TRABAJO FIN DE MÁSTER

MÁSTER UNIVERSITARIO OFICIAL EN CIENCIA DE DATOS E INGENIERÍA DE COMPUTADORES

Classification of Ultra-High Energy Cosmic Rays

Autor
Francisco Carrillo Pérez

Directores
Luis Javier Herrera Maldonado
Alberto Guillén Perales

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

—

Granada, Julio de 2019

Classification of Ultra-High Energy Cosmic Rays

Autor
Francisco Carrillo Pérez

Directores
Luis Javier Herrera Maldonado
Alberto Guillén Perales

Clasificación de Rayos Cósmicos de Ultra Alta Energía

Francisco Carrillo Pérez

Palabras clave: Rayos Cósmicos, Ultra Alta Energía, Redes Neuronales Convolucionales, Máquinas de Vector Soporte, Aprendizaje Profundo

Resumen

Los algoritmos de aprendizaje automático han resultado útiles en una gran variedad de campos, tales como la medicina, la visión por computador o la automatización industrial. Específicamente en el campo de la astrofísica en los últimos años, estos algoritmos han ayudado a acelerar nuestra compresión del Universo y de la interacción entre las particulas. Dentro del aprendizaje automático, los algoritmos de aprendizaje profundo han mostrado unos resultados increíbles en problemas los cuáles presentan una dependencia crucial a características espaciales, como por ejemplo las imágenes o series temporales. Los rayos cósmicos son radiaciones de alta energía, principalmente originadas fuera de nuestro Sistema Solar o incluso en galaxias lejanas, costituyen un problema fascinante para la física de hoy en día. Cuando un Rayo Cósmico de Ultra Alta Energía penetra la atmósferra terreste se genera una Cascada atmosférica extensa. Una Cascada atmosférica es una cascada de partículas la cuál puede ser medida usando detectores de superficie. Este Trabajo Fin de Máster consistirá en el desalloro de un algoritmo de clasificación de las señales recogidas por los detectores de superficie con el objetivo de identificar la partícula primaria que originó la cascáda atmosférica extensa. El desempeño de las redes neuronales convolucionales, las redes neuronales prealimentada y las máquinas de vector soporte será comparado. La agregación de información de un mismo evento recogida por distintos detectores de susperfice será comparada con usar la información de cada detector de superficie por separado. La suma de valores externos medidos será realizada para comprobar si se mejora el desempeño comparado con usar solo las trazas o las características extraídas de las trazas.

Classification of Ultra-High Energy Cosmic Rays

Francisco Carrillo Pérez

Keywords: Cosmic Rays, Ultra High Energy, Mass Composition, Convolutional Neural Network, Support Vector Machines, Deep Learning

Abstract

Machine learning algorithms have shown their usefulness in a countless variety of fields. Specifically in the astrophysics field, these algorithms have helped the acceleration of our understanding of the Universe and the interaction between particles in recent years. Deep learning algorithms, enclosed in machine learning field, are showing outstanding performance in problems where spatial information is crucial, such as images or data with time dependency. Cosmic rays are high-energy radiation, mainly originated outside the Solar System and even from distant galaxies, that constitute a fascinating problem in Physics today. When a Ultra-High Energy Cosmic Ray enters the Earth's atmosphere an extensive air shower is generated. An air shower is a cascade of particles and can be recorded with surface detectors. This master thesis will develop a supervised learning algorithm to classify the signals recorded by surface detectors with the aim of identifying the primary particle giving rise to the extensive air shower. Convolutional Neural Networks along with Feed Forward Neural Networks, and Support Vector Machines will be compared. The aggregation of information from different surface detectors recording the same event will be studied against using the information of a single surface detector. The addition of external measurements would be perform in order to understand if they improve the performance against using the traces features isoletdly.

Yo, **Francisco Carrillo Pérez**, alumno del **Máster en Ciencia de Datos e Ingeniería de Computadores** de la **Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación de la Universidad de Granada**, con DNI 77140580Y, autorizo la ubicación de la siguiente copia de mi Trabajo Fin de Máster en la biblioteca del centro para que pueda ser consultada por las personas que lo deseen.

Fdo: Francisco Carrillo Pérez

Granada a 3 de Julio de 2019 .

D. **Luis Javier Herrera Maldonado**, Profesor del Área de Arquitectura y Tecnología de Computadores del Departamento de Arquitectura y Tecnología de Computadores de la Universidad de Granada.

D. **Alberto Guillén Perales**, Profesor del Área de Arquitectura y Tecnología de Computadores del Departamento de Arquitectura y Tecnología de Computadores de la Universidad de Granada.

Informan:

Que el presente trabajo, titulado ***Classification of Ultra-High Energy Cosmic Rays***, ha sido realizado bajo su supervisión por **Francisco Carrillo Pérez**, y autorizamos la defensa de dicho trabajo ante el tribunal que corresponda.

Y para que conste, expiden y firman el presente informe en Granada a 3 de Julio de 2019 .

Los directores:

Luis Javier Herrera Maldonado **Alberto Guillén Perales**

Yo, **Francisco Carrillo Pérez**, alumno del **Máster en Cencia de Datos e Ingeniería de Computadores** de la **Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación de la Universidad de Granada**, con DNI 77140580Y, declaro explícitamente que el trabajo presentado es original, entendido en el sentido que no he utilizado ninguna fuente sin citarla debidamente.

Fdo: Francisco Carrillo Pérez

Granada a 3 de Julio de 2019 .

Agradecimientos

First of all, I would like to thank both of my supervisors for the tremendous effort they have made in order to help me understand and love each part of this master thesis. Without them, I am sure this would have been a tougher path.

Thanks to my parents and siblings for always supporting me in every choice I have made until now. Thanks to my uncle and aunt, for being as supportive as they have been my whole life.

Thanks to my friends, people that I can always count on. Thankfully, there are too many for citing everyone's name here, but all of them are equally special in my heart. I am truly thankful.

Lastly, thanks to everyone that aims to seek for the truth and improve a tiny bit the world we live in. I look forward being one of them soon.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

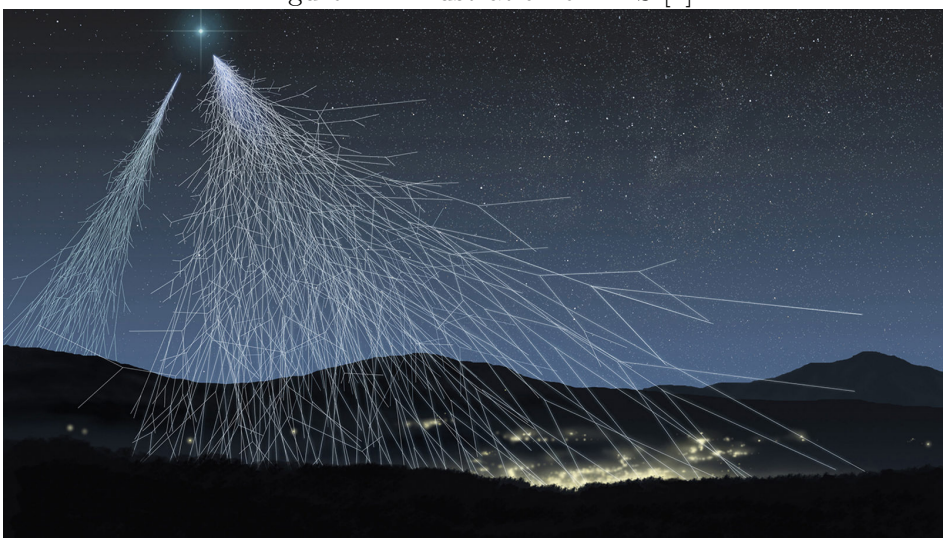
1.1 Cosmic Rays and Extensive Air Showers

Cosmic rays are particles that arrive from space and are constantly bombarding the Earth from every direction. The majority of these particles are atomic nucleus or electrons. Cosmic rays in the scale of several Joules travel at a speed close to that of light and have hundreds of millions of times more energy than particles produced by any accelerator in the world. The first evidences of their existence were collected more than fifty years ago [16]. However, the majority of their properties are still a mystery to us [17].

The source of Ultra-high energy cosmic rays (UHECR) is still unknown. Most of the particles from Low-Energy cosmic rays that arrive to Earth come from inside our galaxy, the Milky Way Galaxy. These particles sources ~~is~~ supposed to be the explosions of supernovas. However, most of the particles from UHECR might come from outside of the Milky Way. No source is known in the Cosmos that can produce particles with this high energy, not even the most violent explosions of stars. Wherever they come from, high-energy particles keep secrets about evolution and possibly the origin of the universe, due to the enigma of their enormous energy millions of times greater than any terrestrial particle accelerator can produce.

Extensive Air Showers (EAS), are a high-energy quantum process that develops in the Earth's atmosphere when a primary cosmic ray penetrates it [18] [19]. Fragments of this collision in turn collide with other air molecules, in a rain that continues until the energy of the original particle is destroyed among millions of particles falling on the earth. By measuring these cascades, the properties of the original cosmic particle or ray can be studied by scientists.

Figure 1.1: Illustration of EAS [1]



One of the problems that arise for its study is that UHECR are very rare and suppose a very small fraction of the whole set of cosmic rays. In fact, for every km^2 only one per year is detected for energies of $10^{19}eV$. This complicate their study and that is why data obtained by Monte Carlo simulations are often used along with actual data. Real measurements are not labeled (which particle has been produced or the particles that have interacted), which complicate tackling the problem as a supervised learning one. Therefore, the implementation of a classifier which could be applied to real data would help enormously physicists' work when they come to draw conclusions from real measurements.

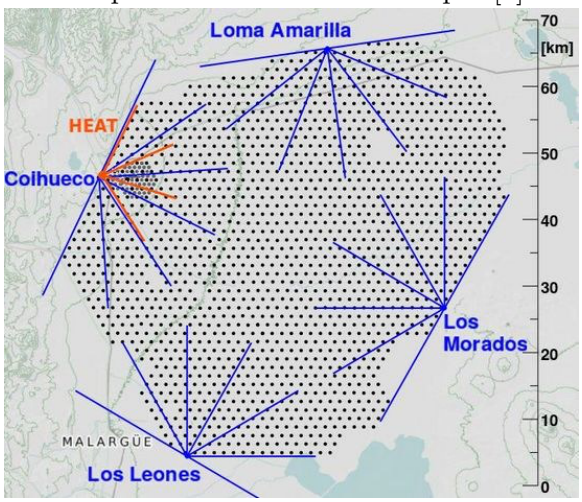
1.2 Pierre Auger Observatory

Pierre Auger Observatory in Argentina has the goal of measuring air showers that are produced when a primary cosmic ray enters the atmosphere. Therefore the energy, the arrival direction and the nature of cosmic rays with energy superior to $10^{19}eV$ can be studied.

The observatory is formed by a network of 1600 surface detectors (SDs) which are spaced 1,5 km apart and covering a total area of 3000 km^2. Each detector is equipped with three PMTs, the signals registered by them are digitized by 40 MHz 10-bit Flash Analog to Digital Converters (FADCs). This SD network is complemented by a set of highly sensitive telescopes that, on clear moonless nights, scrutinize the atmosphere to observe the faint ultraviolet light produced by cosmic ray cascades as they pass through the air. That's why the observatory is considered to be of type *hybrid* because

it used a join of water Cherenkov detectors and an atmospheric fluorescence telescope system.

Figure 1.2: Pierre Auger observatory's map. Each point represents a SD and each line an atmospheric fluorescence telescope. [2].



SD are based on the Cherenkov radiation. This radiation is an electromagnetic radiation emitted when a charged particle (such as an electron) passes through a dielectric medium at a speed greater than the velocity of light in that medium. This produces a characteristic blue light, which can be observed in Figure 1.3, caused by an electromagnetical perturbation [20].

