatment, specifically the depth-dose curve with the Bragg Peak, are a double-edged sword. When correctly applied to the tumor, it will be treated very efficiently, reducing the damage to healthy surrounding tissues. However, any uncertainty leading to an error could be critical. The concentrated radiation dose can hit critical organs, causing severe damage, or not cover the tumor completely, which will make the treatment ineffective. This creates the necessity of applying safety margins which limit the applicability of hadron therapy.

To solve this, online monitoring must be applied. The IRIS group has developed a Compton Camera to do so. However, this camera detects not only the true signal but also a considerable amount of background noise, which worsens image reconstruction.

1.2 Goals

The goals of this work can be summarized in the following 2 points:

- Create a neural network or classifier capable of classifying events detected by a Compton Camera as true signal or background noise, trained with Monte Carlo simulations.

- Improve the results of previous experiments conducted by the IRIS group.

1.3 Structure of the work

This work is divided into seven chapters:

The first chapter is the introduction. In it, a theoretical base is given to present the work's context. This theoretical introduction explains briefly the history of hadron therapy, as well as some medical and physical aspects regarding it and the experiments performed by the IRIS group of the IFIC in the MACACO II project. In this chapter, the problem, goals, and this section, the structure of the work, are explained.

Chapter 2 is the one that explains the state of the art. To do this, it provides an overview of the best technologies used nowadays in general classification using machine learning. After that, it presents the MACACO II approach and results, that is, the state of the art for this exact problem.

In chapter 3 the proposed solution is presented. The different steps taken to achieve our goals are enumerated and explained.

To execute this work many tools have been used, both hardware and software tools. These are vital, so the main ones are reflected in chapter 4.

When applying machine learning techniques, the understanding of the dataset is crucial. For this reason, an extensive analysis of the MACACO II dataset, provided by the IFIC is performed in chapter 5.

Chapter 6 is the longest chapter in this work. This chapter explains the performed experiments, from dataset transformation to different models of neural networks and gradient boosting classifiers. It also provides the analysis of the results of these experiments, both quantitative, using different metrics, and visual, with a multitude of graphs.

Lastly, chapter 7 provides the conclusions of this work.

There also have been written two appendixes: Appendix A, in which more scatterplots for further analysis of the models' output are presented for the interested reader, and appendix B, in which the Sustainable Development Goals (SDG) which match this work are explained.

CHAPTER 2

State of the art

It must be remembered that this work is based on the experiments and information obtained by the IRIS group in their hadron therapy online monitoring project MACACO II. This means that in the review of the state-of-the-art, the most crucial part is to analyze the results obtained in the event selection part of this project.

Their information is extremely relevant since their approach deals with exactly the same problem as us, with the same data and classification goals.

Besides, other solutions facing similar problems, in similar contexts, and with similar goals will be explored in order to obtain ideas on how to improve the results achieved by the MACACO II project until this moment.

To review the state-of-the-art, we will go from a broader view, having an overview of the state-of-the-art classification algorithms in general, to a more specific review of the MACACO II results, which approaches the same problem as us.

2.1 General classification

There are many papers reviewing the state of the art of classification algorithms. One of the most updated ones is *An up-to-date comparison of state-of-the-art classification algorithms*[15], published in October 2017.

In this article, after many experiments to benchmark the best machine learning classification algorithms, it was concluded that the three best performing classification algorithms were stochastic gradient boosted distributed decision trees, random forest classifiers, and support vector machines.

Support vector machines

In the third place, we find support vector machines. [16] A basic support vector machine approaching a supervised learning problem functions in the following way:

Given a bunch of labeled training examples belonging to two different classes, a support vector machine will try to separate them by using a line or a hyperplane. However, there will usually be more than one possible line or hyperplane that can separate these classes.

The way in which support vector machines choose the best line or hyperplane to separate the classes is what gives them their name. To do this, the support learning machine will take one point in each class. These points are called support vectors and will usually be the nearest points to the separating line.

The best line will be that one with the maximized margin, that is, the one with a bigger separation between these support vectors and itself.

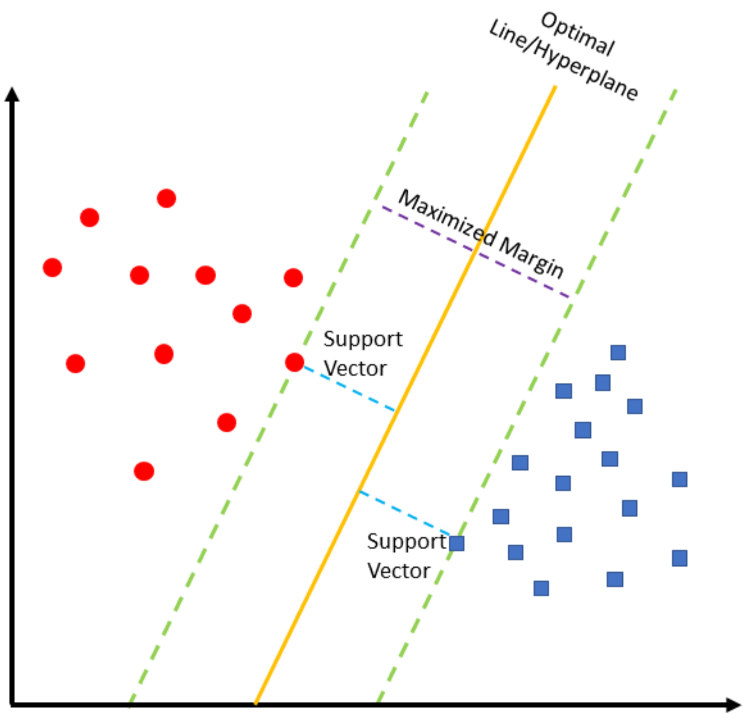


Figure 2.1: Example of Support Vector Machine.

Random forest classifiers

In the second position, there appear Random Forest Classifiers.[17] This is a classification algorithm based on building an ensemble of trees, that is, a group of decision trees.

The word random in their name refers to two main things:

- Each decision tree is trained with random subsets of the dataset by using bootstrapping.
- Each decision tree will only use a few random different features to make its decisions.

These two things are what make the trees in the random forest less correlated, and that is what makes this algorithm interesting.

Once the forest is constructed, when facing a classification task, the class assigned to one sample will be chosen simply by voting. One sample will be assigned to the class the majority of the trees predicted.

Stochastic gradient boosted distributed decision trees

SBDTs are a type of gradient boosting classifier, which, similarly to the random forest classifier, is based on building many decision trees.

However, there is a relevant difference. While Random Forest Classifiers are based on parallel decision trees, SBDTs are based on a sequential process.[18]

This sequential process consists, basically, in adding one decision tree after another in order to improve the performance of the previous one. This is known as boosting.[19]

The decision trees added after each other are called weak learners, and a model obtained from combining many decision trees in series is called a strong learner.

2.2 MACACO II project

2.2.1. Approach

In the MACACO II project, the researchers of the IFIC implemented a fully connected neural network.

This network was composed of a fully connected layer of 80 neurons employing the ReLU activation function, a dropout of 0.25 after this layer for regularization in the training stage, a fully connected neural network of 40 neurons using, again, a ReLU activation function, and finally, an output layer employing a Sigmoid activation function.

The output of this neural network was a number between 0 and 1. If the result was greater than 0.5, then the event was considered a true signal and classified in class 1. If the result was equal to or less than 0.5, it was considered class 0, that is, background noise.[10]

2.2.2. Results

IFIC-Valencia presented the results of the fully connected neural network employed at the MACACO II project with few metrics. Specifically, only the precision and recall of class 1 (true signal) were published. These can be seen in Table 2.1.

class	precision	recall
1	10.3%	84.0%

Table 2.1: MACACO II neural network results. [10]

CHAPTER 3

Proposed Solution

To achieve the goals proposed in Section 1.2 we propose to execute the following steps:

3.1 Dataset transformation

After understanding the characteristics of the dataset provided by the IRIS group, as seen in Section 5.1, we will transform the dataset so the neural networks and classification model designed in the next steps of the solution have a better input, achieving better results.

3.2 Design of alternative architectures of neural networks

The second step of the solution will be to design one or more new neural network architectures to beat the results of the neural network described in the MACACO II paper[10] in certain metrics, such as the precission, and, mainly, the recall.

This will be done not only changing the number of layers and the number of neurons in each hidden layer, but also, for example, trying different activation functions, applying dropout in the training stage between some layers and applying batch normalization.

3.3 Implementation of Gradient Boosting Classifiers

As seen in the state of the art review, gradient boosting classifiers is one of the best-performing kind of classifiers nowadays.

For this reason, we will implement different gradient boosting classifiers, trained using the multitude of options the XGBoost library offers for fine-tunnig. This will let us create classifiers that improve the results obtained by the IFIC, also with them we will try to improve metrics which are not commented in the MACACO II paper but could led to a better performance in the classification.

CHAPTER 4

Utilized software and hardware

The execution of this work has been possible due to the existence and utilization of a multitude of software and hardware tools. In this chapter, the most relevant of these tools used to execute crucial parts of this research, such as data transformation, neural network creation and training, and visualization of results, will be briefly presented.

4.1 Software

4.1.1. TensorFlow and Keras

TensorFlow is a machine learning library designed for large-scale and heterogeneous environments, allowing deployment using CPUs, GPUs and TPUs. The Google Brain team developed it to substitute their previous proprietary machine learning system DistBelief, launched in 2011. Tensorflow was released as an open-source library in 2015.[20]

Even though this work has been executed using Python programming language, and the developed neural networks have been created using the TensorFlow version for this language, Tensorflow is available for other programming languages such as C, C++, or JavaScript.

Keras[21] was first created in 2015, some months before the launch of TensorFlow, by the Google developer François Chollet. It provided an abstraction for Machine Learning. Compared to the Machine Learning libraries used until that moment it was straightforward to use, making it easier to learn and develop the majority of solutions and experiments faster.

To work, Keras needs a backend. The original default backend of Keras was Theano[22]. A while after TensorFlow was released, Keras started supporting it as a backend, quickly gaining popularity. As a consequence, TensorFlow became the default backend of Keras.

After this, in 2017, TensorFlow added support for Keras, and since the release of TensorFlow 2.0 in 2019, Google declared Keras as the high-level API of TensorFlow.

Every neural network in this project has been implemented using these two technologies.

4.1.2. Scikit-learn

Scikit-learn is a Python library widely used in machine learning. It focuses on ease to use, documentation, performance, and API consistency. It offers a wide range of state-of-the-art machine learning algorithms such as K-means, Gradient Boosting Classifier, and

Random Forest Classifier. It also includes many utilities for machine learning, such as train-test dataset division and multiple evaluation functions for machine learning.[23]

It started as a Google Summer of Code project by David Cournapeau in 2007. Later, in 2010, the first public release was made.

4.1.3. Matplotlib

Matplotlib is an open-source portable Python library to plot, image, and, in general, visualize scientific, engineering, and financial data in an easy way.

Its vision is to make the creation of high-quality data representation, ready to publish in scientific texts, easy. Specifically, its goal is to make it possible with a few lines of code instead of the more complex steps needed to create plots with other software existing before the launch of matplotlib. When creating matplotlib other goals were set, such as making it embeddable in GUIs for application development, making the code understandable, and achieving high quality with an especial focus on the text.[24]

4.1.4. XGBoost

XGBoost is a large-scale, scalable, tree-boosting system. It provides a library for many programming languages such as R, C++, and Python, which is the language used for the execution of this work.[25]

It was created by Tianqi Chen, and it was first released in 2014. It gained popularity since it was used by many winning teams in various Machine Learning contests.

Our primary interest in this library is the fact that it allows implementing Gradient Boosting Classifier and, contrary to the version of this ML algorithm implemented in scikit-learn, XGBoost allows for fine-tuning it, using parameters such as the maximum depth, the number of estimators, and more importantly, as we will see in Chapter 6, the parameter called *scale_pos_weight*.

4.1.5. Netron

Netron is a neural network visualization software.[26] It provides multiple implementations, including a web version, which was the one used in this work.

It has been use to represent and visualize the multiple neural network models explained in Chapter 6.

4.2 Hardware

Even though a massive variety of hardware has been used during the development of this work, the most remarkable pieces are the GPUs.

4.2.1. GPU

Despite being possible to execute nearly every machine learning and deep learning algorithm using only CPUs, even on average laptops, this process will be extremely tedious in case of working with large-scale projects, with medium to large quantities of data and lots of experiments to be performed.

It is well known that Artificial Intelligence and Machine Learning are not genuinely new concepts. They have been used for decades since the mid-XX century. However, in the last years, Machine Learning has increased its momentum heavily, gaining much interest and being more present than ever in any field.

This can be explained by multiple factors, such as the reduction in basic usage learning difficulty with the creation of high-level APIs or the enormous availability of data generated by the most digitized world ever seen. However, indisputably, one of the main reasons for this machine learning boom is the availability of computational power. This availability has been made possible, in a relevant part, by the GPUs.

GPUs have proved to perform exceptionally well in Machine Learning and Deep Learning tasks due to the similarity of the graphics applications they were designed for and the computations used in Machine Learning. For this reason, they have been highly used to accelerate ML in the last years.

In this project, we have been able to use some powerful GPUs owned by the Pattern Recognition and Human Language Technology Research Center. Specifically, two units of Nvidia GeForce RTX 2080, with 8GB of GDDR6 VRAM, 2944 CUDA cores, 368 Tensor cores, and 46 RT cores each. This has accelerated the work a lot and, consequently, made it possible to perform more experiments.

CHAPTER 5
The dataset

5.1 Dataset description

When applying machine learning to the solution of any problem, the most critical part is the dataset. It's vital to have a set of data with the necessary information to train and test any classifier or neural network implemented.

In this case, since the goal of the research is to obtain a machine learning solution to the problem of classifying Compton Camera detected events into noise or signal, we need to have a set of examples of these events annotated so we know which events correspond to noise and which correspond to the true signal. However, this is impossible to obtain from a real experiment since, once the planes of a Compton Camera detect an event, it is not known whether it is signal or noise. In fact, if that were possible, this work would not make sense since that is its goal.