Since SDs are based on Cherenkov radiation, they consist of a tank of hyperpure water, which is simply purified distilled water, of 12000 liters with which EAS's particles interact in their arrival to the ground, recording a certain amount of signal. This signal is the total energy recorded during a period of time, which will be explained in detail later on. In Figure 1.4 different parts of the SD are described. The gathered signal by each SD will be the starting point of this work.

Figure 1.3: Light emitted as a result of the Cherenkov effect in a nuclear test reactor [3].

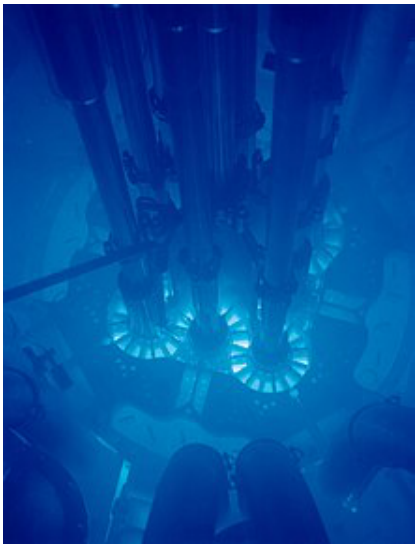
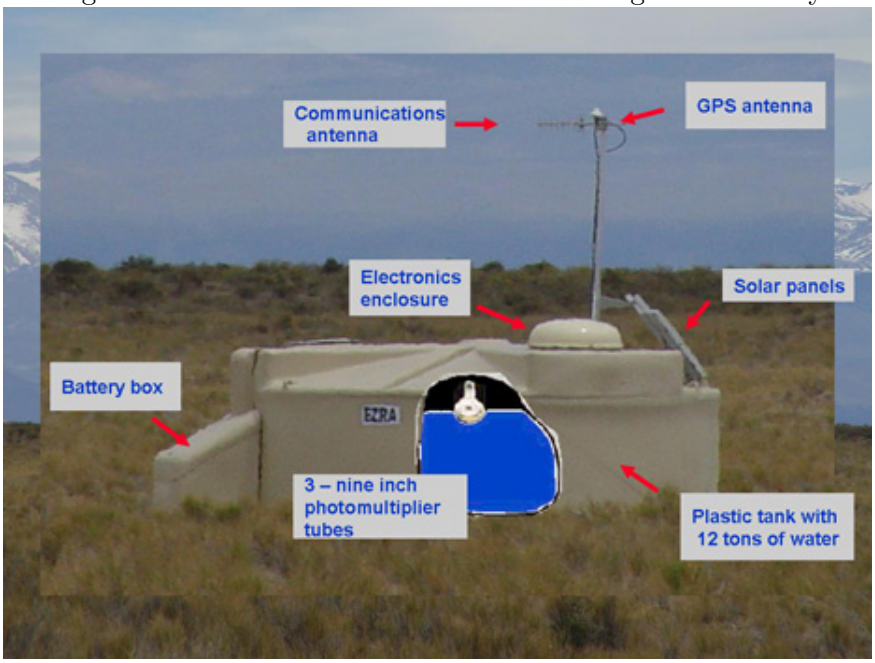


Figure 1.4: Parts of a SD from the Pierre Auger Obervatory.



1.3 Machine Learning in Astrophysics

Recent research has been focused on how Machine Learning (ML) techniques and methods could be applied to astroparticle physics problems like [21], [22] and [23] with interesting results. In [21] authors were able to reconstruct cosmic air showers signals using CNNs and, recently, in [24] the muon number was determined using a Deep Neural Network. In [22] authors used Graph Neural Networks to classify IceCube events, with remarkable results. Also in [23] authors applied CNNs to IceCube pulses in order to reconstruct muon-neutrino events for further study.

1.4 Aims of this work

The aims of this master's thesis were:

- Train a classifier in order to perform binary-classification between one particle and a group of them. Train a classifier to distinguish between all the different particles generated during simulations.
- Testing the automatic extraction of information relevant for the classification from the traces by means of Deep Learning algorithms.
- Obtain a comparison between the performance of the algorithms with different inputs. Specifically, using the trace measured by each SD or combining the traces from an event that have hit at least three SDs.
- Make a comparison between the performance of automatically extracted features and physicist's proposed variables computed from the traces.
- Perform experiments adding information from variables obtained from the cascade to the features from the traces, in order to test if performance is improved.
- Train a classifier with the variables obtained from the cascade. Compare its performance with the one obtained when using the traces.

Chapter 2

Foundations of the techniques used

In this Section the foundations of the different algorithms in this work are going to be described, along with a more thorough description of the problem which is going to be tackled. The algorithms that have been used in this work are the following: Artificial Neural Networks (ANN), Convolutional Neural Networks (CNN) and Support Vector Machines (SVM).

2.1 Machine Learning

2.1.1 Supervised Learning

Within the field of machine learning we can find four different types of learning: supervised learning [25], unsupervised learning [25], semi-supervised learning [26] and reinforcement learning [25].

In a supervised learning problem we labeled data, and our goal is to learn from that data in order to predict future samples. Differently, in an unsupervised learning problem, data are unlabeled and our goal is to learn the intrinsic statistical distribution of the data. Semi-supervised learning make use of unlabeled data for training, typically a small amount of labeled data with a large amount of unlabeled data. Lastly, reinforcement learning is concerned with how software agents ought to take actions in an environment so as to maximize some notion of cumulative reward.

All of these types can be applied to two different goals. One of them is classification, where a label want to be predicted. On the other hand, the goal could be regression, where a quantity is what wanted to be predicted.

In this work a supervised learning problem is faced, specifically a classi-

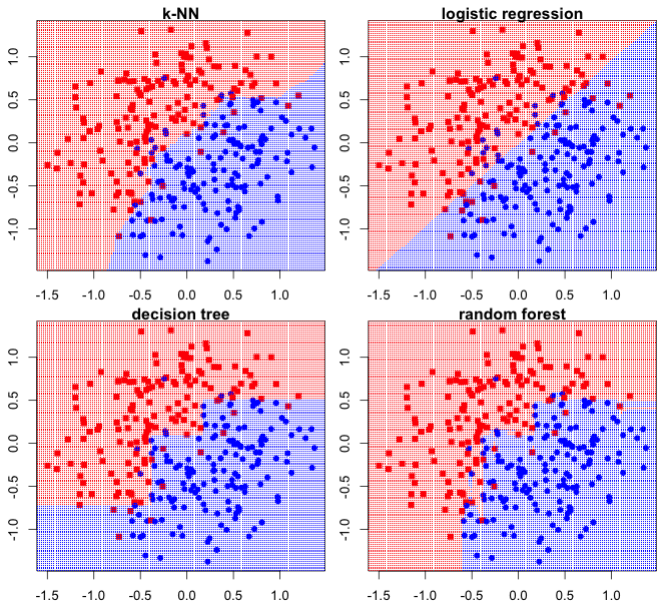
fication problem. A supervised learning problem is formed by a dataset, X of size NxM, and the class label for each data, Y. N denote the number of values in the dataset and M the number of features for each data sample.

$$X = \begin{bmatrix} X_{1,1} & X_{1,2} & X_{1,3} & ... & X_{1,M} \\ ... & ... & ... & ... & ... \\ X_{N,1} & X_{N,2} & X_{N,3} & ... & X_{N,M} \end{bmatrix} \tag{2.1}$$

$$Y = \begin{bmatrix} Y_1 \\ \vdots \\ Y_n \end{bmatrix} \tag{2.2}$$

Therefore, the function $f(X)$, that describes data behaviour is what is aimed to be found. In a classification problem, that function would be the one that separates in the best possible way the classes. However, since this function is unknown, a function $\hat{f}(X)$ such that $\hat{f}(X) \approx f(X)$ is what we search for.

Figure 2.1: Different examples of functions found for the separation of two classes using different classification algorithms [4].



2.2 Artificial Neural Networks

Artificial Neural Networks (ANNs) are learning algorithms based on the functioning of biological neural networks. They can be used for supervised,

unsupervised and reinforcement learning problems. This is one of the reasons why they've been used in a very large variety of problems.

A specific type of ANNs are the Feed Forward Neural Networks wherein connections between the nodes do not form a cycle. In this type of networks, the information moves in only one direction, forward, from the input nodes, through the hidden nodes (if any) and to the output nodes. There are no cycles or loops in the network [27].

2.2.1 Perceptron

The first model presented within the FFNN family was the perceptron [28]. The perceptron is formed by an unique neuron. The neuron, in turn, is formed by a weights matrix, W, a unique value named bias, B, and the activation function. The activation function determines if the neuron will activate or not. Given the sum of the product of each weight with the input, if the sum exceeds the threshold θ, it will return a one. If it doesn't, it will return a zero. An graphical representation of the perceptron can be observed in Figure 2.2.

The weights matrix is expressed as:

$$W = \begin{bmatrix} w_1 \\ \vdots \\ w_n \end{bmatrix} \tag{2.3}$$

The output, given a input x, would be:

$$f(X) = \begin{cases} 1 & \text{if } \sum_{n=1}^{N} w_i \vec{x_i} > \theta \\ 0 & \text{if } \sum_{n=1}^{N} w_i \vec{x_i} < \theta \end{cases} \tag{2.4}$$

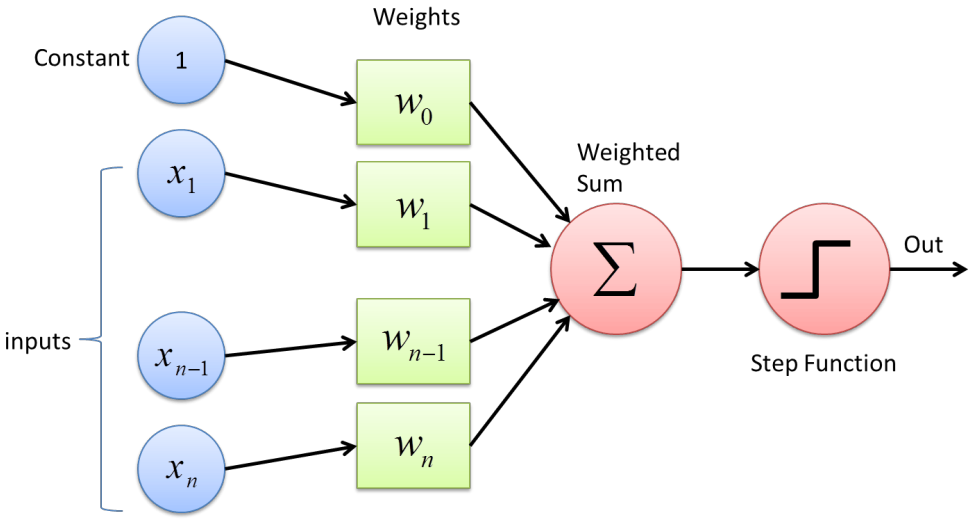
Based on this, an algorithm that given an input produces an output is obtained. The initial weights of the network are randomly intilialized. For each iteration an output is obtained and it is compared with the real value. But the weights of the network need to be learnt. That's when the Hebbian Learning Rule is used.

The Hebbian Learning Rule states that the synaptic connections strengthen when two or more neurons are activated adjoiningly both in space and time [29]. Therefore, the rules for updating the weights after each iteration are the following:

$$w_i^{k+1} = w_i^k + \triangle w_i$$

$$\triangle w_i = \eta \cdot t \cdot \vec{x_i}$$

Figure 2.2: Diagram of a perceptron [5].



Where η is the learning rate, t the real output and x_i the ith input.

It was proved that the perceptron is able to approximate linear functions. Nevertheless, it is not able to approximate non-linear functions. This was denoted as the XOR problem, given that there not exist a linear separation for this problem. As a solution the multi-layer perceptron was proposed.

2.2.2 Multi-layer perceptron

The multi-layer perceptron is formed by:

- **Layer**: Set of neurons.
- **Input layer**: Layer that receive the input data.
- **Output layer**: Layer that determines the output of the network. The output layer would have the same number of neurons and the number of classes in the dataset. Each neuron have the probability of the input to belong to each one of the classes.
- **Hidden layer**: Layer that process the information from the previous layers and connect with the next layer.

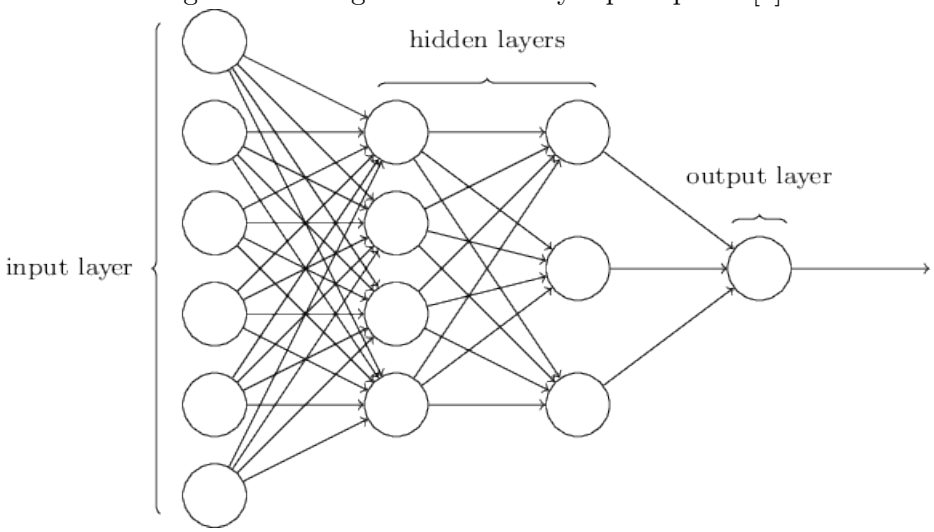
However, a new learning algorithm was needed. It wasn't until 1988 that it was proposed, the backpropagation algorithm [30] was proposed. With this new algorithms the weights are updated as follows:

$$\boldsymbol{w}^{k+1} = \boldsymbol{w}^k + \triangle \boldsymbol{w}$$

$$\triangle \boldsymbol{w}_i = -\eta \cdot \frac{\partial E}{\partial \boldsymbol{w}}$$

Where η is the learning rate and $\frac{\partial E}{\partial \boldsymbol{w}}$ is the errors' gradient in respect to the weights. The gradient gives how a function vary in respect to the variable is being derived. The negative sign is used since the error wants to be minimized. An example of a multi-layer perceptron can be observed in Figure 2.3. In addition, it was proven that the multi-layer perceptron is a universal approximator [31].

Figure 2.3: Diagram of multi-layer perceptron [6].



2.3 Convolutional Neural Networks

Based on the popularity of the multi-layer perceptron, other architectures have been proposed. One crucial field where ANNs have succeeded is in Computer Vision. Specifically, an architecture based on the convolution operation. In this case the convolution operator is applied as a matrix multiplication between a convolutional filter and the data. Convolution Neural Networks (CNNs) have been around for a long time. They where firstly proposed by Fukushima in [32]. However, it wasn't until diverse modifications were applied to the training algorithms, the quantity of data available was hugely increased and the necessary computing platforms were developed, that CNNs were revisited. One of the most important works from this new era of CNNs was proposed by Yann Lecun et. al. in [33]. In this work CNNs were applied to a digit classification problem. After that, a lot of works have been published showing how CNNs outperform the

state-of-the-art techniques in multiple computer vision problems [34].

2.3.1 Automatic extraction of features

What makes CNNs special is the ability to automatically extracting features from data. Previously, features were needed to be extracted by hand. This was one of the toughest tasks when tackling computer vision problems. Such features that are automatically extracted, such as lines, textures or corners were extracted using specific algorithms. For instance, if corners wanted to be detected, the most popular algorithms for this task was the Harris Corner detector [35]. The algorithm makes use of some filters to perform the detection. Those filters are the ones that are learnt by the CNN, applying the convolution operation and through the backpropagation of the error for the weights' update.

Two famous filters that were previously used, in the computer vision literature for instance, are the Sobel Filters [36]. Using both of them corners along the X and the Y axis could be detected. The X Sobel filter is

$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

while the Y Sobel filter is

$$\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

These filters were capable of detecting boundaries between objects in images. Equally, this filters can be applied to signals and different characteristics from them can be learnt. The great advantage of CNNs is that these types of filters can be automatically extracted and they later can be used by any kind of model in order to perform the classification. Thanks to the usage of different layers and the backpropogation algorithm, more complex or simpler filters can be learnt. These more complex filters would not only be able to detect more specific shapes in images or in the signals presented and that more explicit to the problem tackled, but they would also improved the model's performance agains other more traditional methods.

2.3.2 Types of layers

CNNs are formed by different kind of layers:

- **Convolutional layer**: As explained, they are based on the convolutional operation, which is applied to the whole data spatial domain

(image or signal). Their main goal is to extract the information of the data, transforming the input into another representation of features. Convolutional filters will be learnt. They are matrices of weights that are learned during the epochs of the algorithm and are able to discern certain types of patters within the data. In the case of images an specific filter learning would be able to determine is there are corners within the image.

- **Pooling layer**: Pooling layers reduce the dimensions of the data by combining the outputs of neuron clusters at one layer into a single neuron in the next layer. This would reduce the amount of information extracted by the convolution filters. They usually are placed after the activation function [37].

- **Fully connected/Dense Layer**: Fully connected layers connect every neuron in one layer to every neuron in another layer. It is in principle the same as the traditional multi-layer perceptron. They will used the features learned by the convolutional layers in order to perform the classification. One of them would be the output layer, as explained before.

An example of how convolutional filters and pooling operations are applied can be observed in Figures 2.7 and 2.8 respectively.

2.3.3 Pooling operators and activation functions

Different kinds of pooling operator have been presented in the literature. Max pooling uses the maximum value from each of a cluster of neurons at the prior layer [38]. Average pooling uses the average value from each of a cluster of neurons at the prior layer. Min pooling uses the maximum value from each of a cluster of neurons at the prior layer.

As explained in the previous section for the ANNs, the activation function would determine if the neuron is fired or not. When it comes to the activation function many options are available. In the hidden and input layers one of the utmost used activation functions in the late years is the Rectified Linear Unit(ReLU) [39], that is defined as the positive part of its argument. Then, it would be defined as

$$f(x) = x^+ = max(0, x)$$

. A smoothed approximation to ReLU would be the softplus or SmoothReLU that is defined as

$$f(x) = \log(1 + e^x)$$

. However better results have been obtained using ReLU in the literature [40]. An example of the different between ReLU and SmoothReLU can be observed in Figure 2.4. For the output layer different activation functions can be applied. In a binary classification problem, two functions can be used. One of them is the Sigmoid function which is described as

$$f(x) = \tau(x) = \frac{1}{1 + e^{-x}}$$

. Usually it is used for binary classification since it will return the class to which the inputs belong. However, when facing a multi-classification problem, another function is used. This function is the Softmax function that is described as

$$f_i(\overrightarrow{x}) = \frac{e^{x_i}}{\sum_{j=1}^{J} e^{x_j}} \text{for i=1,..., J}$$

.

An example of the Sigmoid activation can be observed in Figure 2.5. A graphical representation of the Softmax function can be observed in Figure 2.6. Softmax activation is used instead of sigmoid with the cross-entropy loss because softmax activation distributes the probability throughout each output node. When tackling a binary problem both of them can be used since sigmoid is just a special case of softmax function:

$$f(x) = \frac{1}{1 + e^{-x}} = \frac{1}{1 + \frac{1}{e^x}} = \frac{1}{\frac{e^x + 1}{e^x}} = \frac{e^x}{1 + e^x} = \frac{e^x}{e^0 + e^x}$$

Figure 2.4: Plot of the ReLU (blue) and SmoothReLU (green) functions near x = 0 [7].

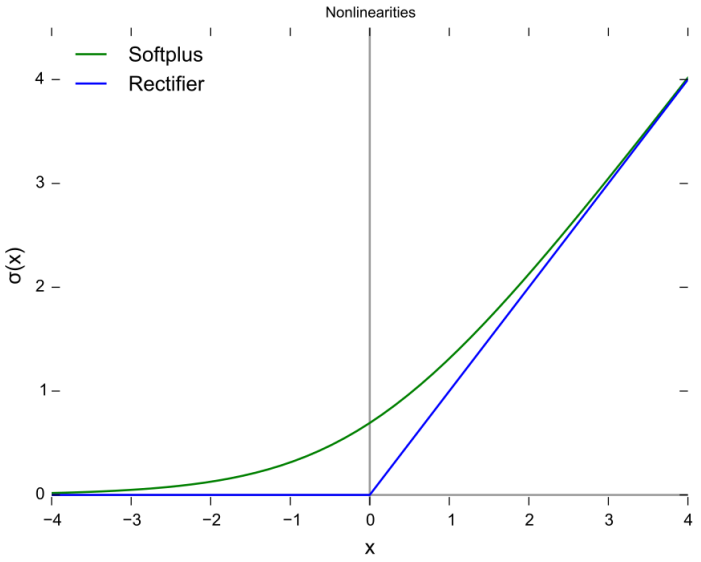


Figure 2.5: Graphical representation of sigmoid activation [8]

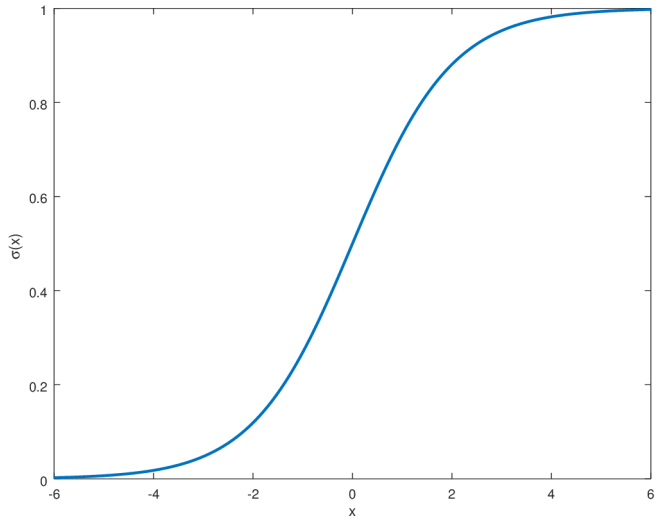


Figure 2.6: Graphical representation of softmax activation [9]

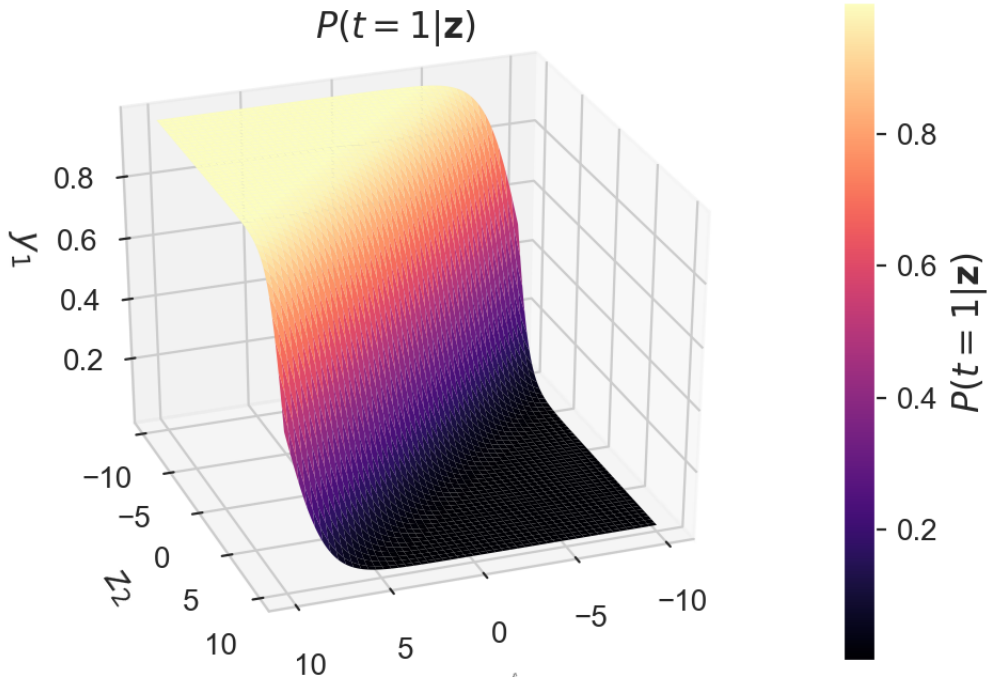


Figure 2.7: Example of application of a convolutional filter of size 3x3 [10].