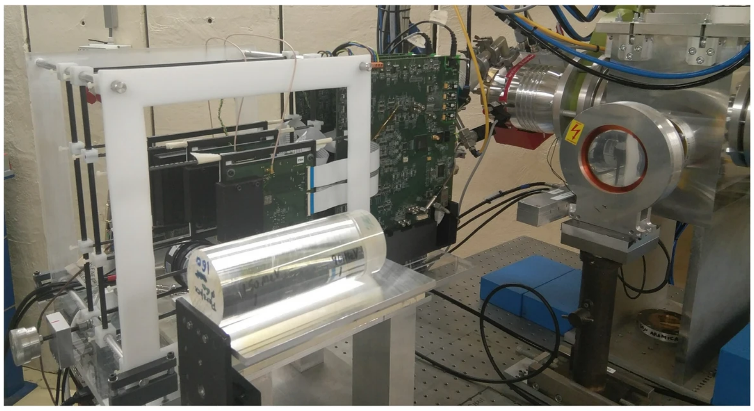
Then, since we need to have this information, we will have to rely on Monte Carlo simulations. These simulations make it possible to know precisely which detected events are part of the signal and which are noise. For this reason, data generated using Monte Carlo simulations will be ideal for training neural networks and other kinds of classifiers.

The dataset provided by IFIC Valencia, which will be the base of this experimentation, consists of three files and can be divided into two main parts: The simulated data and the real data, obtained from experiments performed at the irradiation facility of the AGOR cyclotron at KVI-Center for Advanced Radiation Technology, University of Groningen, the Netherlands, as shown in Figure 5.1.[10]

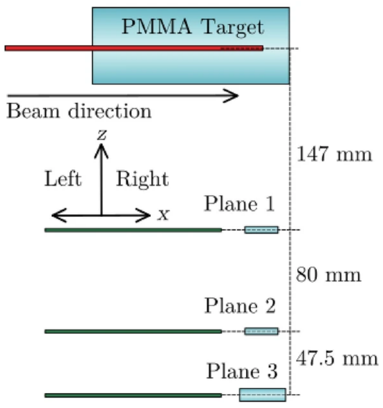
In these experiments a beam of particles incided into a PMMA target generating photons, which can be detected by the Compton Camera. This camera also detects noise events such as other photoelectric processes or events with neutrons, protons, electrons or positrons which come from the phantom.

5.1.1. Simulated data specifications

There are two simulated data files: one of them containing noise events detected by the Compton Camera and another containing signal events. These files are the biggest ones, with 1,599,131 noise events and 211,291 signal events. These simulations were performed using GATE 8.2[27] with Geant4 10.5[28], and the physical processes and probabilities were defined using the QGSP-BIC-HP-EMZ physics list. In the simulated experiments, the detector planes were placed at the same position as the detector planes of the Compton Camera are situated in the real experiment, as represented in Figure 5.1b. However, its dimensions differ from the planes in the real experiment, being significantly bigger in



(a)



(b)

Figure 5.1: (a) Picture of the experimental set-up showing beam exit, PMMA target and Compton Camera. (b) Diagram of relative distances in the experimental set-up.[10]

the simulations (where its area was set to 100 x 100 mm^2) in order to get more data and reduce simulation time considerably.[10]

5.1.2. Real data specifications

The real data is the one obtained with the experimental setup shown in Figure 5.1b, without any kind of modification. It is stored in one text file which contains 254,368 events.

5.1.3. Event specifications

It has to be remarked that, despite the fact that the used Compton Camera is composed of three detector planes, the data of two of them were given to us by the IRIS group, precisely, only the data of the second and third detector planes. Consequently, we will also use only these data to train and evaluate the models in our experiments. [10]

A vector of 16 components represents each event from the raw dataset; however, a big part of them will not be useful for us. This fact will be further explained in Section 6.1: Data treatment and transformation. The first component of every event is always the number 2. It is followed by the information of the three Compton Camera planes, where each one of them has a first component indicating the number of the plane. It is crucial to note that this number identifying the plane is not associated with the order of the planes in the Compton Camera, which means that the number 1 in this vector representation does not correspond to the first plane. Specifically, the plane represented with the number 1 is the second plane, the one represented with 2 is the third plane, and the plane represented with the number 3 is the first plane. After this identifying number, there is a second component indicating the X coordinate of the place where the event is detected in the plane, a third component indicating the Y coordinate of the place where the event is detected in the plane, a fourth component indicating the Z coordinate of the place where the event is detected in the plane and a fifth component indicating the energy of the event in the plane.

All together, each vector of the dataset looks as follows:

$$(2, 1, X1, Y1, Z1, E1, 2, X2, Y2, Z2, E2, 3, X3, Y3, Z3, E3) \tag{5.1}$$

To make this structure clearer, it is helpful to see and comment on some examples, so in the table below, a few of them are provided (with values rounded to the second decimal digit).

2	**1**	**X1**	**Y1**	**Z1**	**E1**	**2**	**X2**	**Y2**	**Z2**	**E2**	**3**	**X3**	**Y3**	**Z3**	**Z3**
2	1	-10.17	1.35	225.58	87.62	2	-4.78	6.62	275.76	194.34	3	0	0	0	0
2	1	7.96	11.71	226.31	145	2	-2.77	6.46	278.06	444.97	3	0	0	0	0
2	1	-6.84	11.18	224.5	148.26	2	-18	16.2	273.56	290.54	3	0	0	0	0
2	1	2.66	4.99	227.33	67.81	2	9.87	4.11	278.10	446.46	3	0	0	0	0
2	1	9.14	-5.88	224.5	4189.3	2	-16.74	2.28	274.59	453.49	3	0	0	0	0

Table 5.1: Examples of events obtained from the real experiment.

Since, as commented at the beginning of this subsection (5.1.3), the dataset given by the IFIC only contains information about the planes 2 and 3, it can be clearly seen that the columns corresponding to plane 1 (which in the vector representation is the number 3) are always filled with zeroes. In contrast, the data of the planes stored as number one and two (representing planes 2 and 3 respectively) are filled with different measurements.

CHAPTER 6
Experimentation and results

To achieve the goals set in Section 1.2 it's essential to perform many experiments. In this sixth chapter all these experimental procedures will be listed in order of execution, describing them, their purpose and their results.

6.1 Data treatment and transformation

The first approach to improving the results of the MACACO II project was to perform a data transformation in the dataset and use the transformed dataset later to train the experimental classification models.

As it has been seen in Section 5.1, in the original event vectors of the dataset, there existed many relations between components, such as the relation between the coordinates where an event has been detected on the second and third plane, as well as the one corresponding to the energies of a single event on each of the two planes. However, the ease with which humans can see and interpret these relations is caused by their knowledge of the field and the context and the components of the vector separating the different planes.

In order to improve the machine learning comprehension of these relations, they should be reflected mathematically.

For these reasons, it was decided to transform the data in order to remove the unnecessary components and reflect the relations between the necessary ones.

The dataset was treated and transformed, going from event vectors of 16 components to event vectors formed by the following six components:

$$(|r|, \theta, \phi, E1, E2, \Delta E) \tag{6.1}$$

These vectors can be divided into two main parts. The first one, with the first three components, represents the data of the coordinates in the two planes. The second part represents the data of the energy of each event in the planes.

6.1.1. Coordinate representation

The positional data of the events was previously represented by two sets of three-dimensional Cartesian coordinates, one for the position where each event was detected in each plane

(X1, Y1, Z1, X2, Y2, Z2). That is, two points in the three-dimensional space represented the positional information of an event.

The most intuitive ways to mathematically relate two points in a 3D space are a vector or a line. Considering that, in this case, what these 2 points represent is the detection of a moving particle and that the movement of the particle should be from the second plane to the third plane, the more appropriate these two ways is the representation as a vector, because the vector includes the concept of sense. This concept will let us represent the aforementioned fact that the particle should travel from the Compton Camera's second to the third plane.

A three-dimensional vector can be represented in many ways. The one chosen in our approach was the representation of our vector as the radius vector of spherical coordinates. This concept will be further explained in Subsection 6.1.1.

Spherical coordinates

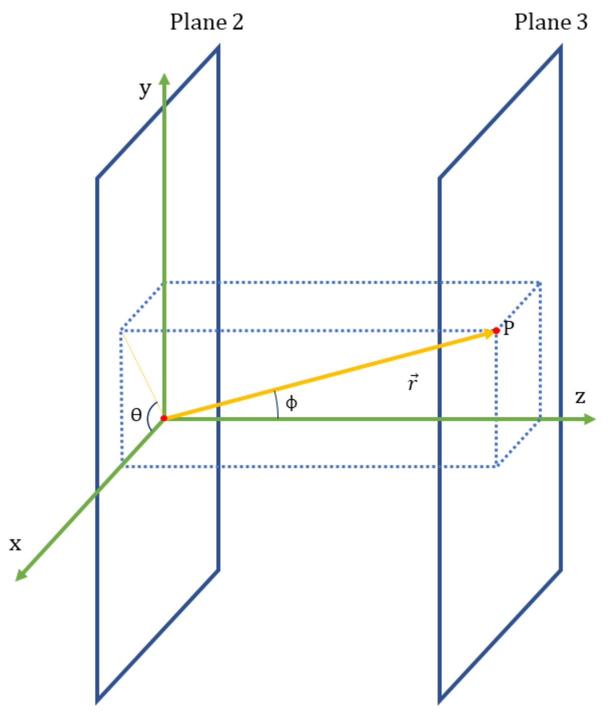


Figure 6.1: Spherical coordinates diagram.

Spherical polar coordinates are, as their name suggests, a system of coordinates commonly used to define positions in a sphere. They can be seen as an extension of the polar coordinates used in two dimensions.

On the one hand, polar coordinates are defined by two parameters: the distance between the origin and the point P, denoted by r and the angle θ, which is defined as the angle where the point P is measured counterclockwise from the x-axis. Polar coordinates are also frequently used to express a vector in its polar form, defining the vector as the one starting at the origin of spherical coordinates and ending at the point P.

On the other hand, spherical coordinates have one more defining parameter, another angle. That is, three parameters define spherical coordinates:

- r, which is, again, the distance from the origin to P

- The azimuthal angle θ
- The polar angle ϕ

The radius vector of the spherical coordinates is the one starting at the origin and ending at point P. The angles can also be defined by taking this vector as a reference. For instance, the azimuthal angle can be defined by the angle formed by the projection of the radius vector in the plane defined by axes X and Y from the x-axis in the counterclockwise direction.

In the original dataset, as mentioned in Section 5.1, the positions of the detected particles were represented as 2 points, one in the second plane and the other in the third one. The first step to transforming it to the spherical representation is calculating the vector $\vec{v}$ starting at the spot where the particle hit the second plane and ending at the point where it hit the third plane. That is done as shown in Equation 6.2

$$\vec{v} = (x_3 - x_2, y_3 - y_2, z_3 - z_2) \tag{6.2}$$

Once we have the vector $\vec{v}$, formed by its 3 components x, y and z, to obtain the polar representation, we must apply equations 6.3, 6.4 and 6.5

$$r = |\vec{r}| = |\vec{v}| = \sqrt{x^2 + y^2 + z^2} \tag{6.3}$$

$$\theta = tan^{-1}\left(\frac{y}{x}\right) \tag{6.4}$$

$$\phi = tan^{-1}\left(\frac{\sqrt{x^2 + y^2}}{z}\right) \tag{6.5}$$

6.1.2. Energy

The last three elements of the new vector representing events are energy-related components, and are:

- The energy of the particle detected in the second plane ($E1$)
- The energy of the particle detected in the third plane ($E2$)
- The energy differential of detected events between planes (ΔE), defined as seen in Equation 6.6

$$\Delta E = E2 - E1 \tag{6.6}$$

6.2 Replicating the MACACO II neural network and training it with the whole transformed dataset

As it was explained in Section 2.2, the neural network implemented in the MACACO II poject waf formed by a fully connected layer of 80 neurons employing the ReLU activation function, a dropout of 0.25 after this first layer for regularization, another fully connected neural network of 40 neurons using a ReLU activation function, and an output layer employing a Sigmoid activation function.[10] A diagram of this neural network, adapted to our new transformed input, can be seen in Figure 6.2

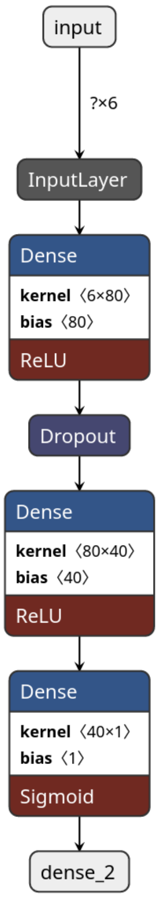


Figure 6.2: Diagram of the neural network implmented for the MACACO II project, adapted to the transformed dataset.

The performed data transformation was done following the goal of improving the performance of the classification. However, it's vital to check if this input format is actually good for achieving this mission. To check this, we have implemented the model described in the MACACO II paper, with a few adaptations to use as input our transformed dataset.

First, we trained this neural network with the whole transformed dataset and obtained the results seen in Table 6.1

	precision	recall	f1-score
0	0.97	0.65	0.78
1	0.24	0.84	0.38
accuracy			0.68
macro avg	0.60	0.74	0.58
weighted avg	0.88	0.68	0.73

Table 6.1: Results of the neural network implemented for the MACACO II project, adapted and trained with the whole transformed dataset.

To compare with the results obtained in the MACACO II project, as it was already commented on Subsection 2.2.2, we can only rely in 2 metrics, the precision of the class 1 and the recall of the class 1.

When comparing these metrics, we can see that the data transformation lead better results. Even though the recall of the class 1 stayed at 84%, we have increased the precision of the true signal by a 133.01%, achieving a class 1 precision of 24%.

As mentioned in Section 5.1, the detector planes on the simulations used to train this neural network were significantly bigger than the ones used in the real experiment Compton Camera. For this reason, another interesting experiment is to train this same neural network using only the events that happened inside the simulated area corresponding to the real detector area of the plains. That is, to filter every event which any of its coordinates was bigger than 15 or smaller than -15.

This was done, and the results of training the neural network with the filtered transformed dataset were the following:

	precision	recall	f1-score
0	0.96	0.62	0.75
1	0.22	0.79	0.35
accuracy			0.64
macro avg	0.59	0.71	0.55
weighted avg	0.87	0.64	0.70

Table 6.2: Results of the neural network implemented for the MACACO II project, adapted and trained with the masked transformed dataset.

It can be clearly seen that these results have a poorer precision and recall for class 1. Despite the fact that the recall of class 1 is lower than the one obtained by the MACACO II neural network, the class 1 precision continues to be more than double, reaching 22%.

It is obvious that, in plain numbers, the neural network trained with only filtered data yields poorer results. However, this not mean necessarily that when processing real data it will lead to worse image reconstruction, since the dataset used for its training will be more similar to real data. This will be analized in Section 6.6.

6.3 Trying to maximize class 1 recall

After observing the results obtained in Section 6.2, we can arrive to the conclusion that we have already extremely improved the precision compared to the MACACO II results. However, the recall stays the same when using the whole dataset.

Even though we can not known it for sure at this point, class 1 recall can be assumed to be very important to reconstruct the images, that is, we must preserve the maximum number of true signal events if we don't want to lose information.

For these reasons we try some different neural network architectures trying to maximize the recall.

6.3.1. First experimental neural network with 2 hidden layers

In the first attempt of increasing class 1 recall a neural network with 2 hidden layers was designed, containing each one 64 neurons and using the ReLU activation function. After each hidden layer, a dropout layer with a percentage of 30% was also included.

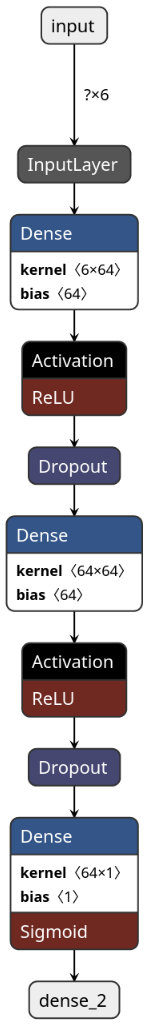


Figure 6.3: Diagram of the designed 2 hidden layer neural network.

This new architecture obtained the results displayed in Table 6.3.

	precision	recall	f1-score
0	0.97	0.63	0.76
1	0.23	0.86	0.37
accuracy			0.66
macro avg	0.69	0.74	0.57
weighted avg	0.89	0.66	0.72

Table 6.3: Results of the new 2-layer neural network trained with the whole transformed dataset.

When applying the new 2 layer neural network architecture to the unfiltered transformed data we can see a quite significant improvement of the class 1 recall, which in the MACACO II neural network was 84%. With this new architecture the class 1 recall reached 86%. However it must be noted that class 1 precision was decreased from 24%

to 23%. It also has to be noted that other relevant metrics, which were not commented in the MACACO II paper, such as the recall weighted average also decreased.

If we train this new neural network architecture with the filtered transformed dataset we obtain the results shown in Table 6.4.

	precision	recall	f1-score
0	0.96	0.60	0.74
1	0.22	0.82	0.35
accuracy			0.63
macro avg	0.59	0.71	0.54
weighted avg	0.87	0.63	0.69

Table 6.4: Results of the new 2-layer neural network trained with the masked transformed dataset.

When comparing the results of the 2 layer neural network trained with filtered transformed data and the results of the MACACO II neural network trained with the filtered transformed data we see amazing results.

In the first place, the goal of increasing the class 1 recall was perfectly achieved, going from a class 1 recall of 79% to a class 1 recall of 82%, which is a three percentual point increase. Moreover, the precision of class 1 stayed the same (22%), so we can state that the 2-layer model improved the MACACO II one. The recall weighted average has slightly decreased, as happened with the network trained with the whole dataset. However, this decrease has been of only one percentual point.

6.3.2. 3 hidden layer neural network with Batch Normalization

The next step in trying to maximize the class 1 recall was to add another equal hidden layer to the previous neural network and to apply batch normalization after every hidden layer.

This lead to an architecture of 3 hidden layers, with 64 neurons each, using the ReLU activation function and applying batch normaliation after every hidden layer.A dropout layer with a percentage of 30% was also included after each hidden layer.

A diagram of this neural network can be seen in Figure 6.4

Again, this neural network was trained first, with the whole transformed dataset, and after that, with the filtered transformed dataset. The results of training this neural network with the whole transformed dataset are presented in Table 6.5.

	precision	recall	f1-score
0	0.98	0.55	0.70
1	0.21	0.92	0.34
accuracy			0.59
macro avg	0.60	0.73	0.52
weighted avg	0.89	0.59	0.66

Table 6.5: Results of the new 3-layer neural network with Batch Normalization trained with the whole transformed dataset.

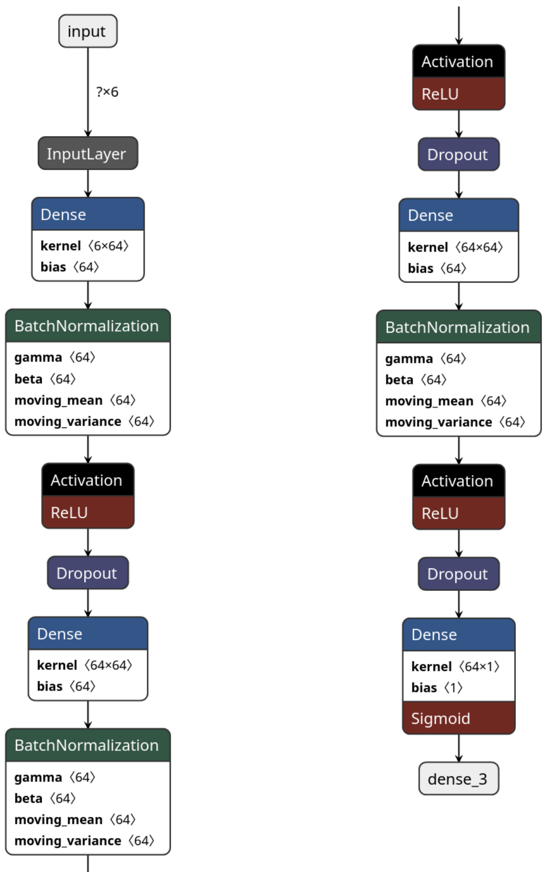


Figure 6.4: Diagram of the desiged 3 hidden layer neural network.

With this new 3 hidden layer neural network, with batch normalization, we achieved to increase the class 1 recall, which was our mission, reaching 92%.

However, it has to be taken into account that in order to pursue the recall of class 1, the recall weighted average was reduced compared to the previously implemented model. The precision of class one also decreased by one percentual point compared with the 2-layer model.

Again, the same model was trained using only filtered transformed data, and the results obtained are reflected in Table 6.6. With these results it's clear that the transformed

	precision	recall	f1-score
0	0.97	0.53	0.68
1	0.21	0.88	0.34
accuracy			0.57
macro avg	0.59	0.70	0.51
weighted avg	0.87	0.57	0.64

Table 6.6: Results of the new 3-layer neural network with Batch Normalization trained with the masked transformed dataset.

dataset has made a huge improvement in the previous results. However, it will be very interesting to know how every component of the transformed event vector, described in equation 6.1, influences in the classification performed by the implemented neural networks. This will be done in Section 6.4.2

6.4 Neural network output analysis

6.4.1. Introduction

To see the effect of the different parameters of each input vector in the classification of the implemented neural networks we will plot a series of graphs.

Specifically, since first we were trying to increase the recall of class 1 it is very interesting to use the neural network with higher score on this metric. That is the neural network of 3 hidden layers with batch normalization.

The classification of unmasked simulated data, masked simulated data and real data have been analyzed, both for the 3 layer neural network trained with unmasked data and the 3 layer neural network trained with masked data.

In short, we will analyze these six scenarios:

- Unmasked simulated data classified by the neural network trained with unmasked simulated data (Figure 6.5).
- Masked simulated data classified by the neural network trained with unmasked data (Figure 6.6).
- Real data classified by the neural network trained with unmasked data (Figure 6.7).
- Unmasked simulated data classified by the neural network trained with masked simulated data (Figure A.1).
- Masked simulated data classified by the neural network trained with masked simulated data (Figure A.2).
- Real data classified by the neural network trained with masked simulated data (Figure A.3).

To do this, in each of the six scenarios, 15 scatter plots will be created, each one representing one parameter of the input in one axis versus each other parameter of the input in the other axis.

Each one of these scatter plots has been generated using a value of 0.1 for the α channel, that is, a 90% of transparency.

In the figures representing the classification of simulated data the dots have been printed in the following order:

- First the **blue** ones, representing **true negatives**.
- Then, the **red** ones, representing **false positives**.
- In the third place, the **green** ones, representing **true positives**.
- Lastly, the **orange** ones, representing the **false positives**

6.4.2. Neural network trained with unmasked data

Figure 6.5 represents the unmasked simulated data filtered by the 3 layer neural network trained with unmasked simulated data.

We will start analyzing Figure 6.5 from the top-left graph. This first graph represents the module of the radius vector on the X-axis and the θ angle on the Y-axis. It can be

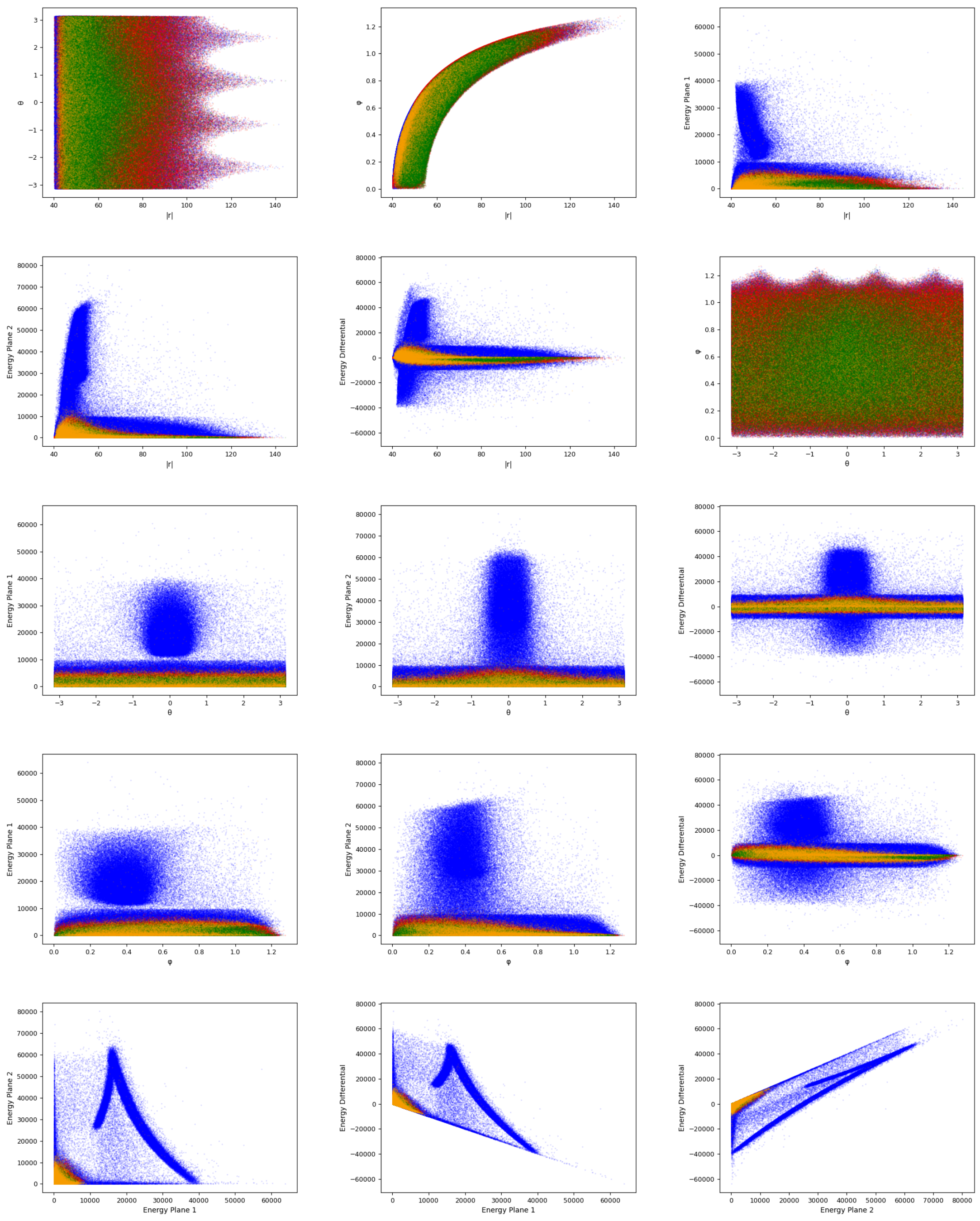


Figure 6.5: Unmasked simulated data classified by the neural network trained with unmasked data into true negatives (●), false positives (●), true positives (●), and false negatives (●).

clearly seen that there exists a thin vertical line at the start of the X-axis, near the value of 40, where the network classifies the events correctly as noise. Then, moving to the right, we can see another vertical strip containing both many false negatives and many false positives as a transition to the next zone, populated mainly by true positives. This zone extends to the right of the X-axis, reaching values between 60 and 80 on it. In the center of the Y-axis near the theta angle of zero, it can be seen that this zone extends slightly to the right. Once this zone ends, the rest to the right is dominated by red. That is, the network starts to classify most events as noise, and since true negatives are printed before false negatives, the color of these last ones is the most visible one.

The subfigure in the position [1,2] shows that the relation between the module of the radius vector and the phi angle is not seen as especially relevant by the neural network. We can see both elements classified as signal and elements classified as noise along the whole spectrum of the graph. There is only a small zone in the top right of the graph where the neural network tends to classify the events as background noise.

The top right subfigure shows the module of the radius vector on the X-axis and the energy deposited in the plane one on the Y-axis. These two parameters seem significant since we can see a clear separation in the graph. The events classified as class 1 by the neural network are the ones on the lower part of the graph, occupying the bottom part of the 0-10,000 range of energy in plane 1, mainly in the middle values of the X-axis. In the bottom left part, we can also see a bunch of false negatives surrounding the true positives. The cloud of points with energies over 10,000, located mainly between the values 40 and 60 of the vector of the radius module, is classified as noise.

What we have visualized can be generalized to a pattern that will be repeated in multiple subfigures. The generalization of this pattern is presented in every graph where the energy of one of the planes is represented at the Y-Axis. This generalization can be described as a bottom fringe under the Y value of 10000, where true positives and false negatives are concentrated, although distributed differently in each of the graphs. A little higher on the Y-axis, there is a transition line with an abundance of false negatives and a top part of the fringe, and an upper cloud of points that are classified as true negatives. This pattern can be seen in subfigures [1,3], [2,1], [3,1], [3,2], [4,1] and [4,2].

The following analyzed graph will be the one in the position [2,2]. In it, we see that the majority of events classified as signal by the neural network are very close to the value 0 for the energy differential. We see a lot of false negatives in the central zone situated in the same X-axis range as the giant cloud of true negatives in the range of 40-60 for the module of the radius vector. Again, around the false negatives and true positives, we see a transition zone with false positives, mainly in the lower part.

We can also generalize this pattern, seeing that it is similar to the graphs in positions [3,3] and [4,3]. These have the following features: A central strip, near the value of 0 for the energy differential, where true positives and false negatives are mixed, and that can vary slightly in shape, mainly in its two ends, and a cloud of true negatives in different positions in each of the graphs.