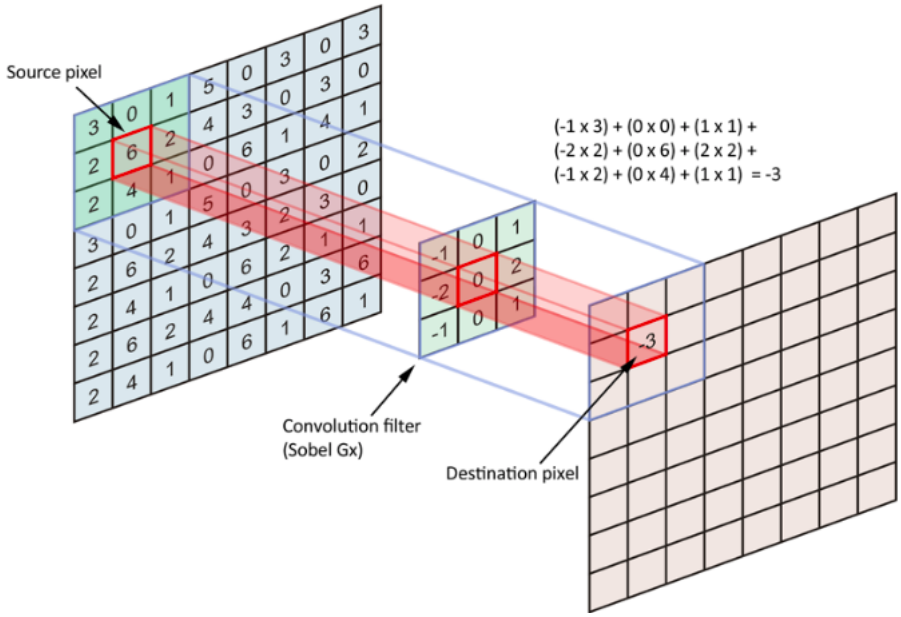
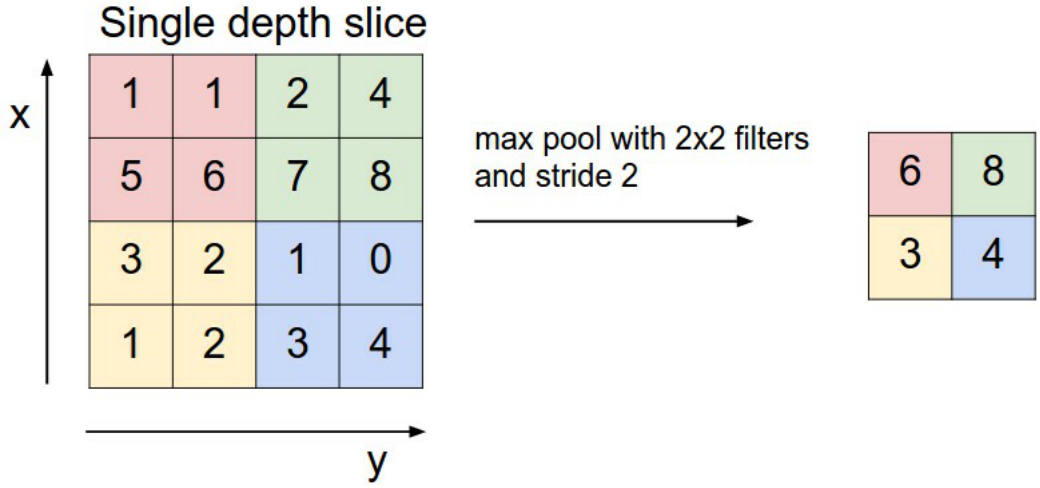


Figure 2.8: Example of application of MaxPooling operation. In Maxpooling the maximum within the values grouped by the pooling filter is the one that is selected. [11]



2.3.4 Training of CNNs

Optimizers and loss functions

As it has been said in the previous section, based on the loss function the weight's update is computed. How to compute how much to change this weights is an open question were multiple methodologies have been proposed. What we want to find are the weights that produce the global minimum loss. Since iterative methods for finding the global minimum, such as the Newton's method, can not be used since they are highly computational intensive, other methods such are gradient descent are needed. However, these methods do not guarantee to find the global minimum. Another approach is to start using gradient descent for the first iterations of the training process and then apply Newton's method to find the global minimum. Nevertheless, gradient descent has been the preferred method for computing how to change the gradients. There are many proposed algorithms in literature, however two stand out among the others. One of them is the Stochastic Gradient Descent (SGD) [41], which is an iterative method for optimizing an objective function with suitable smoothness properties (e.g. differentiable or subdifferentiable). It is called stochastic because the method uses randomly selected (or shuffled) samples to evaluate the gradients. Another vastly used is the Adaptive Moment Estimation optimizer (Adam) [42]. In this optimization algorithm, running averages of both the gradients and the second moments of the gradients are used. Adam has been vastly used in recent literature, obtaining great results [43].

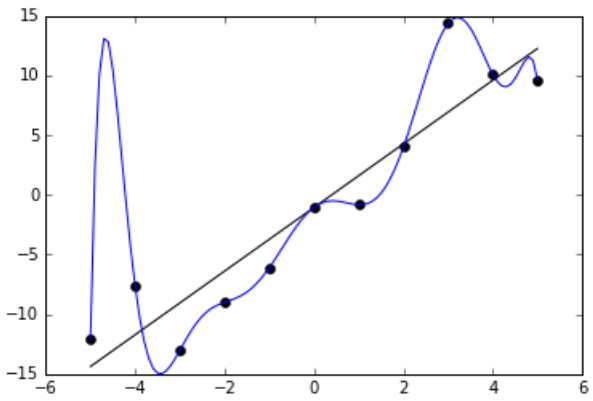
The loss function used in order to compute the error is equally crucial as other elements of the architecture. This can be a challenging problem as the function must capture the properties of the problem and be motivated by concerns that are important to the project. The main loss function used when tackling a classification problem is the Cross-Entropy Loss that make use of the Maximum Likelihood Estimation (MLE). MLE is based on the concept of entropy. Entropy is a measure of uncertainty of a random variable, that is defined as $H(X) = -\sum_x p(x) \log p(x)$ being $p(x)$ the probability mass function. The Cross-Entropy Loss can be applied to C classes and it is expressed as:

$$loss(x, class) = -\log\left(\frac{exp(x[class])}{\sum_j exp(x[j])}\right) = -x[class] + \log\left(\sum_j exp(x[j])\right) \tag{2.5}$$

Overfitting

Another great advantage for CNNs, as it is for the rest of Deep Learning models, is the ability to work with huge datasets. One of the basis of this ability is the ability to train using batches of data instead of needing to train with the whole dataset at once. A batch is just a chunk of the data of a certain size. If, for example, the dataset is formed by 1000 samples and a batch size of 10 is used, 100 batches would be used for the training. One iteration over the whole dataset, as described in the previous section, it is called an epoch. Epochs can be determined beforehand. However, if too many epochs are declared this could turn into overfitting. Overfitting happens when the error in the train set decreases while the error in the test set increases. What this means is that the generalization of the model is being badly affected. The term generalization means to the performance of the model when new samples are presented. In terms of the function, it will mean that what the function learnt is too similar to the function presented within the training data. A graphical example of overfitting can be observed in Figure 2.9.

Figure 2.9: Example of overfitting in a function. As observed a function that passes through all the points in the training set is presented, which would perform badly when new samples would be presented [12].



Hyper-parameters selection and preventing overfitting techniques

The choice of the hyper-parameters of the optimizer is very difficult task. They need to be chosen beforehand, even though there are some rules that can be generally applied. The hyper-parameters that can be chosen are the following:

- **Learning rate**: The learning rate defines how quickly a network updates its parameters. Low learning rate slows down the learning

process but converges smoothly. Larger learning rate speeds up the learning but may not converge.

- **Betas**: The betas in the Adam optimizer control the decay rates of the moving averages.

- **Weight decay**: During training, a regularization term is added to the network's loss to compute the backpropagation gradient. The weight decay value determines how dominant this regularization term will be in the gradient computation.

- **Number of hidden layers**: Increasing the number of hidden layers might improve the accuracy or might not, it really depends on the complexity of the problem that is being solved. Also, increasing the number of them could cause overfitting, since the model would overfit to the training set and wouldn't be able to generalize to new samples [44].

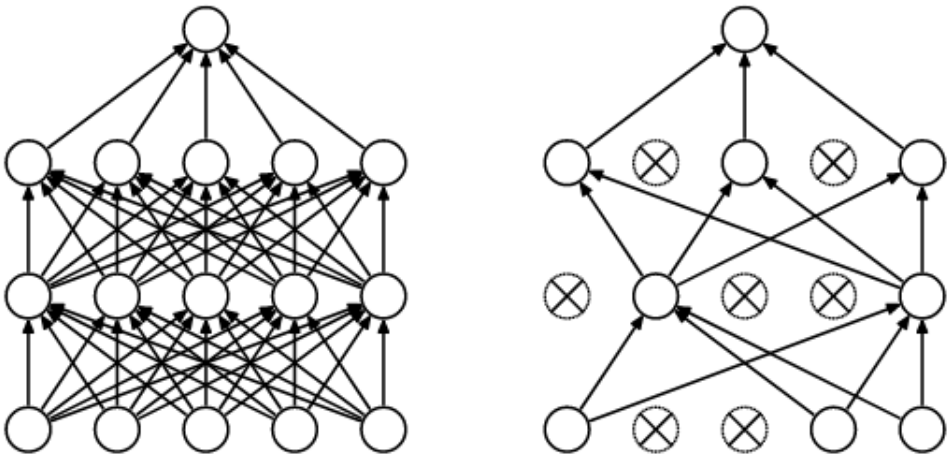
- **Number of neurons**: The number of neurons could affect the complexity of the function that will be learnt by the model. However, they could not be increased without taking into account the number of samples that are available since using more neurons' parameters that samples would be completely inefficient and unnecessary [44].

Other hyper-parameters such as the batch size could improve the generalization of the model, however it is not such an important factor. Another method in order to reduce overfitting and improving the generalization is the Dropout technique [13]. The Dropout techniques cancels randomly the connection of a number of N connections between neurons each epoch.