The next interesting graph is the one located at position [2,3]. This graph is the one that shows the least clear separation. We see a vast cloud filling the center with many true positives, combined with a few false negatives and false positives. The rest of the graph is dominated by false positives, only seeing a more significant presence of true negatives at the edges.

Looking at the last three figures, we can see peculiar formations with four sides. These shapes seem very different in each of the tables. However, they are similar in one of the critical facts. The corner of each of these formations situated at the 0 value for both the X-axis and the Y-axis is the only place where we find events that are not true negatives. We mainly see false negatives in these corners since they are the last events printed.

There are two obvious things to notice if we compare the unmasked and the masked data classified by the NN.

First, some plots have a different look in the masked data. Due to the filtering of the data, some remarkable structures have been reduced considerably. The clouds of true negatives and the V-shaped structures in the last three graphs are clear examples. Moreover, some axes have changed due to the elimination of events. The more significant changes can be seen in the axes representing $|r|$ and φ.

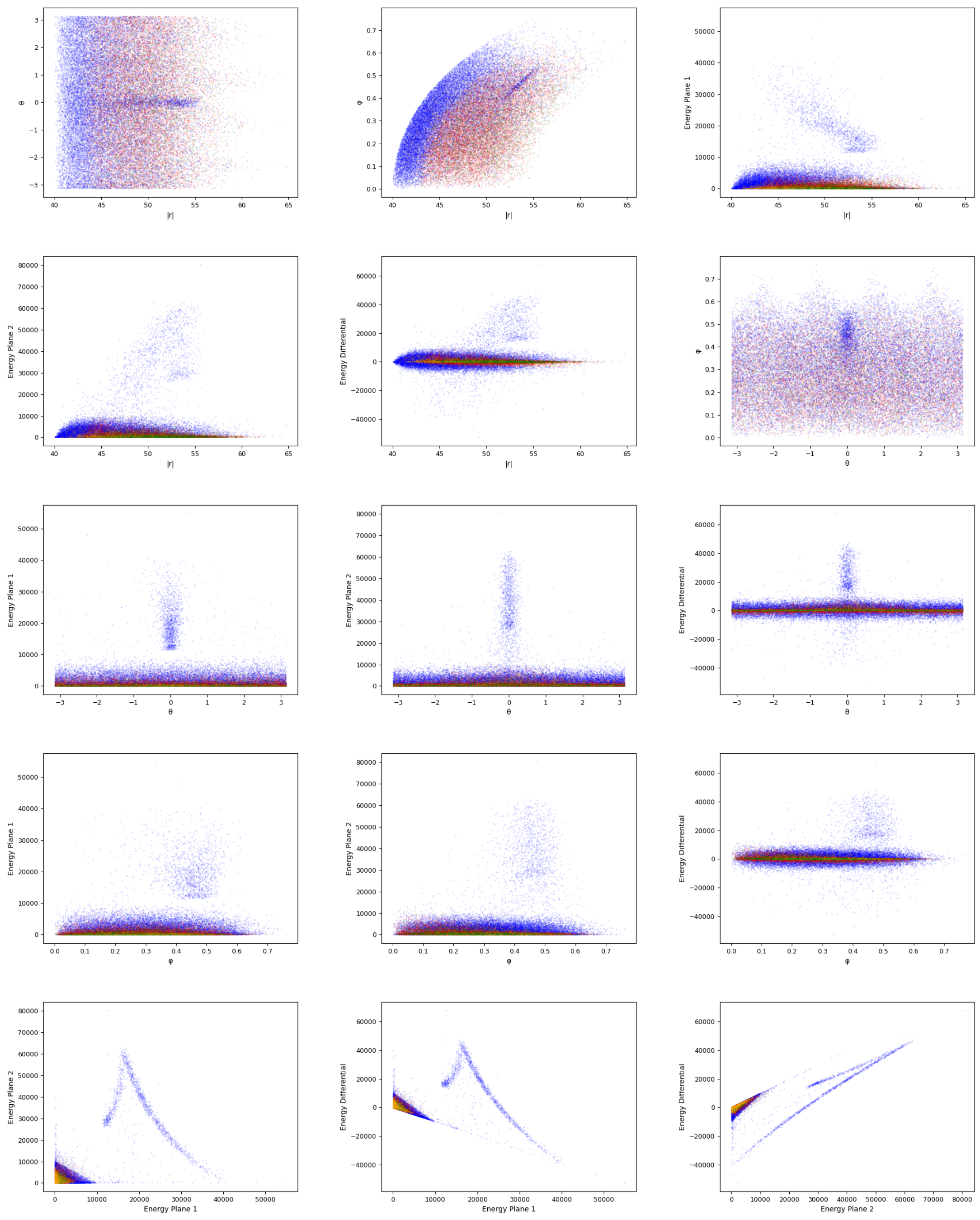


Figure 6.6: Masked simulated data classified by the neural network trained with unmasked data into true negatives (●), false positives (●), true positives (●), and false negatives (●).

The second noticeable thing is that since we are using the transparency of 90% and in this case there are fewer events, we can almost say that we only see true positives and false negatives. However, we can see some orange and green points.

Once that is said, we can see the main changes. In the graphs at positions [1,1] and [1,2], the separation between what the neural network classifies as noise and what it classifies as signal is clearly seen. The graphs representing $|r|$, as we have commented, are now zoomed in the range between 10 and 65. Furthermore, again, as we have already commented in every graphic, the remarkable structures have been reduced.

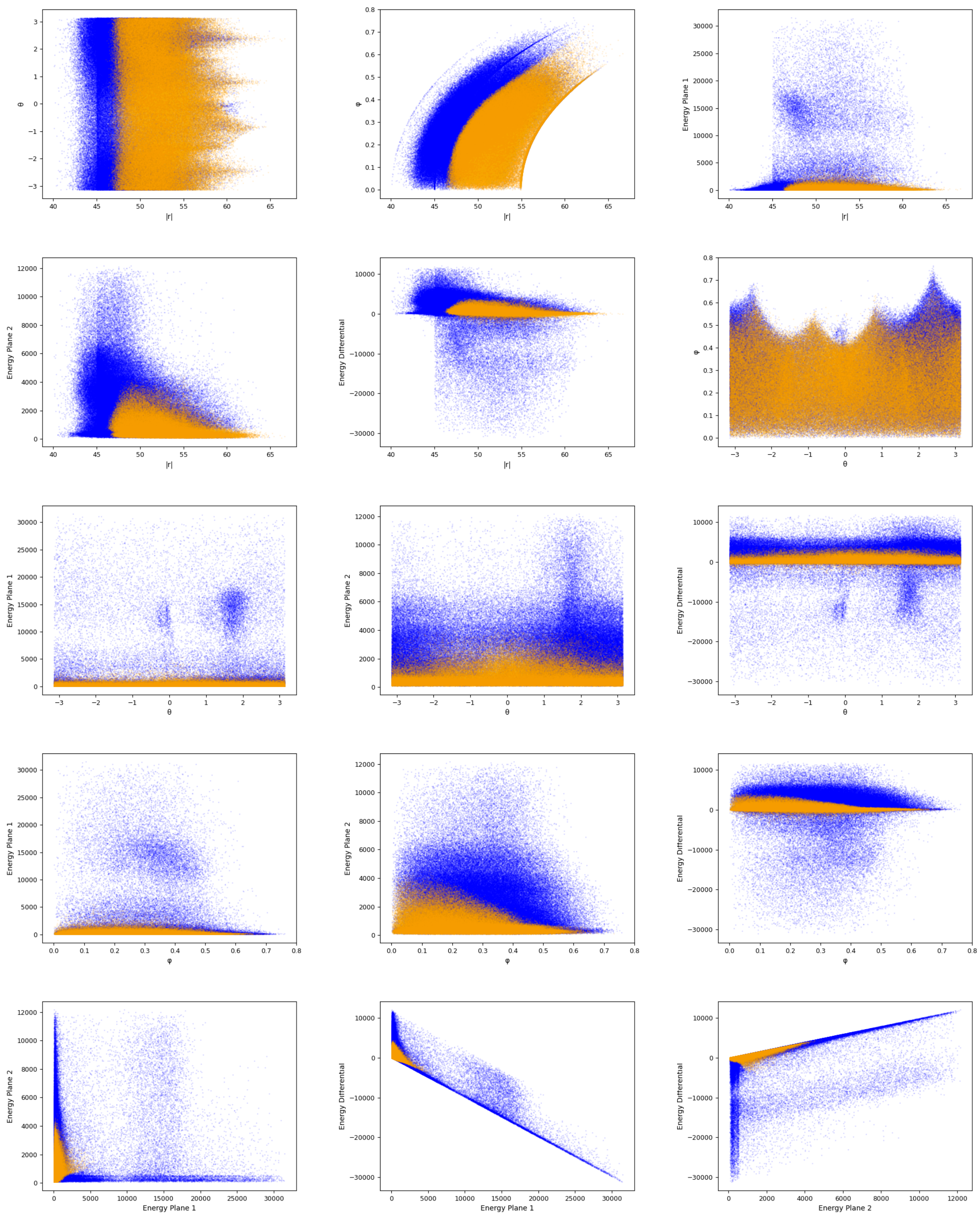


Figure 6.7: Real data classified by the neural network trained with unmasked data into noise (●) and signal (●).

The real data distribution is quite different from the simulated one, both masked and specially unmasked.

In some of these graphs we can clearly see the division of what the neural network classifies as noise and what it classifies as true signal, so in this section that graphs will be briefly commented.

In the graph at position [1,1] it can be clearly seen that the network starts classifying events as true signal for values of $|r|$ higher than approximately 47.5.

The graph at position [1,2] shows a very sharply delineated curve which sets the boundaries of events classified as signal.

The graphs implying energies of different planes on the Y-axis show that only events with low energy are classified as signal. We see a thin line on the bottom of the graphs in which the energy deposited on the plane 1 is displayed in the Y-axis (graphs at positions [1,3], [3,1] and [4,1]), and a bigger zone in the bottom of the graphs which show the energy deposited on plane 2 in the Y-axis (graphs [2,1], [3,2] and [4,2]). This bigger zone seen in the graphs implying the energy deposited on plane 2 is partially caused by the fact that the maximum energy deposited in this plane is lower, so the scale of the Y-axis in these graphs is bigger than the one used in graphs representing the energy deposited in plane 1. This zones have different distributions along the X-axis.

In the graphs representing the energy differential against the module of the vector ([2,2]), the theta angle ([3,3]), and the phi angle ([4,3]) we can see a horizontal line that shows that nearly none of the events with a negative energy differential is classified as signal.

Lastly, the bottom 3 graphs ([5,1], [5,2] and [5,3]) are interesting, since one of the most remarkable structures that appeared on their equivalents which represented simulated data, the V-Shape, has disappeared. However, we still can see that the events classified by the neural network as signal are the ones in the corners where the value of the two axes is relatively low.

6.5 Exploring the use of XGBoost

As it has been seen in Chapter 2, the gradient boosting classifiers are great models for classification. However, the first tries made in this work were implemented with the *GradientBoostingClassifier* of scikit-learn, but the results of these experiments were really bad. This was because, in spite of the fact that the gradient boosting classifiers are good when dealing with class imbalance, the imbalance of the classes in our dataset was too big for it.

Luckily, the XGBoost library for Python offered us many parameters for creating different gradient boosting classifiers. One of these parameters is called *scale_pos_weight* and is used for scaling the gradient of class 1, that is, to make the mistakes made in classifying these events more important, so the model focuses on avoiding them. Using this parameter, we can deal with a class imbalance as big as the one in our dataset.

The first value to consider for this parameter is the number of elements in class 0 divided by the number of elements in class 1. In our dataset this value is 7.572. With this value for *scale_pos_weight* and every other parameter set to default we fitted the first gradient boosting classifier implemented with XGBoost, and the results are presented in Table 6.7

	precision	recall	f1-score
0	0.97	0.71	0.82
1	0.27	0.83	0.41
accuracy			0.72
macro avg	0.62	0.77	0.61
weighted avg	0.89	0.72	0.77

Table 6.7: Results of the gradient boosting classifier, implemented with XGBoost, with *scale_pos_weight* = 7.572 and trained with the whole transformed dataset.

The results on Table 6.7 are great. We can see a recall very close to the one obtained by the IRIS group researchers in the MACACO II project, the best weighted average in the class 1 recall seen until now, and, by far, the best precision in class 1, with a 27%. This is definitely an excellent and well-balanced model.

Some parameters offered by XGBoost have been tweaked from this base to see if some specific parameters can be increased.

First, we tried to increase the recall, as we did with the neural networks. To do this, we tried to increase the *scale_pos_weight* progressively. This obviously reduced other metrics such as the class 1 precision and the weighted average of the recall.

Increasing this value a lot we can achieve the same numbers in class 1 recall as with the neural networks (91%), having better values in the class 1 precision and the weighted average of the recall. We can even achieve a higher class 1 recall (94%), maintaining the same precision of class 1 and weighted average of recall of the neural networks, with values for the *scale_pos_weight* as high as 16. However, even though these solutions are valid if we were only pursuing class 1 recall, they are extremely focused on doing that, while it seems logical to think that a more balanced model will be beneficial.

For this reason, and following the known fact that the best performing values for *scale_pos_weight* are the ones near the number of events in class 0 divided by the number of events in class 1, the highest value of *scale_pos_weight* we will deeply analyze will be 10. This is because, with this value, we achieved very high values of class 1 recall, near the ones obtained with neural networks, but with much more precision of class 1 and weighted average of the recall. The results of this model are presented in Table 6.8.

	precision	recall	f1-score
0	0.98	0.65	0.78
1	0.25	0.89	0.39
accuracy			0.68
macro avg	0.61	0.77	0.58
weighted avg	0.89	0.68	0.73

Table 6.8: Results of the gradient boosting classifier, implemented with XGBoost, with $scale\_pos\_weight = 10$ and trained with the whole transformed dataset.

In this model we have used the default parameters, except for the *scale_pos_weight*, which has taken a value of 10. With this configuration, a recall of 89% is achieved for class 1. However, this is not the only excellent metric. Precision for class 1 of 25% is achieved, together with a weighted average for the recall of 0.68, which despite not being very high, is way better than the ones obtained by any of the implemented neural networks with the whole transformed dataset.

It is also interesting to see what we can achieve if we change our approach radically. Until now, we have tried to increase the recall without taking into account the consequences for the other metrics (as crucial as the precision of class 1 and the weighted average of the recall). Then, the opposite approach is to try to increase these two metrics even if the recall decreases a bit. To do this, we will decrease the *scale_pos_weight*. However, as the recall of class 1, as we have said, is very important, we will only do this slightly.