Batch Normalization [45] was introduced as a overfitting preventing method as well as for reducing the training time. What Batch Normalization does is normalize the output of a previous activation layer by subtracting the batch mean and dividing by the batch standard deviation. Also, batch normalization allows each layer of a network to learn by itself a little bit more independently of other layers.

An example of the effect of Dropout can be observed in Figure 2.10.

Figure 2.10: Effect of dropout in a Neural Network [13]



(a) Standard Neural Net (b) After applying dropout.

Weight initialization

As it has been said before, the weights of the network are randomly initialized. However, it has been shown in literature that a good weight initialization can led to a better performance [46]. One widely used method in the recent years it has been the Xavier weights initialization method, presented in [47]. Xavier weights initialization doesn't assure a better performance, but the same performance could be achieved in less time than using random initialization weights.

2.4 Support Vector Machines

SVM is also a supervised learning algorithm. Nonetheless, the main difference with the ANN and CNN is that is, in its basis, a non-probabilistic binary linear classifier. This means that the output of the SVM is not the probability of belonging to each one of the classes but the class itself. An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall. Even though SVM is a binary classifier it can also be applied as a

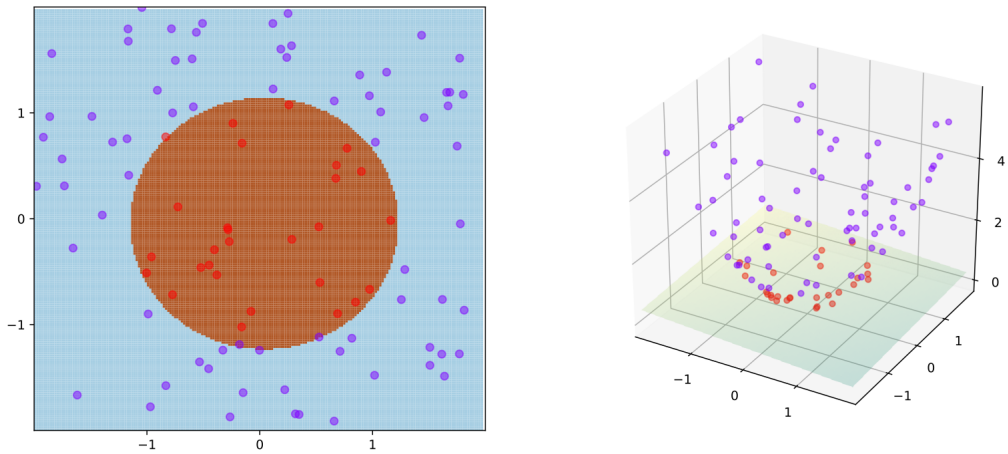
multi-class classifier following an One-Against-One (OVO) methodology for the classification. In an OVO methodology $K(K1)/2$ binary classifiers would be trained. At prediction time, a voting scheme is applied: all $K(K1)/2$ classifiers are applied to an unseen sample and the class that got the highest number of "+1" predictions gets predicted by the combined classifier [48].

In addition to performing linear classification, SVMs can efficiently perform a non-linear classification using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces. The kernel trick helps mapping the inputs to a high-dimensional feature space but without explicitly knowing the space, the inner product in the space is the only thing that needed to be known. For all x and x' in the input space X, certain functions $k(x, x')$ can be expressed as an inner product in another space V, being $k(x, x')$ the so called kernel function. The computation is made much simpler if the kernel can be written in the form of a "feature map" $\varphi : X \longrightarrow V$ which satisfies:

$$k(x, x') = \langle \varphi(x), \varphi(x') \rangle v$$

.

Figure 2.11: Example of how the kernel trick is applied to input data. The training points are mapped to a 3-dimensional space where a separating hyperplane can be easily found [14].



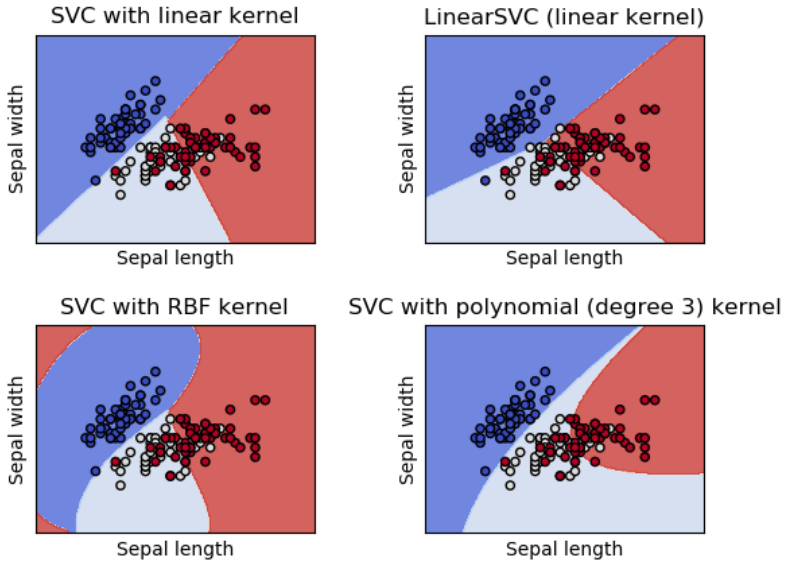
2.4.1 Types of kernels

Different kernels function are available. The most used ones are the following:

- **Radial Basis function kernel**: $k(x, x') = exp\left(\frac{||x-x'||^2}{2\sigma^2}\right)$.
- **Polynomial Kernel**: $k(x, x') = (x^T x' + c)^d$
- **Linear Kernel**: $k(x, x') = x^T x'$
- **Sigmoid Kernel**: $k(x, x') = \tanh(\gamma x^T x' + r)$

An example of how the different kernel functions would affect the hyperplane found could be observed in Figure .

Figure 2.12: Example of the classification hyperplane found when using different kernel functions [15].



Chapter 3

Auger's data

3.1 Signals recorded by SDs and external variables

In Chapter 1 how SDs record a signal based on the interaction between subatomic particles and the SDs has been described. The signal, which is denoted in bibliography as S_{total}, is measured in *Vertical-Equivalent-Muons* (VEMs) [49]. One VEM correspond to the energy deposited by a muon, which is an elementary particle found only in cosmic rays, that vertically crosses the water in the SD.

In literature it has been studied that S_{total} can be discomposed in two components, the muonic component (S_μ) and the electromagnetic component S_{em}) [50][51][52][53]. Therefore, we can describe $\int S_{total}$ as $\int S_{total} = \int S_\mu + \int S_{em}$.

As explained in Chapter 1, extensive air showers are very rare and they are not labeled. Therefore simulations need to be produced in order to have data to work with. These data are provided by the QGSJET-II package [54]. QGSJET-II is a model used by CORSIKA [55]. From certain particles, which for this work will be helium, iron, nitrogen, proton and photon, the simulators are capable of generating cosmic air showers which, in turn, simulate the new subatomic particles created by the different interactions (from which the muonic signal and the electromagnetic signal are obtained). Therefore, simulators are able to provide examples where S_{total}, S_μ and S_{em} are simultaneously known.

Each signal is composed by measurements taken in 25 ns bins. A total of 200 bins of energy are recorded. One or more SDs can record the signal. The distance to the center of the event and the mass of the particle would determine the amount of signal that is gathered. The signal presents a characteristic shape where it increases until reaching a peak and then decrease until it reaches zero. An example of signals for each particle can be observed

in Figures 3.1, 3.2 and 3.3.

Other variables can be obtained from the signals or from other external measurements. They have been proposed by physicist in literature [56]. The usage of these variables has been proposed in literature for predicting the number of muons in an event [57]. These variables can be grouped in two main categories: computed variables (those that have been computed from the traces' gathered) and external variables (those that have been measured from the event). These computed variables are:

- **Total signal**: Integral of the total signal registered by the station. It is measured in VEMs.
- **Muon signal**: Integral of the muon total signal registered by the statation. It is measured in VEMs.
- **Risetime,** $t_{1/2}$: The risetime is defined as the time it takes for the integrated signal to grow from 10% up to 50% of its total value and it is measured in ns.
- **Falltime**: Time for the integrated signal to go from 50% to 90% of its total value and it is measured in ns.
- **Area over Peak**: It is defined as the ratio of the integral of the trace to its peak value.

Whereas the external variables are:

- **Energy**: The true energy of the event simulated by the Monte Carlo simulator. It is measured in eV.
- **Zenith Angle**: Impact's angle of the cosmic ray. It is measured in degrees.
- **R**: Distance of the station to the reconstructed core of the air-shower. It is measured in meters.
- **Xmax**: The depth of the shower maximum. Not always Xmax can be measured.
- **Azimuthal angle**: Azimuthal angle of the detector. It is measured in radians.

3.2 Data collection

Data collection from the simulators, is experts' responsibility. QGSJET-II obtains a series of events (a event is how cosmic air shower is termed). For

each event, multiple detectors record the signal from particles that arrive to the surface. From this signal all the attributes described in the previous section are known. QGSJET-II also attends to the idiosyncrasy of each primary, allowing us to simulate events by knowing which primary interacts with the environment. Consequently, we have also the label for the supervised learning algorithms.

Based on how close the SD has been to the center of the event, the signal recorded may be saturated. What this means is that the peak of the signal is unknown. When the detector's hardware components saturate they produce a flat line where the saturation has been produced. In order to not loose information, a dataset formed only by non saturated samples was asked for.

Figure 3.1: Example traces for S_{total}, S_{μ} and S_{em} for Photon and Proton. The three components are presented from a recording of the same event for each particle. As it can be observed, the summation of the area of S_{μ} and S_{em} would be equal to the area of S_{total}.

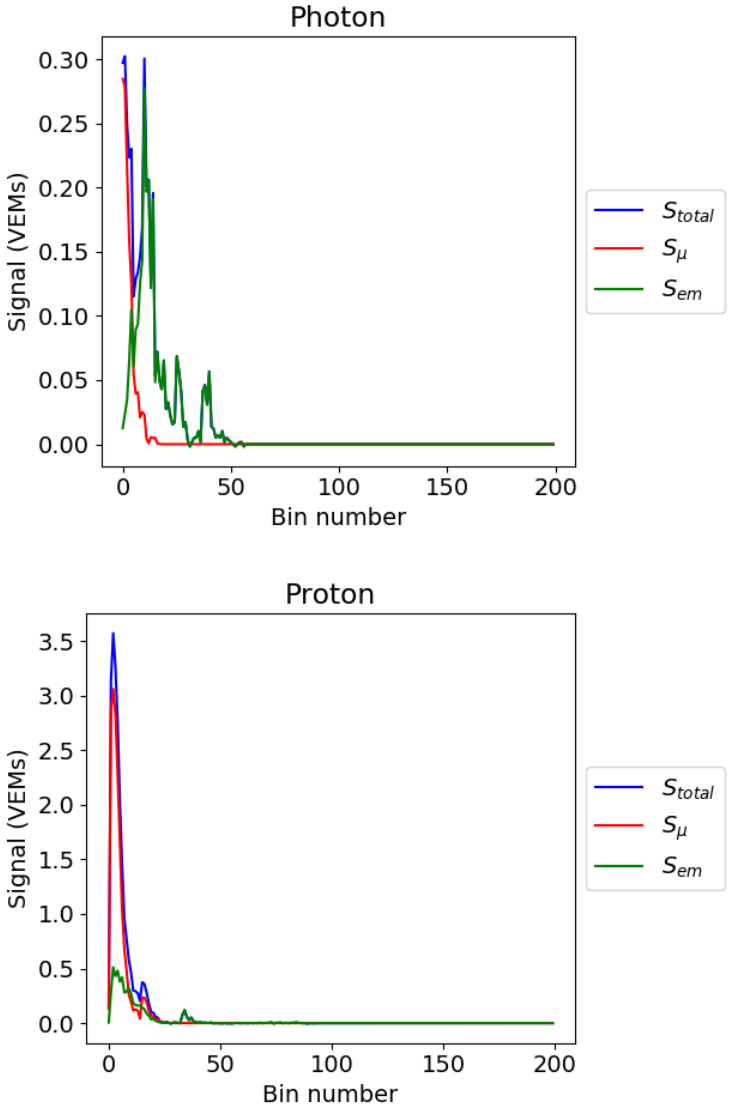


Figure 3.2: Example traces for S_{total}, S_{μ} and S_{em} for Helium and Nitrogen. The three components are presented from a recording of the same event for each particle. As it can be observed, the summation of the area of S_{μ} and S_{em} would be equal to the area of S_{total}.

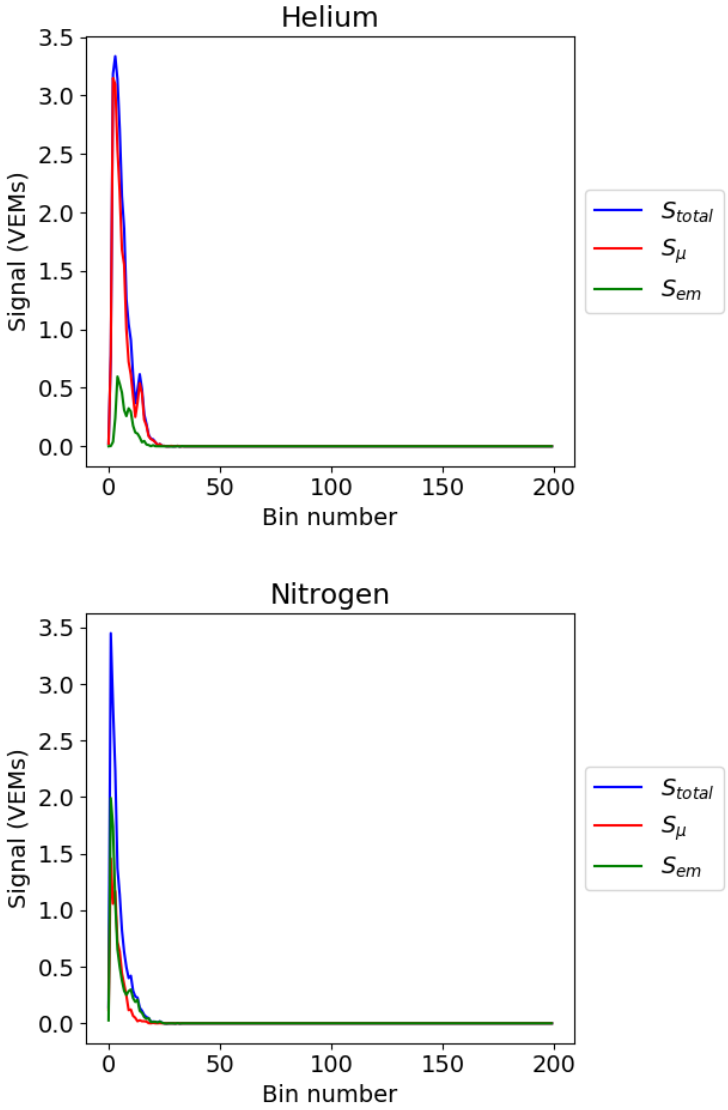
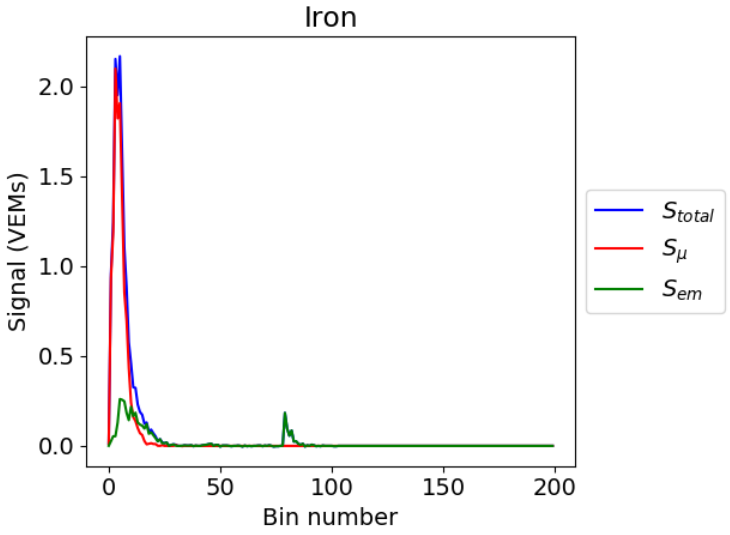


Figure 3.3: Example traces for S_{total}, S_{μ} and S_{em} for Iron. The three components are presented from a recording of the same event the particle. As it can be observed, the summation of the area of S_{μ} and S_{em} would be equal to the area of S_{total}.



Chapter 4

Methodology

In this section the explanation of how the problem has been addressed and which steps have been taken in order to arrive to the best solution will be given.

4.1 Problem definition

A supervised classification problem is addressed as it has been explained in Chapter 2, specifically in Section 2.1.

As is has been previously explained it is a problem of multi-classification, in this case with five classes denoted as: **Photon, Proton, Helium, Nitrogen and Iron**.

Each particle has a different mass. Depending on the mass, the proximity to the surface detector and other variables the gathered energy will be higher or lower. One clear differentiation can be made between the Photon, which practically lacks of mass, and the rest of particles, each one having more mass than the previous ones.

Tanking this into consideration, and also because the experts suggested it is the most important problem in cosmic rays identification, it was first decided to approximate the problem as a two-class classifier, Photon vs Hadron (which is how the group of no-photon particles are determined). Once a proper methodology was found for this problem, a more complicated problem was decided to be tackled, the training of a classifier in order to classify between the five classes described. Given the requirements of the algorithms used, classes needed to be described by numbers. When it comes to the binary classification problem classes were labeled as 0, if it was the Photon, and 1 otherwise. In case of the five classes problem, they were labeled from 0 to 1, being 0 the particle with the lower mass and 4 the one

with the higher one.

Energy traces, as it has been previously defined, can be separated in muonic and electromagnetic components. However, what SD are gathering is the total signal. Therefore, experiments were performed in order to probe with which traces or grouping we would obtain the best results in the classification task. Lately, it was probed if the combination of the three components presented better results than using each component isolatedly. Works are currently being carried in order to predict S_μ from S_{total} with successful results.

In addition, if combining the information from more than one SD improved the classification result was a question that was wanted to be answered in this work.

As it has been explained in Section 3.1, variables computed from the traces were provided. Therefore, if results when using the DNN with automatically extracted features outperform those obtained by using physics' defined variables extracted from the, it would show the convenience of this type of techniques to extract valuable knowledge that might be out of the experts' knowledge. An important part of the experiments aimed to answer that question.

Lastly, newer experiments wanted to be performed in order to understand if adding external information of the traces improved the classification. External variables are not always measured in the observatory, therefore work with traces or computed variables of the traces was prioritized. This methodology has been followed step by step.

For a better assessment of the models in the problem tackled, five different random train-validation-test subdivisions of the whole available dataset were performed, using a specific seed in order to maintain consistency across different experiments. The same event could be recorded by more than one SD. Therefore, in order to avoid information leakage when performing the different splits, SDs from the same event could belong to only one partition, being it training, validation or test. When performing the binary classification, since taking all the samples from all classes would led to an unbalanced dataset, a subsampling from the Hadrons class was performed, tanking the exact same number of samples from the subsampling classes. All of them were chosen randomly using a seed in order to maintain consistency. The number of samples in each of the splits can be observed in Tables 4.1 and 4.2.

Metrics used in order to evaluate models performance were accuracy, sensitivity, specificity and F1-Score (when multi-classification was addressed) both in validation and in test sets. Accuracy represents the overall performance of the network over the test set. Sensitivity and specificity are

computed based on the confusion matrix. Sensitivity represents how many true positives (photons) are correctly classified. Specificity represents how many true negatives (hadrons) are correctly classified. F1-Score represents the overall performance for each given class against the total of them.

Table 4.1: Number of samples in each of the splits for the binary classification when using the information of each SD separately.

Photon vs Hadron			
	Train	Test	Val.
Split 1	44086	14668	14689
Split 2	44147	14662	14634
Split 3	44120	14669	14654
Split 4	44123	14673	14647
Split 5	44216	14596	14631
5 classes			
	Train	Test	Val.
Split 1	95835	31649	31908
Split 2	95850	31611	31931
Split 3	95785	31789	31818
Split 4	95696	31972	31724
Split 5	95773	31659	31960

Table 4.2: Number of samples in each of the splits for the binary classification when using the information of a event that hit more than 2 SD.

Photon vs Hadron			
	Train	Test	Val.
Split 1	4182	1382	1420
Split 2	4207	1397	1380
Split 3	4185	1408	1391
Split 4	4205	1400	1379
Split 5	4262	1354	1368
5 classes			
	Train	Test	Val.
Split 1	9308	2959	3105
Split 2	9305	2951	3116
Split 3	9245	3040	3088
Split 4	9259	3117	2996
Split 5	9320	2991	3061

4.2 Data preprocessing

Data preprocessing is the key in order to obtain remarkable results. This is based on the internal computations of the algorithms,which sometimes need a previous set of transformations, for instance to help them homogenize data ranges.

Based on this, traces, computed variables and external variables have been normalized with mean zero and standard deviation one. Given a data vector X its normalization would be given by

$$X = (X - \mu)/\sigma$$

.

Those SDs that showed to be nearer to the center of the event have more information than those which are more distant from it. That's why a prior selection of the SDs has been performed, selecting those SDs that where between 500 and 1000 to the center of the event they gathered information from. Distances where chosen in order to minimize the distance to the center of the event while maximizing the number of events available.

When combining the information of various traces, in order to feed them to a CNN, as in Sections 5.2 and 5.3, the traces were concatenated along the Y axis as follows:

$$\begin{bmatrix} S_{total} \\ S_{\mu} \\ S_{em} \end{bmatrix}$$

The same method of combination was performed in the experiments of Section 5.1 when integrating the information of three SDs, but using only S_t or S_μ:

$$\begin{bmatrix} S^1_{total} \\ S^2_{total} \\ S^3_{total} \end{bmatrix} \qquad \text{OR} \qquad \begin{bmatrix} S^1_{\mu} \\ S^2_{\mu} \\ S^3_{\mu} \end{bmatrix}$$

Nonetheless, when this same information was needed to be fed into the FFNN, the traces were concatenated along the X axis as follows:

$$\left[S^1_{total}, S^2_{total}, S^3_{total}\right] \qquad \text{OR} \qquad \left[S^1_{\mu}, S^2_{\mu}, S^3_{\mu}\right]$$

For SVM, depending on the number of variables used, the inputs vary. However, it follows the same principles as the FFNN. When using the information from three SDs the variables are concatenated along the X axis as follows:

$$\left[V^1, V^2, V^3\right]$$

4.3 Architectures' design

In this section the architectures used during this work would be described. Since several architectures were tested for the CNN, some of them will be omitted in order to maintain simplicity.

4.3.1 Groups of experiments performed

Two different major groups of experiments were performed. One of them with preliminary experiments associated to the publication [58], which results are presented in Section 5.1, and another group of experiments with improvements and tuning of the architectures, the comparison with non ANNs models and the addition of external information from the events, which results are presented in Sections 5.2 and 5.3. From now on, these two groups will be denoted as Group A and Group B respectively.