We established a maximum of 10% reduction of recall compared to the one obtained in MACACO II. To comply with this maximum, we used a $scale\_pos\_weight = 6$. Moreover, we tweaked some more parameters. We set the *max_depth* to 10 and the *n_estimators* to 200. This configuration yielded the results shown in Table 6.9

	precision	recall	f1-score
0	0.96	0.78	0.86
1	0.31	0.74	0.43
accuracy			0.77
macro avg	0.63	0.76	0.65
weighted avg	0.88	0.77	0.81

Table 6.9: Results of the gradient boosting classifier, implemented with XGBoost, with *scale_pos_weight* = 6, *max_depth* = 10, *n_estimators* = 200 and trained with the whole transformed dataset.

With this configuration, we achieved an impressive precision of class 1 of 31%. This means that we multiplied by three the precision of class 1 obtained in the MACACO II project. Moreover, we achieved an outstanding weighted average of recall of 77%.

These results are excellent. However, a 10% decrease in class 1 recall could be too much. For this reason, the next and last step in the experimentation with XGBoost was to create a balanced model, maintaining the recall of the MACACO II experiment or even increasing it a bit while increasing the precision of class 1 and the weighted average of recall significantly.

To achieve this we created a gradient boosting classifier with *scale_pos_weight* = 9, *max_depth* = 9, and every other parameter set to default, and that model generated the results shown in Table 6.10

	precision	recall	f1-score
0	0.97	0.69	0.80
1	0.27	0.86	0.41
accuracy			0.71
macro avg	0.62	0.77	0.61
weighted avg	0.89	0.71	0.76

Table 6.10: Results of the gradient boosting classifier, implemented with XGBoost, with *scale_pos_weight* = 9, *max_depth* = 9 and trained with the whole transformed dataset.

This model definitely has very balanced results, with a recall 2 points higher than the one obtained in the MACACO II project by the researchers of IRIS group. Also, we obtain one of the highest values in the precision of class 1 of all the results obtained during this work and a very good weighted average of recall, being over 70%, exactly 71%.

If we create the scatter plots for the analysis of the model classification for the models implemented using XGBoost we can see the most remarkable differences in the ones representing real data.

If we compare these graphs with the ones representing real data filtered by the 3-layer neural network here, the separation of events classified as noise and the events classified as signal is much clearer in the neural network classification.

The clearest examples of this are the graphs in positions [1,1] and [1,2]. In these graphs, an obvious separation line was seen in the classification performed by the neural network. However, here, in Figure 6.8, the orange dots cover the majority of the populated area of the graph.

In the majority of other graphs, despite having a more clear separation, we can also see that the zones covered by events classified as signal have significantly expanded.

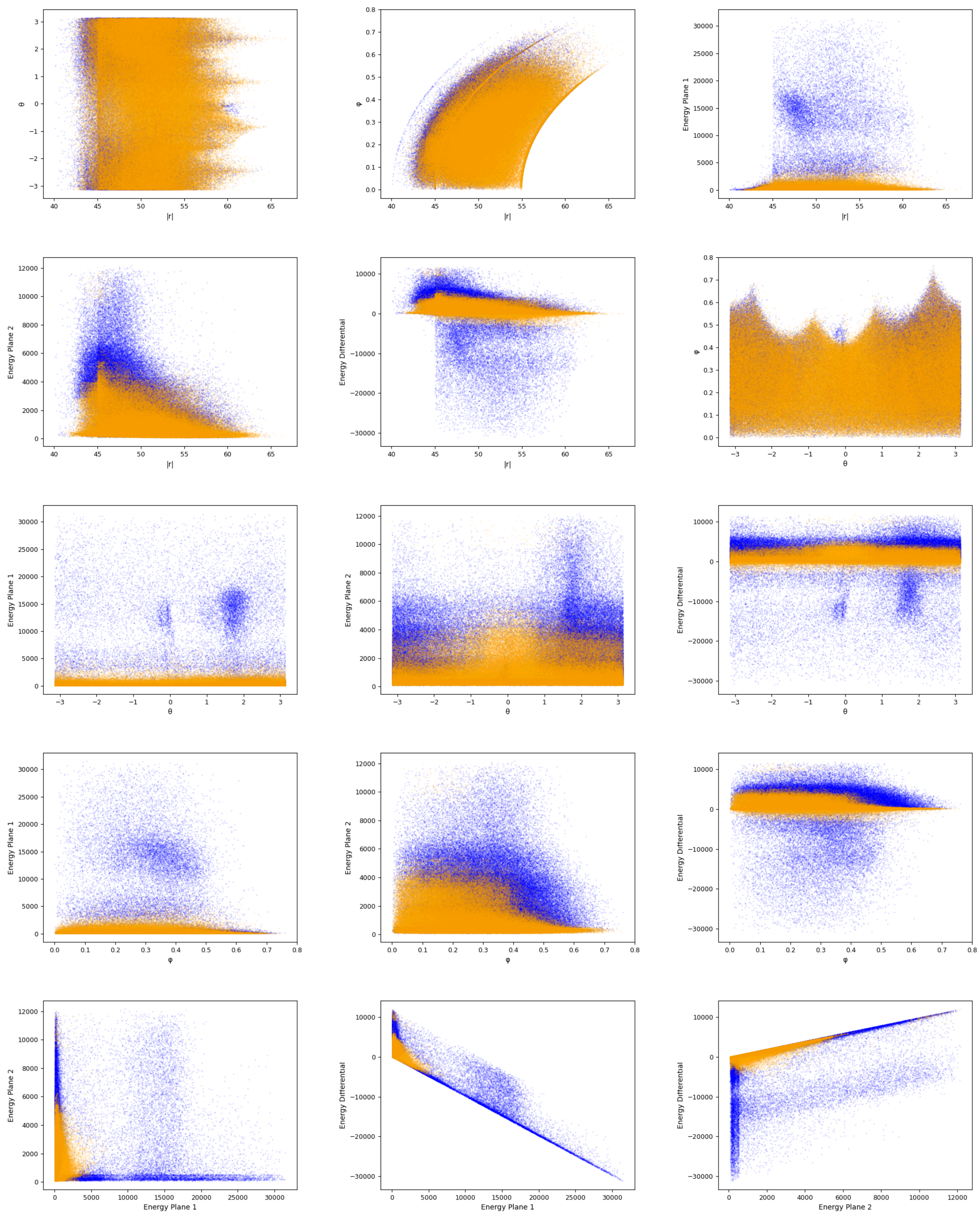


Figure 6.8: Real data classified by the XGBoost model with *scale_pos_weight* = 9 and *max_depth* = 9 into noise (●) and signal (●).

Moreover, it is interesting to see that in the graphs presented in Figure 6.8, many of the frontiers of separation are formed by straight horizontal and vertical lines, which did not appear in the graphs showing the classification of the neural network. An extremely clear example of this can be seen in the graph at position [2,1] of Figure 6.8, there, we see a vertical line and a horizontal one intersecting near (45, 4000).

Many other graphs have been created to analyze the behavior of the implemented model. However, analyzing every one of them will be extremely tedious and inefficient since the majority of them, despite having remarkable differences look similar. For the interested reader, a big set of these graphs is presented in Appendix A.

6.6 Image reconstruction

This section has been elaborated in collaboration with the IRIS group, which has performed the image reconstruction.

There is no doubt that when using numeric metrics, such as the precision or the recall of class 1, our neural networks have improved the results of the MACACO II paper [10]. However, it is also important to analyze the effect of using the models implemented in this work on the reconstructed images after event selection. First Figure 6.9 will be analyzed.

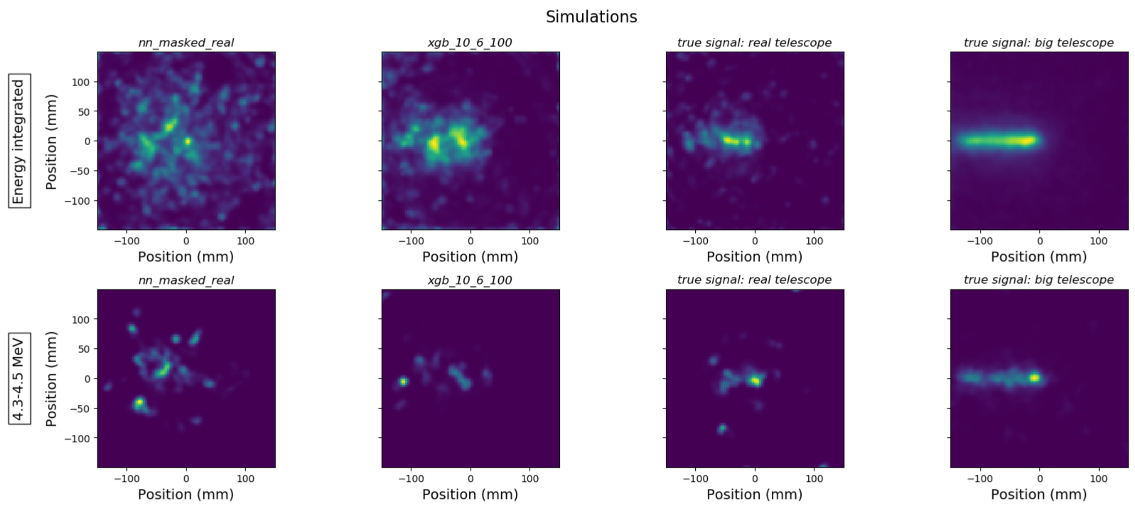


Figure 6.9: Image reconstruction of simulated data filtered by our models, compared to the simulated masked and unmasked signal.

In Figure 6.9 we see, in the top row, the reconstructed images from simulations, including all photon energies (energy integrated), and in the bottom row reconstructed images selecting only photons with energies in the range 4.3-4.5 MeV.

Let us start analyzing both rows simultaneously, from the right, where the image reconstructed from the pure true signal (only good events) simulated data is located. To its left, we can see the image reconstructed with the pure true signal masked data. In this case, we see a worse result, caused mainly due to the low quantity of available events. In the third column from the right, we can see the reconstructed images from one of the gradient boosting classifiers implemented with XGBoost, specifically the one with $scale\_pos\_weight = 10$ and the other parameters set to default. We don't see a perfect result in this image, again caused partially by the low number of available events. To finalize the analysis of the image, the first column from the left show the images reconstructed which shows a poorer result, suggesting that the training of models with simulated masked data is not ideal.

Figure 6.10 shows images reconstructed with real data compared to the image obtained from the simulation with only good events (in the last column). From left to right:

- The image reconstructed from data classified by the NN trained with masked data.
- The image reconstructed from data classified by the NN trained with unmasked data.
- The image reconstructed from data classified by the XGBoost model with $scale\_pos\_weight = 10$ and the other parameters set to default.

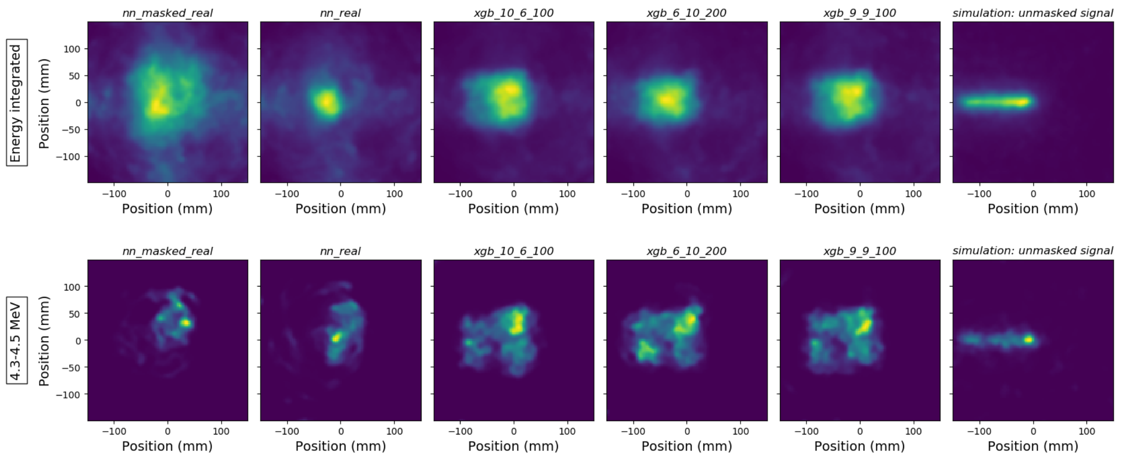


Figure 6.10: Image reconstruction of real data filtered by our models, compared to the simulated unmasked signal.

- The image reconstructed from data classified by the XGBoost model with $scale\_pos\_weight = 6$, $max\_depth = 10$, $n\_estimators = 200$ and the other parameters set to default.
- The image reconstructed from data classified by the XGBoost model with $scale\_pos\_weight = 9$, $max\_depth = 9$ and the other parameters set to default.
- The reconstructed image with the whole pure true signal simulated data.

By looking at them, we can summarize the analysis in two main observations:

In the first place, we can see that, when observing the two differently trained neural networks, the one trained with unmasked data seems to perform better than the one trained with the data masked to the size of the real Compton Camera.

In the second place, we can clearly state that the models implemented using XGBoost perform better than the neural networks, generating a distribution more similar to the simulated one, despite being different.

CHAPTER 7

Conclusions

In this chapter, the different parts of this work will be analyzed in order to extract conclusions. We will also review the goals set in Section 1.2 to see whether they have been reached and how. Additionally, some potential future work will be proposed taking these conclusions as a base.

This section can be summarized by claiming that the two goals of this work have been achieved. These were to create one or more Machine Learning classification models able to separate Compton Camera detected events into true signal and background noise, and to improve the results obtained by the IFIC with the neural network presented in the MACACO II paper[10]. Let us further analyze the achievement of each of these goals.