4.3.2 FFNN

For group A experiments, the architectures used for FFNN can be observed in Figure 4.1.

Two different architectures were chosen in order to understand if the depth of the networks would influence the network's performance. Therefore, two FFNN architectures were proposed, one with three dense layers and the other one with two dense layers. From now on, they will be denoted as FFNN1 and FFNN2 respectively.

4.3.3 CNNs

A first CNN architecture was proposed for Group A experiments. It was formed by two convolutional layers. As dense layers, the architecture of FFNNs that obtained the best performed was added to the output of the convolutional layers. The architecture used for Group A experiments can be observed in 4.2.

Based on these results, Group B experiments were performed applying changes. All these change, will be described below.

Different kernel sizes where tested as well as the number of layers, the usage of a pooling layer and number of neurons. After all the experiments, the final architecture used for experiments performed using the grouping of all signal components can be observed in Figure 4.3.

When using external or computed variables they were concatenated with the output of the convolutional layers, as described in Figure 4.4. Different

experiments were performed in order to choose whether combining the variables with the output of the convolutional layers or the dense layers. Better results were obtained following the first approach.

When using three SDs another approach was taken. Given that we have already trained a feature extractor for the signals, we use the weights of the previous network. Specifically those from the convolutional layers and the fully connected layers. Weights and biased from this layer are frozen. Only another fully connected layer is the one that is trained. A graphical representation of the architecture can be observed in Figure [**?**].

In order to choose the CNN architecture, the features produced by the different convolutional layers and the dense layers were observed. It seemed that increasing the number of convolutional layers above three didn't improve the feature's separation between the classes. Tests were performed looking at the features produced after the pass through the dense layers. The features produced after the first dense, were different enough to perform the classification. However, if the network is deeper, what means adding more dense layers, the gap between the features of the samples that are incorrectly classified from both classes is wider. Nevertheless, stacking more than three dense layers didn't result in a better performance. Therefore, at the end, three convolutional layers were chosen as well as three dense layers.

In order to avoid overfitting different methods were chosen. A Batch Normalization layer was added after the convolution. Also, weight decay was used during training. In order to speed training up and also start in a better point for the optimizer, Xavier weight initialization was used for each dense and convolutional layers.

One difference between the data inputs between experiments of Group A and Group B was the inputs' shape. For Group A, data inputs were used as one dimensional in order to apply one-dimension convolutions, whereas in the second case data were expanded one dimension in order to apply two-dimensional convolutions. Henceforth, the dimensions of inputs in experiments in Group A was (3,200) -or (1, 600) if FFNN were being used, while in Group B it was (1, 3,200).

4.3.4 SVMs

Given the quantity of data and the computational needs of SVMs where it is high in size, when training with the information of one SD, default parameters were selected. However, when training with the information of three SDs a Grid Search was used in order to find the best hyper-parameters. The ranges used were:

- **Kernel**: RBF.

- **C**: $2^{-11}, 2^{-8}, 2^{-5}, 2^{-2}, 2.2^{4}, 2^{7}, 2^{10}$
- **Gamma**: $2^{-11}, 2^{-8}, 2^{-5}, 2^{-2}, 2.2^{4}, 2^{7}, 2^{10}$

4.4 Implementation's details

All the implementation has been performed using Python 3.6. There are two main scripts which are, **main.py** and **svm.py** for performing the experiments for the FFNN and CNN and SVM respectively. As a Deep Learning framework for the coding of the FFNN and CNN architectures, Pytorch was chosen [59]. It was chosen based on the utilities it provides for easy and fast prototyping. For SVM implementation the library Scikit-Learn [60] was chosen, which SVM's implementation is based on the famous library LIBSVM [61]. For data preprocessing and treatment the Pythons libraries Pandas [62] and Numpy [63] were used. In order to perfrom plots from results, Matplotlib [64] library was used.

Besides the two main scripts, other files were used providing the necessary functions for execution. In the file **NN.py** the FFNN and CNN architectures for performing the different experiments are described. In the file **utils.py** utilities for creating the different train-validation-test splits are provided. In the file **train.py** the different functions for training, validating and testing the neural networks architectures are declared. In the file **read_data.py** the functions for the reading the data are defined.

The two scripts have different options that can be set via the command line for further use. These options are:

- –**conv**: whether to use the CNN or the FFNN.
- –**name**: name used for saving the files produced (plots and models).
- –**epochs**: number of epochs for training the model.
- –**onetrace**: whether to use the information of each SD separately or combining the information from the different SDs.
- –**onevsall**: if binary or multiclassification want to be performed.
- –**elec**: whether to use all the traces together.
- –**exo**: if the information of the computed and external variables want to be added.
- –**threeconvs**: if three different networks want to be used when using the combination of SDs' information.

- –**variables**: if exo is true then which variables are wanted to be used.
- –**splits**: number of splits that wanted to be used.
- –**saveweights**: if the model weights are saved.
- –**fc**: number of neurons for the dense layers.
- –**fullbatch**: if the batch size is equal to the number of samples.
- –**savefig**: if the plots are saved.
- –**svm**: if a SVM is used as the classifier for the characteristics extracted by the network.

Figure 4.1: The two selected Feed Forward Neural Network architectures for experiments in Section 5.1 (input layer -with no calculi- and output layer were omitted for simplicity).

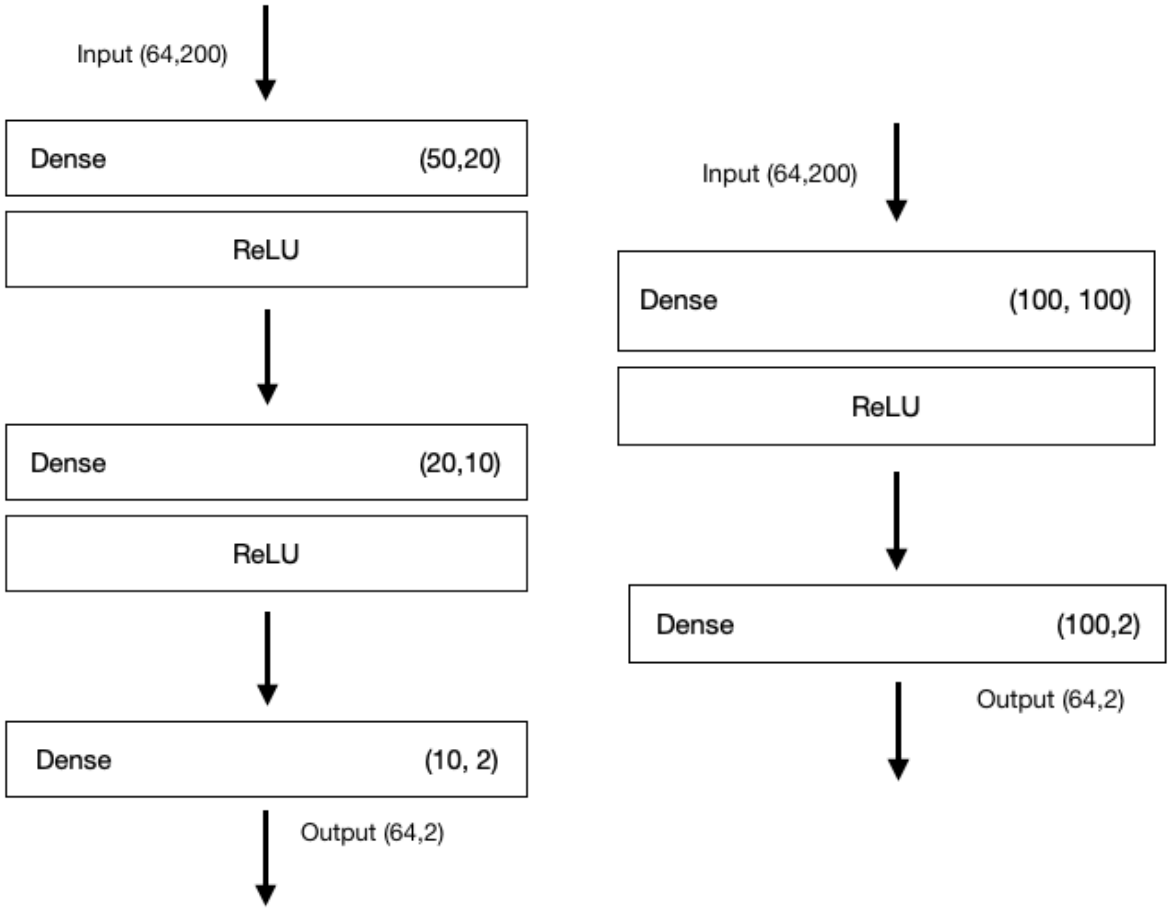


Figure 4.2: Convolutional Neural Network architecture for experiments in Section 5.1 (input layer -with no calculi- and output layer were omitted for simplicity).

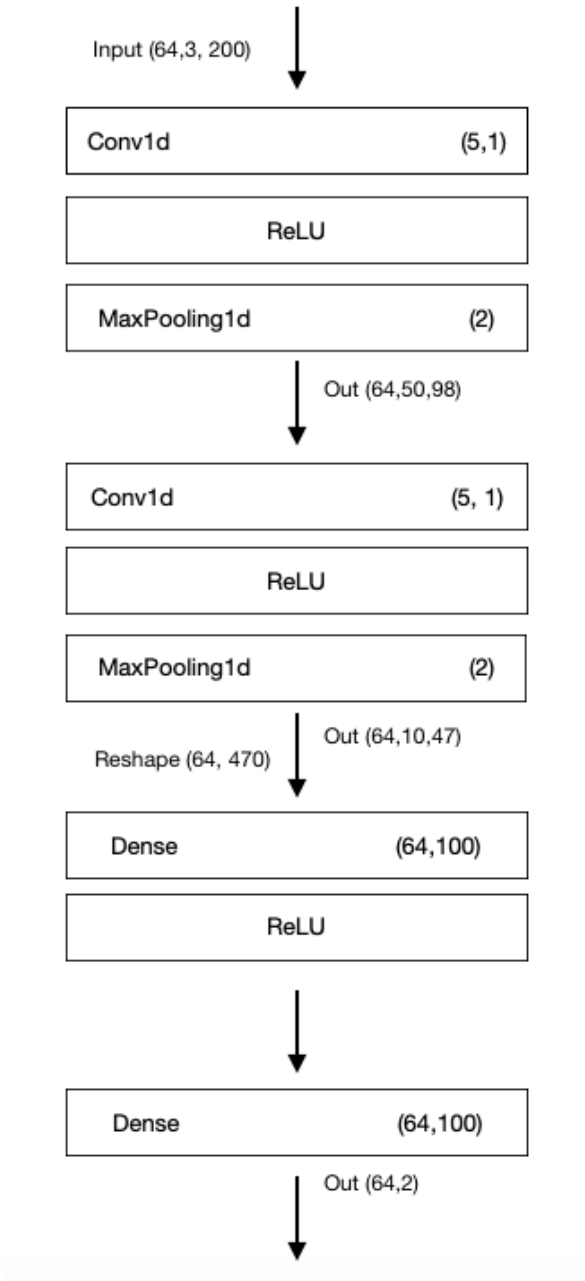


Figure 4.3: Convolutional Neural Network architecture for S_{total}, S_{μ} and S_{em} inputs for experiments in Section 5.2 (input layer -with no calculi- and output layer were omitted for simplicity).