The first goal "Create a neural network or classifier capable of classifying events detected by a Compton Camera as true signal or background noise, trained with Monte Carlo simulations", was easily achieved, creating not one but multiple classifiers able to perform this task in an effective way. From these created classifiers, the most relevant ones were presented in this memory and are the following 10:

- The neural network implemented for the MACACO II project, adapted and trained with the whole transformed dataset. (Section 6.2)
- The neural network implemented for the MACACO II project, adapted and trained with the masked transformed dataset. (Section 6.2)
- The 2-layer neural network trained with the whole transformed dataset. (Subsection 6.3.1)
- The 2-layer neural network trained with the masked transformed dataset. (Subsection 6.3.1)
- The 3-layer neural network with Batch Normalization trained with the whole transformed dataset. (Subsection 6.3.2)
- The 3-layer neural network with Batch Normalization trained with the masked transformed dataset. (Subsection 6.3.2)
- The gradient boosting classifier, implemented with XGBoost, with *scale_pos_weight* = 7.572 and trained with the whole transformed dataset. (Section 6.5)
- The gradient boosting classifier, implemented with XGBoost, with *scale_pos_weight* = 10 and trained with the whole transformed dataset. (Section 6.5)
- The gradient boosting classifier, implemented with XGBoost, with scale_pos_weight = 6, max_depth = 10, n_estimators = 200 and trained with the whole transformed dataset. (Section 6.5)

- The gradient boosting classifier, implemented with XGBoost, with scale_pos_weight = 9, max_depth = 9 and trained with the whole trans- formed dataset. (Section 6.5)

The second goal was also achieved. Even though the image reconstruction is not perfect and can possibly be improved in future works by analyzing it and increasing the number of events in the dataset, we can certainly claim that we have quantifiably improved the results presented in the MACACO II paper. [10]

This paper presented only two metrics to evaluate the implemented neural network. These metrics were the precision and the recall of class 1. In all the models presented in this work, at least one of these metrics has been improved. In fact, in six of the ten models, both metrics have improved significantly, or one of them has improved significantly while the other stayed the same. There are four exceptions where the class 1 recall has slightly decreased; however, in three cases (seen in tables 6.2, 6.4 and 6.7), these models were the first or second tries of a new approach, and, as so, were taken as a base to create others with both the precision and recall of class 1 better than the ones obtained by the MACACO II neural network. In the other case, the class 1 recall decreased because we were trying to maximize the class 1 precision; this goal was achieved by getting the fantastic result of multiplying by three this metric (as seen in Table 6.9).

In summary, all the goals set for this work were achieved by implementing multiple classifiers able to separate Compton Camera events detected while applying Hadron Therapy into signal or noise, which improved the results of the IFIC MACACO II paper.

Bibliography

[1] The International Agency for Research on Cancer. Global Cancer Observatory. https://gco.iarc.fr/, last access: July 02, 2022.

[2] Robert R. Wilson. Radiological Use of Fast Protons. *Radiology*, 47(5):487–491, November 1946.

[3] Manjit Dosanjh, Ugo Amaldi, Ramona Mayer, and Richard Poetter. ENLIGHT: European network for Light ion hadron therapy. *Radiotherapy and Oncology*, 128(1):76–82, July 2018.

[4] Particle Therapy Co-Operative Group. Particle therapy: Pacients and facilities statistics. https://www.ptcog.ch/index.php, last access: July 02, 2022.

[5] Harald Paganetti and Hanne Kooy. Proton radiation in the management of localized cancer. *Expert Review of Medical Devices*, 7(2):275–285, March 2010.

[6] Jaap D. Zindler, Charles R. Thomas, Jr., Stephen M. Hahn, Aswin L. Hoffmann, Esther G.C. Troost, and Philippe Lambin. Increasing the Therapeutic Ratio of Stereotactic Ablative Radiotherapy by Individualized Isotoxic Dose Prescription. *JNCI: Journal of the National Cancer Institute*, 108(2):djv305, February 2016.

[7] International Commission on Radiation Units and Measurements. Prescribing, Recording, and Reporting Proton-Beam Therapy (ICRU Report 78). *2*, 7, December 2007.

[8] Katja Langen and Minesh Mehta. Proton Beam Therapy Basics. *Journal of the American College of Radiology*, 12(11):1204–1206, November 2015.

[9] Harald Paganetti. Range uncertainties in proton therapy and the role of Monte Carlo simulations. *Physics in Medicine and Biology*, 57(11):R99–117, June 2012.

[10] Enrique Muñoz, Ana Ros, Marina Borja-Lloret, John Barrio, Peter Dendooven, Josep F. Oliver, Ikechi Ozoemelam, Jorge Roser, and Gabriela Llosá. Proton range verification with MACACO II Compton camera enhanced by a neural network for event selection. *Scientific Reports*, 11(1):9325, April 2021.

[11] M. Moteabbed, S. España, and H. Paganetti. Monte Carlo patient study on the comparison of prompt gamma and PET imaging for range verification in proton therapy. *Physics in Medicine and Biology*, 56(4):1063–1082, January 2011.

[12] Chul-Hee Min, Chan Hyeong Kim, Min-Young Youn, and Jong-Won Kim. Prompt gamma measurements for locating the dose falloff region in the proton therapy. *Applied Physics Letters*, 89(18):183517, October 2006.

[13] J. Kiener, N. de Sereville, and V. Tatischeff. Shape of the 4.438 MeV gamma-ray line of ^12C from proton and alpha-particle induced reactions on ^12C and ^16O. *Physical Review C*, 64(2):025803, June 2001.

[14] J. Krimmer, D. Dauvergne, J. M. Létang, and É. Testa. Prompt-gamma monitoring in hadrontherapy: A review. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 878:58–73, January 2018.

[15] Chongsheng Zhang, Changchang Liu, Xiangliang Zhang, and George Almpanidis. An up-to-date comparison of state-of-the-art classification algorithms. *Expert Systems with Applications*, 82:128–150, October 2017.

[16] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, September 1995.

[17] Leo Breiman. Random Forests. *Machine Learning*, 45(1):5–32, October 2001.

[18] Jerry Ye, Jyh-Herng Chow, Jiang Chen, and Zhaohui Zheng. Stochastic gradient boosted distributed decision trees. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management*, CIKM '09, pages 2061–2064, New York, NY, USA, November 2009. Association for Computing Machinery.

[19] Jerome H. Friedman. Stochastic gradient boosting. *Computational Statistics & Data Analysis*, 38(4):367–378, February 2002.

[20] Martin Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. TensorFlow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pages 265–283, 2016.

[21] Francois Chollet. Keras, 2015. https://keras.io/, last access: July 02, 2022.

[22] The Team, Rami Al-Rfou, Guillaume Alain, Amjad Almahairi, Christof Angermueller, Dzmitry Bahdanau, Nicolas Ballas, Frederic Bastien, Justin Bayer, Anatoly Belikov, et al. Theano: A Python framework for fast computation of mathematical expressions. May 2016.

[23] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12(85):2825–2830, 2011.

[24] John D. Hunter. Matplotlib: A 2D Graphics Environment. *Computing in Science & Engineering*, 9(3):90–95, May 2007.

[25] Tianqi Chen and Carlos Guestrin. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794, August 2016.

[26] Lutz Roeder. Netron, Visualizer for neural network, deep learning, and machine learning models, 12 2017.

[27] S. Jan, G. Santin, D. Strul, S. Staelens, K. Assié, D. Autret, S. Avner, R. Barbier, M. Bardiès, P. M. Bloomfield, et al. GATE: A simulation toolkit for PET and SPECT. *Physics in Medicine and Biology*, 49(19):4543–4561, October 2004.

[28] S. Agostinelli, J. Allison, K. Amako, J. Apostolakis, H. Araujo, P. Arce, M. Asai, D. Axen, S. Banerjee, G. Barrand, et al. Geant4—a simulation toolkit. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 506(3):250–303, July 2003.

APPENDIX A

Scatterplots for the analysis of the classification performed by the different models

As we have already seen in Chapter 6 a very interesting thing to do when implementing different classification models, to understand its behavior, is to analyze the output using different plots. In this work, this has been done using scatter plots.

However, in Chapter 6 we only presented 4 sets of graphs. This is because every set of graphs is pretty similar and, once a certain point of understanding is reached, further analysis will yield very little knowledge.

Despite this fact, every set of graphs has differences that, despite looking negligible to the untrained eye when looking at the graphs, show that different models understand the data in a significantly different way. For this reason, in this section, other generated sets of scatter plots, which were not finally presented in Chapter 6, are displayed, so these differences can be observed.

The sets of graphs shown in this Appendix are:

- The three graphs corresponding to unmasked simulated data, masked simulated data and real data classified by the NN trained with the masked transformed data. (Respectively figures A.1, A.2 and A.3).
- The graph showing unmasked simulated data classified by the XGBoost model with $scale\_pos\_weight = 9$ and $max\_depth = 9$. (Figure A.4).
- The two graphs showing unmasked simulated data and real data classified by the XGBoost model with $scale\_pos\_weight = 10$. (Respectively figures A.5 and A.6).
- The two graphs showing unmasked simulated data and real data classified by the XGBoost model with $scale\_pos\_weight = 6$, $max\_depth = 10$ and $n\_estimators = 200$. (Respectively figures A.7 and A.8).

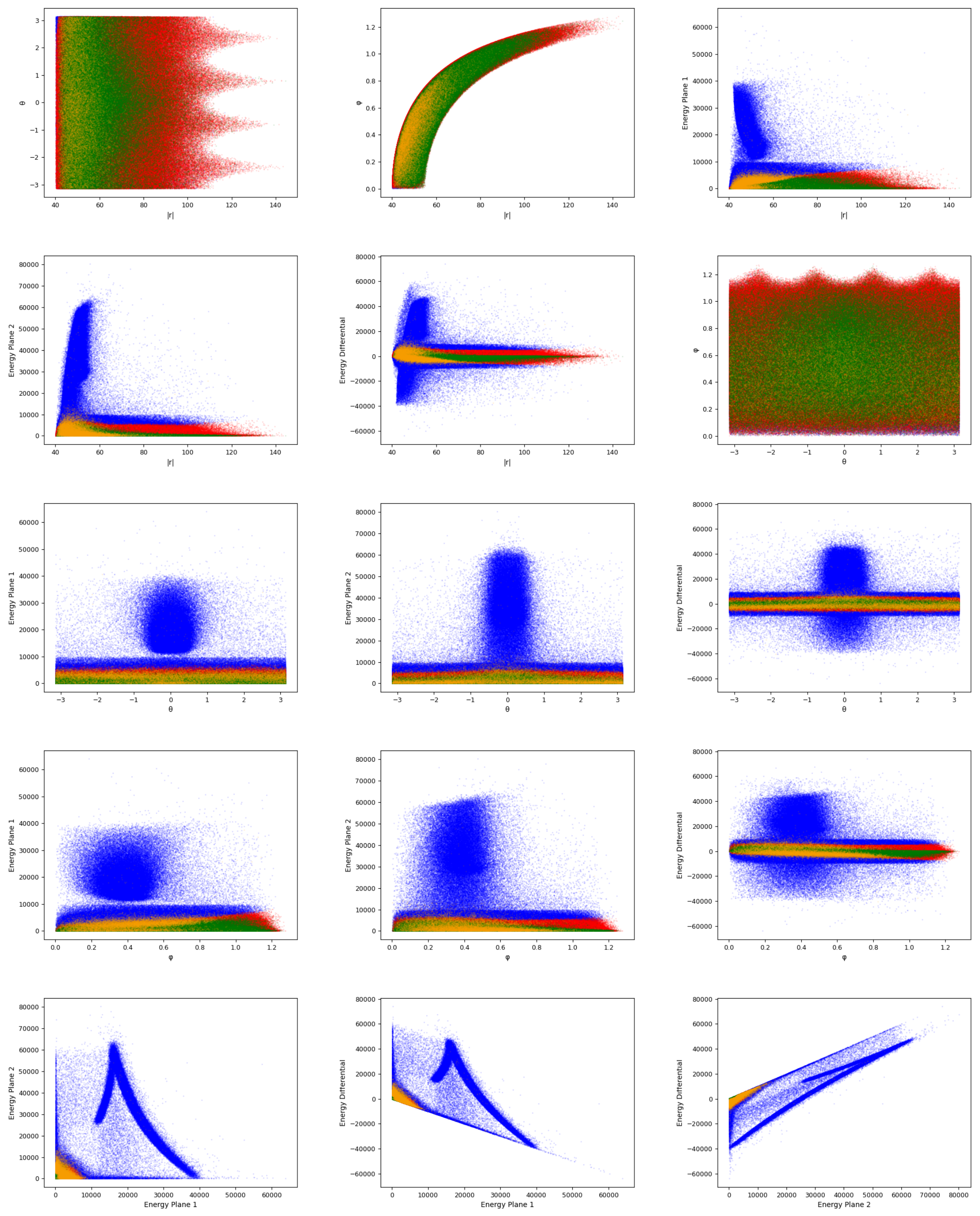


Figure A.1: Unmasked simulated data classified by the 3 hidden layer NN trained with masked simulated data into true negatives (●), false positives (●), true positives (●), and false negatives (●).

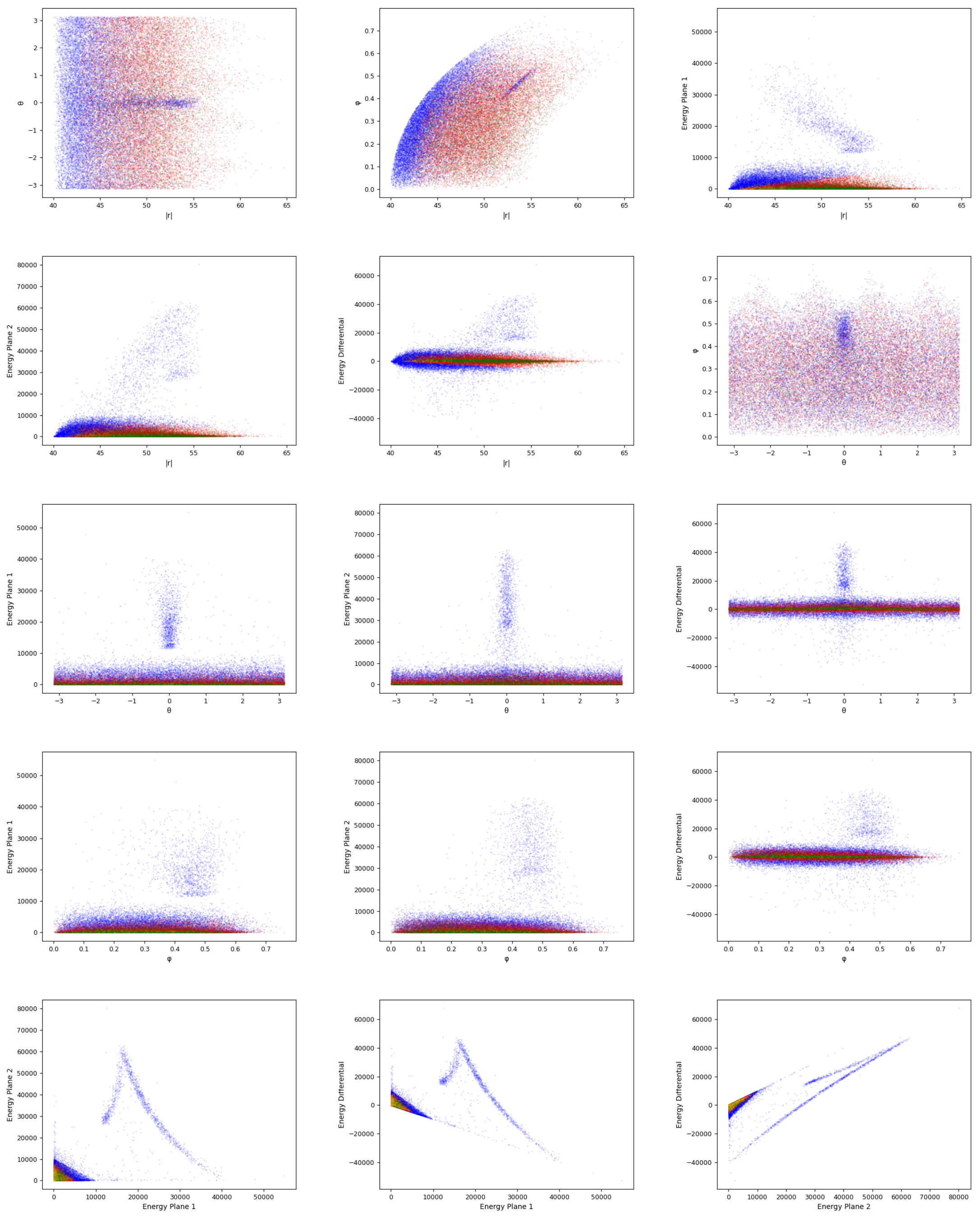


Figure A.2: Masked simulated data classified by the 3 hidden layer NN trained with masked simulated data into true negatives (●), false positives (●), true positives (●), and false negatives (●).

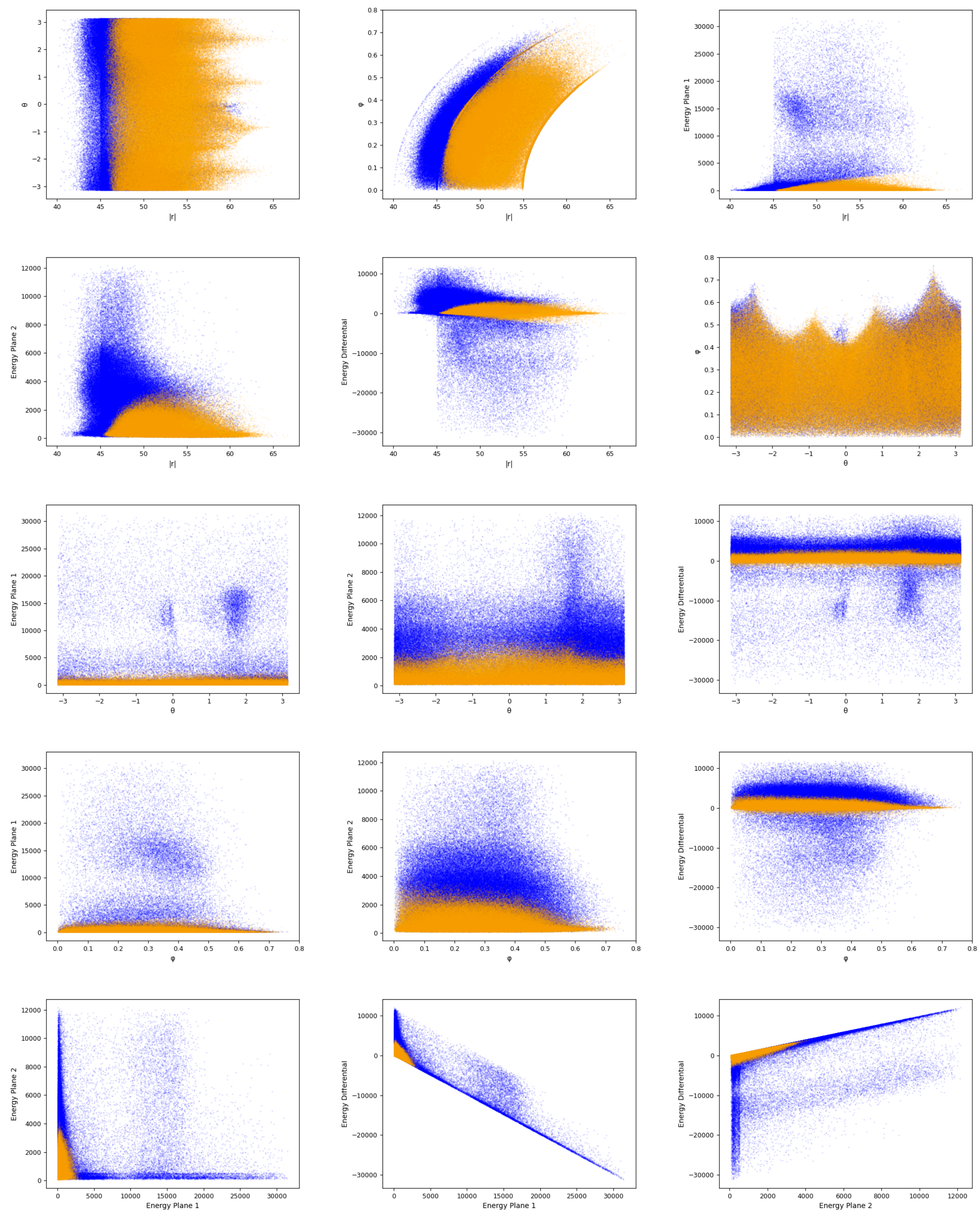


Figure A.3: Real data classified by the 3 hidden layer NN trained with masked simulated data into noise (●) and signal (●).

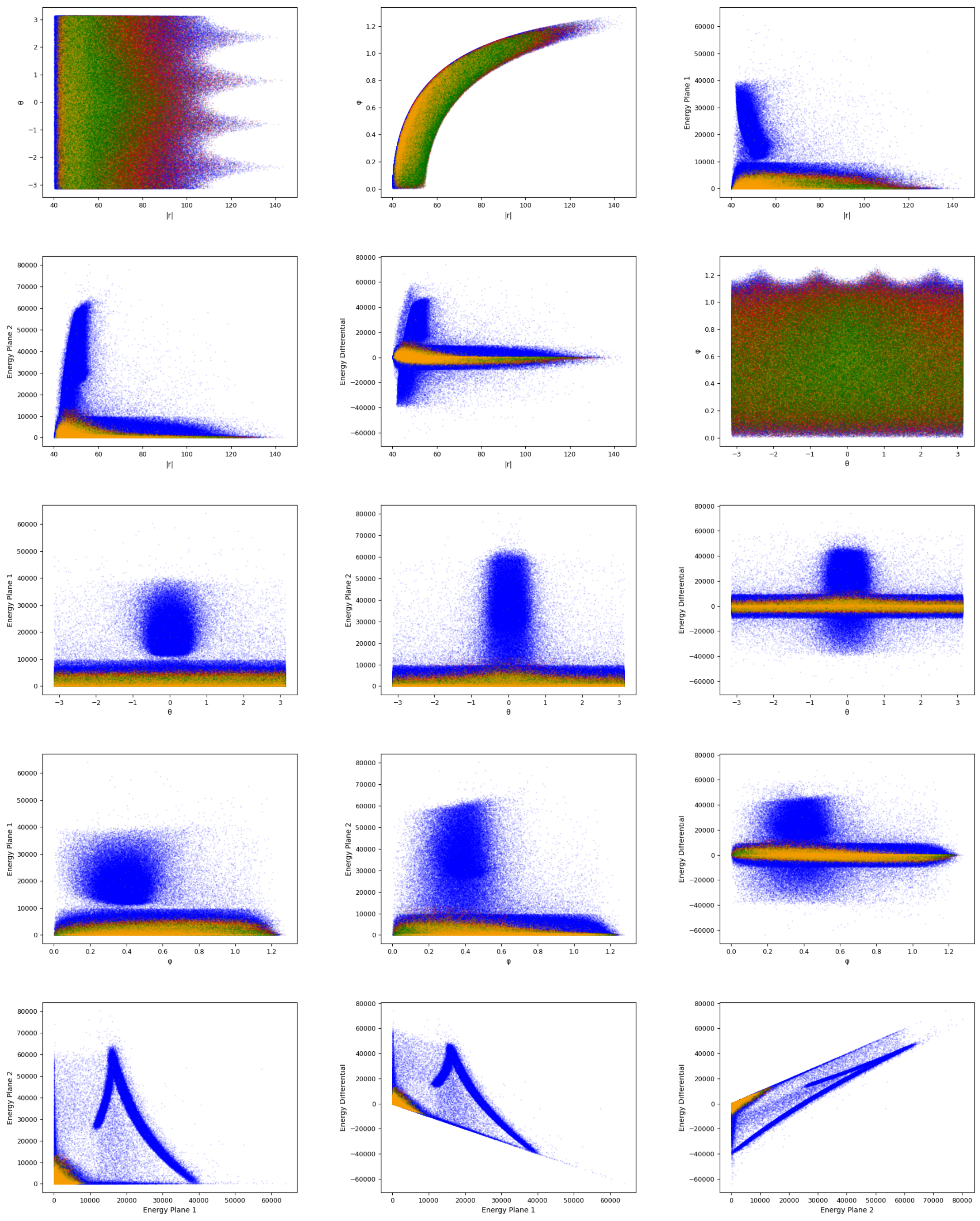


Figure A.4: Unmasked simulated data classified by the XGBoost model with $scale\_pos\_weight = 9$ and $max\_depth = 9$ into true negatives (●), false positives (●), true positives (●), and false negatives (●).

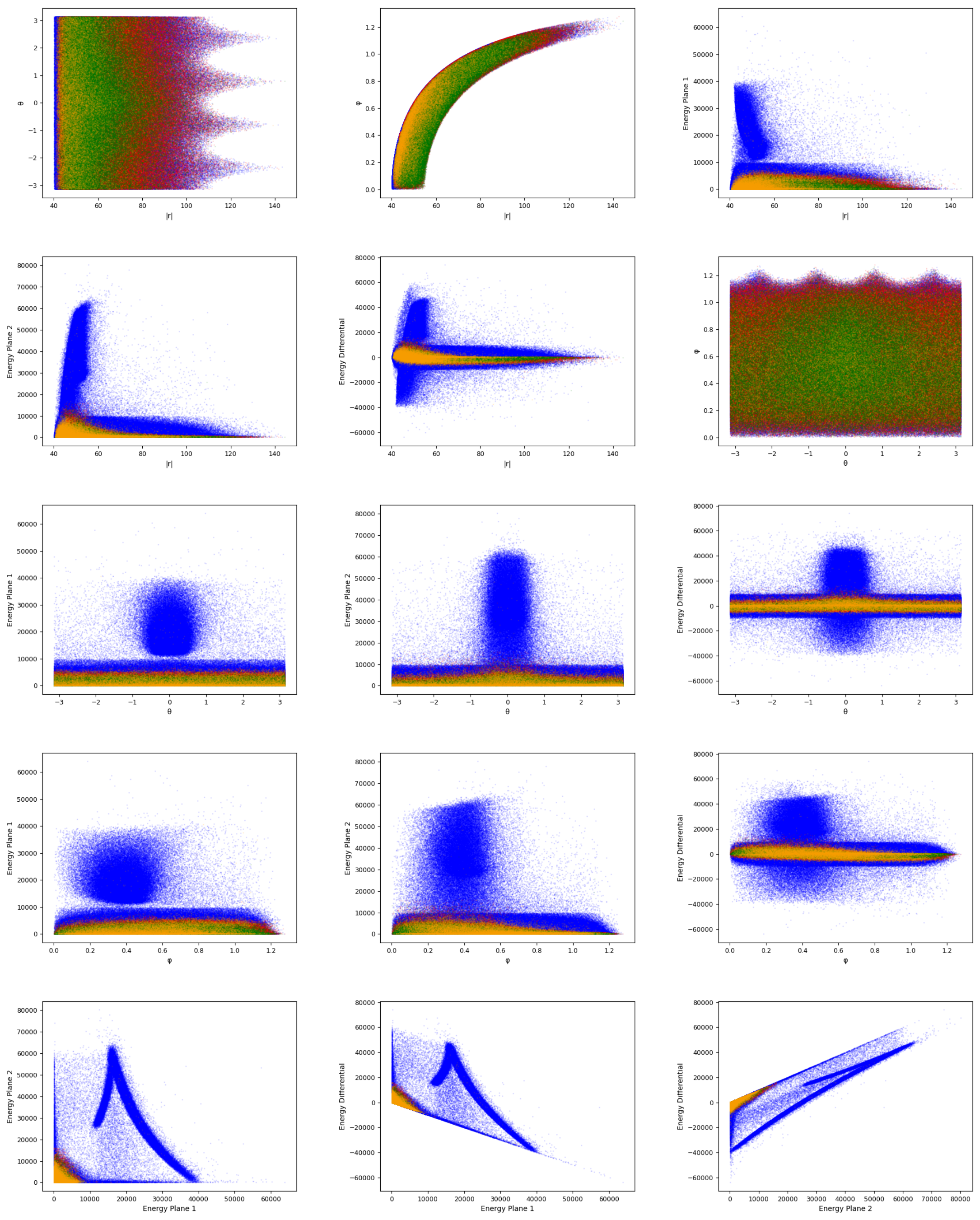


Figure A.5: Unmasked simulated data classified by the XGBoost model with *scale_pos_weight* = 10 into true negatives (●), false positives (●), true positives (●), and false negatives (●).

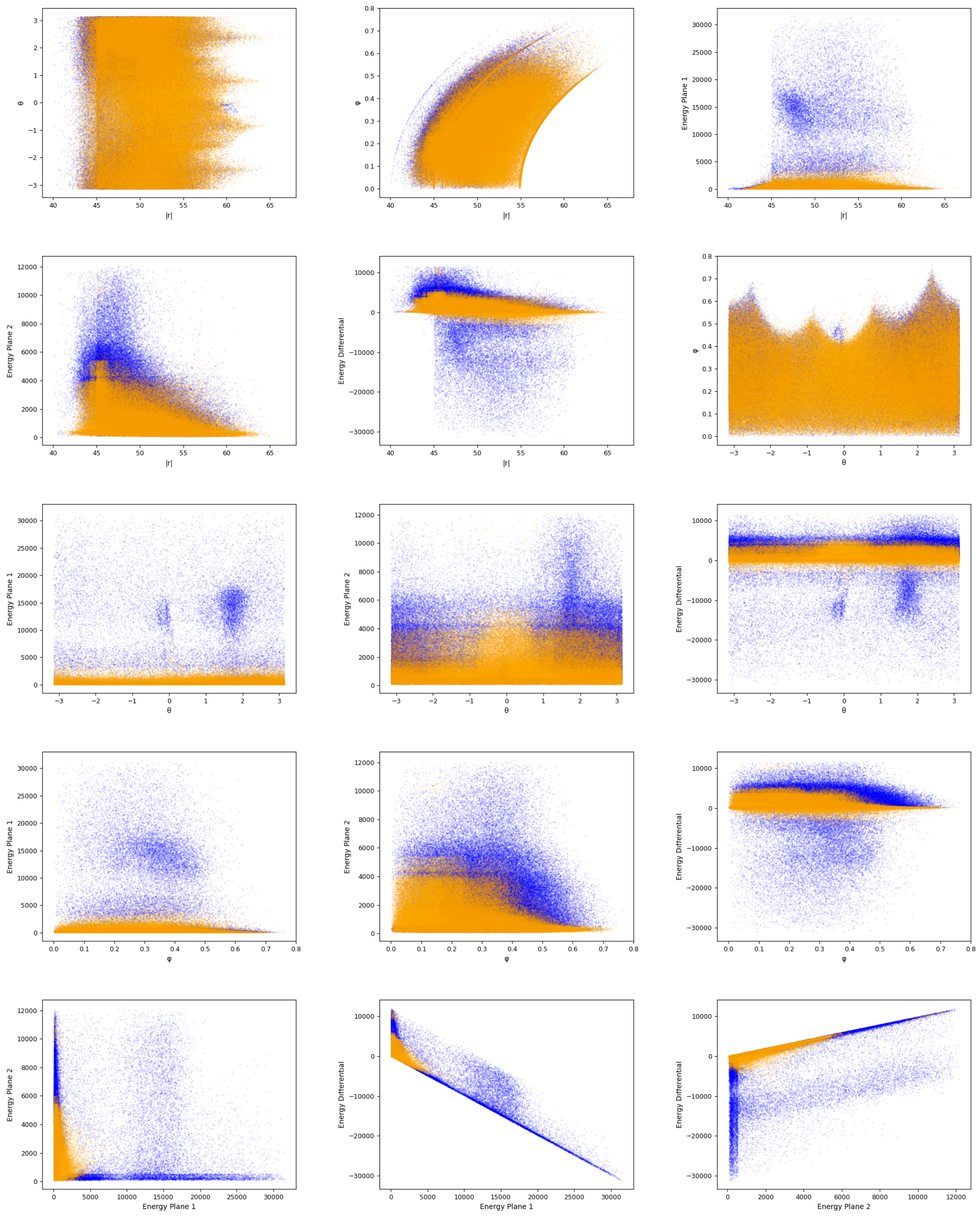


Figure A.6: Real data classified by the XGBoost model with *scale_pos_weight* = 10 into noise (●) and signal (●).

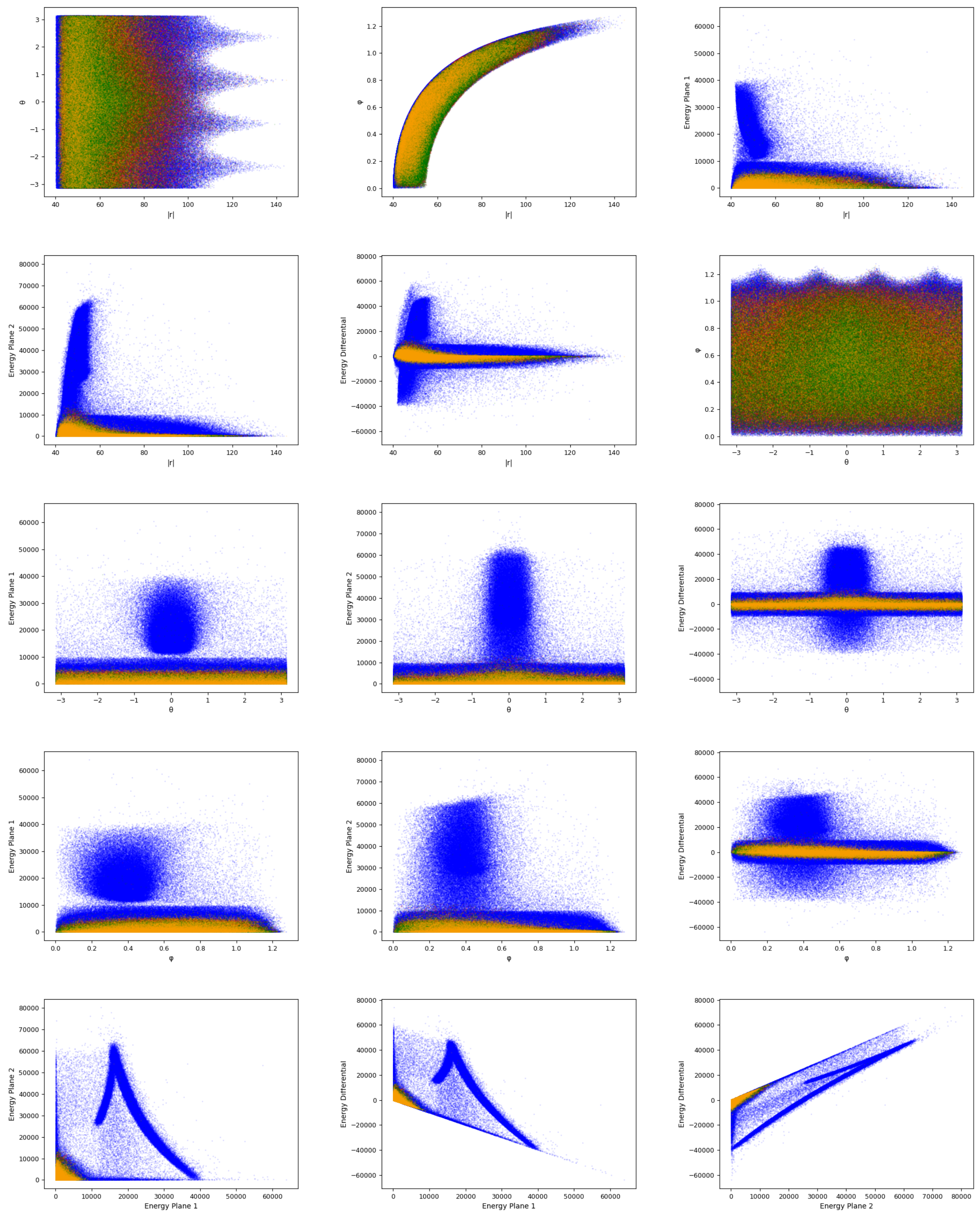


Figure A.7: Unmasked simulated data classified by the XGBoost model with $scale\_pos\_weight = 6$, $max\_depth = 10$ and $n\_estimators = 200$ into true negatives (●), false positives (●), true positives (●), and false negatives (●).

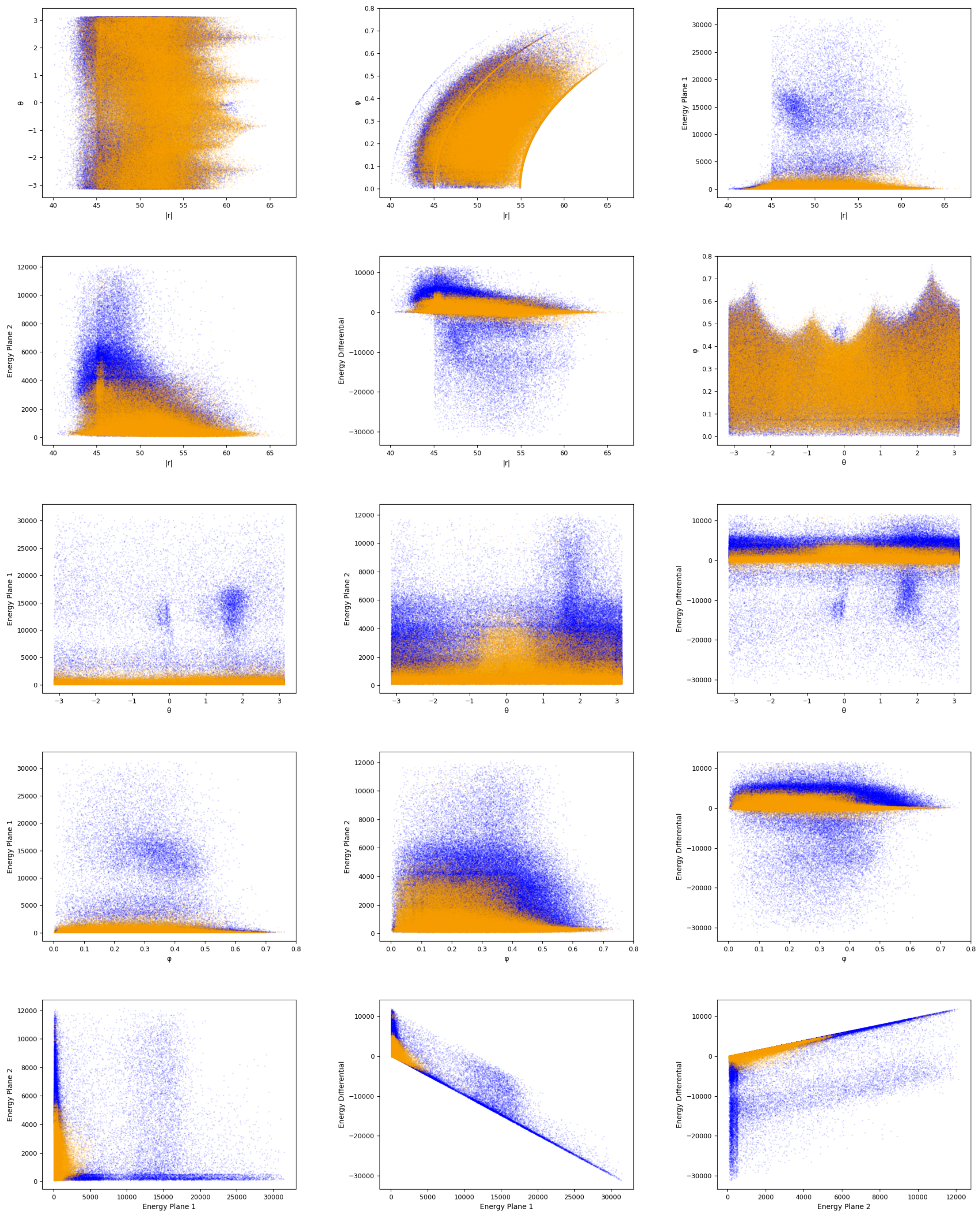


Figure A.8: Real data classified by the XGBoost model with $scale\_pos\_weight = 6$, $max\_depth = 10$ and $n\_estimators = 200$ into noise (●) and signal (●).

APPENDIX B

Sustainable Development Goals

Sustainable Development Goals (SDGs)	**High**	**Medium**	**Low**	**Not applicable**
SDG 1. **No Poverty.**				✓
SDG 2. **Zero Hunger.**				✓
SDG 3. **Good Health and Well-Being.**	✓			
SDG 4. **Quality Education.**				✓
SDG 5. **Gender Equality.**				✓
SDG 6. **Clean Water and Sanitation.**				✓
SDG 7. **Affordable and Clean Energy.**				✓
SDG 8. **Decent Work and Economic Growth.**				✓
SDG 9. **Industry, Innovation, and Infrastructure.**				✓
SDG 10. **Reduced Inequalities.**				✓
SDG 11. **Sustainable Cities and Communities.**				✓
SDG 12. **Responsible Consumption and Production.**				✓
SDG 13. **Climate Action.**				✓
SDG 14. **Life Below Water.**				✓
SDG 15. **Life on Land.**				✓
SDG 16. **Peace, Justice and Strong Institutions.**				✓
SDG 17. **Partnerships for the Goals.**	✓			

As it was explained in the Introduction of this work, cancer is one of the leading causes of death nowadays. This assertion is based on data provided by the International Agency for Research on Cancer (IARC) of the World Health Organization. The statistics published by this institute show that in 2019 more than 19 million people were diagnosed with cancer, and nearly ten million people died because of cancer. [1]

Moreover, the prediction by this same agency is not hopeful. According to IARC, in 2040, with the projections of demographic change, the statistics will get worse, surpassing the thirty million new cancer cases that year.

With these predictions, the rapid development of methods to fight cancer is crucial. This can be done in many ways, for instance, by improving its early detection, creating new therapies and improving the ones which are already in use, or creating methods to reduce the suffering that people with cancer have to experiment. Researchers around the world are working in all this field. This work focuses on improving one of the most promising cancer treatments, which use has been rapidly increasing these last years, hadron therapy.

Hadron therapy is a kind of radiotherapy, cancer treatment using radiation. However, it has many differences if we compare it to the most used form of radiotherapy nowadays, which is photon therapy. The main advantage that hadron therapy has over photon therapy is that due to the depth-deposition curve of the particles used in hadron therapy (which shows the dose of radiation deposited at each point of its trajectory) presents the Bragg Peak. The Bragg Peak is, as its name suggests, a high and narrow peak at the end of the particles' trajectory where a lot of radiation is deposited.

The existence of the Bragg Peak allows this treatment to deposit more dose on the tumor and less on the surrounding healthy tissues and organs. This can be highly relevant in cases where the tumor is close to critical structures.

However, the Bragg Peak is a double-edged sword. If applied correctly, it will damage more of the tumor, and fewer tissues, as we commented, but any imprecision when applying the therapy could lead to the Bragg Peak hitting the wrong spot.

To avoid these possibilities, nowadays, broad safety margins are being applied. Nevertheless, this limits the applicability of hadron therapy a lot.

The reduction of these safety margins is crucial, and to achieve it online monitoring must be applied. There are many ways of performing online monitoring. One of them implies detecting prompt gamma-rays with a Compton Camera.

This method has a drawback. Compton Cameras not only detect the signal produced by hadron therapy but also a lot of background noise, which makes the reconstruction of the images more difficult and leads to poorer results.

In this work, we deal with this fact by training different Machine Learning models to separate true signal from background noise.

For all these reasons, we can conclude that this work strongly complies with the "Good Health and Well-Being" Sustainable Development Goal, corresponding to the number three.

The other Sustainable Development Goal which this work materializes is the number 17: "Partnerships for the Goals.". This can be claimed since the base of the work is a partnership with the IRIS group of the IFIC (CSIC), which stated the problem and provided the dataset generated by their previous projects.