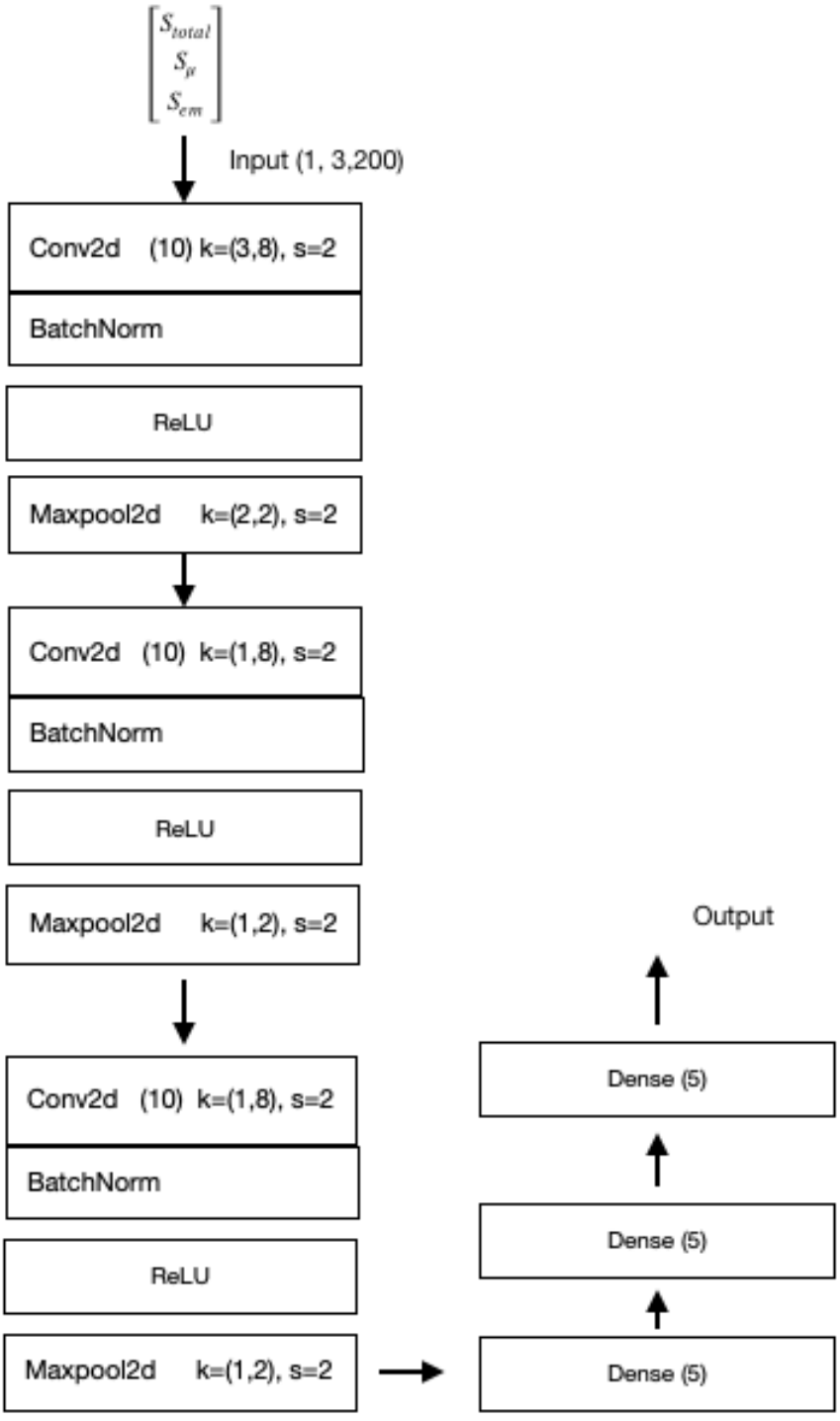


Figure 4.4: Convolutional Neural Network architecture for S_{total}, S_{μ} and S_{em} inputs concatenating variables information for experiments in Section 5.3 (input layer -with no calculi- and output layer were omitted for simplicity).

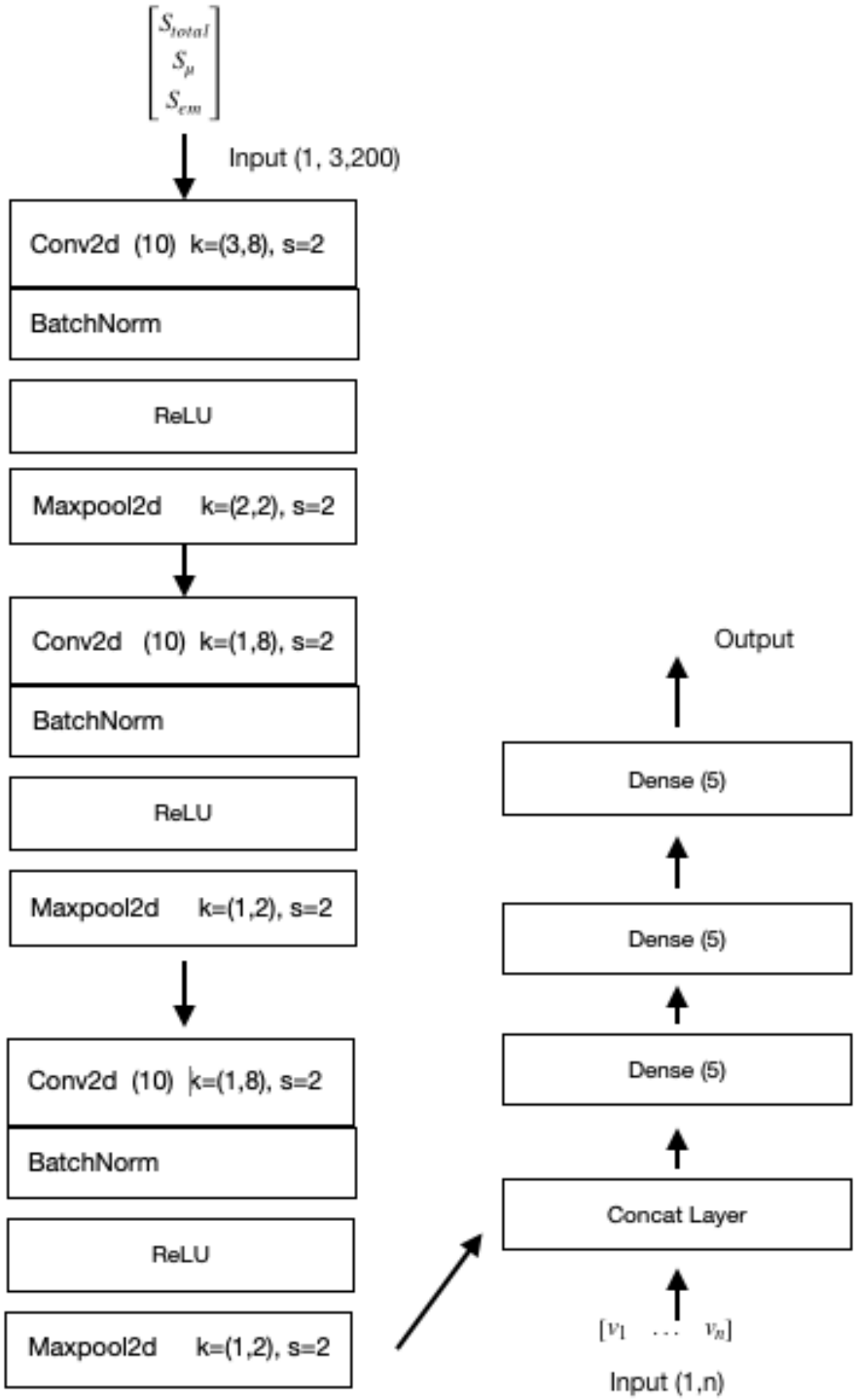
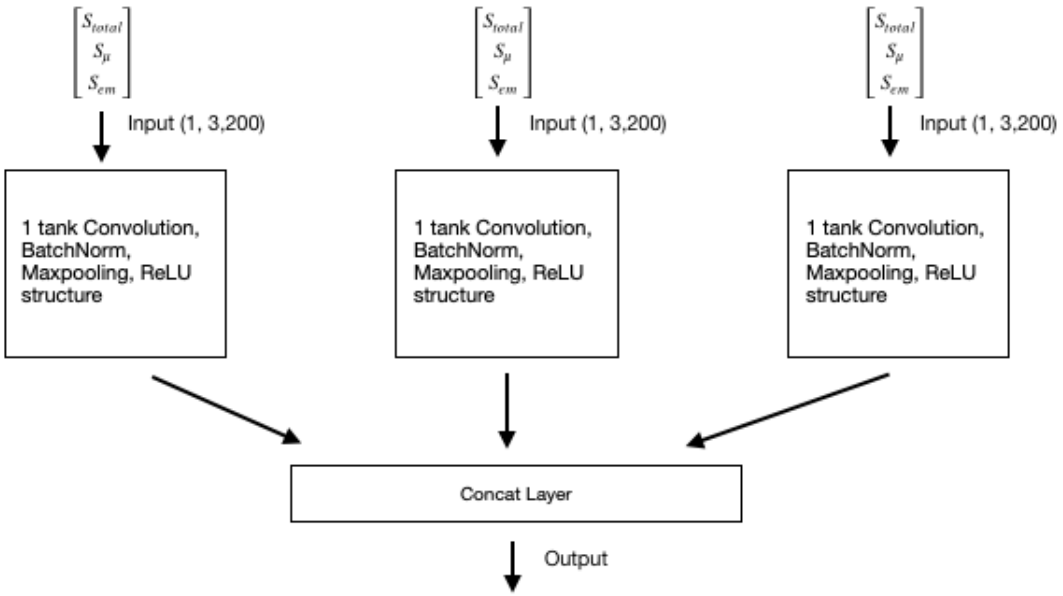


Figure 4.5: Convolutional Neural Network architecture for S_{total}, S_{μ} and S_{em} inputs using the information of three SDs for experiments in Sections 5.2 and 5.3 (input layer -with no calculi- and output layer were omitted for simplicity).



Chapter 5

Experiments and Results

In this chapter results obtained will be presented as well as discussed. They will be presented following the incremental methodology used during the development of this Master thesis.

Firstly, described in Section 5.1, experiments were carried out in order to understand if the usage of CNNs improved results over a FFNN. S_μ and S_{total} were used in order to prove which of the two types of signal provided itself the best performance. In addition, experiments considering events covering three SDs were assessed.

Latterly, outlined in Section 5.2, experiments were performed in order to understand whether the combination of the three traces' components, S_{total}, S_μ and S_{em},attained better results than the usage of two components separately. In addition, SVMs results using traces or computed variables are presented in order to compare them.

Lastly, detailed in Section 5.3, a performance comparison was performed when adding the information of external variables to both CNNs and SVMs algorithms.

Adam optimizer algorithm was chosen for the experiments performed in Section 5.1, with a learning rate equal to 0.01 and without weight decay. The algorithms were trained during 40 epochs both for one and three SDs.

SGD optimizer algorithm was chosen for the experiments performed in Sections 5.2 and 5.3, with a learning rate equal to 0.01, a value of weight decay equal to 0.0001 and a momentum of 0.9. The algorithms were trained during 15 epochs when using the information of one SD or three SDs.

5.1 Photon vs Hadron Classification using S_{total} or S_μ

These results were presented in the International Work-Conference on Artificial Neural Networks (IWANN) [58].

Tables 5.1 and 5.2 show the results obtained for each FFNN and CNN topologies, declared in Section 4.3.2. Accuracy using two alternatives for input data (muonic signal and total signal) is presented in table 5.1. Table 5.2 shows the sensitivity and specificity for each topology.

Table 5.1: Mean Accuracy for each FFNN and CNN topologies using one and three traces. Results are presented for S_{total} and S_μ both for test and validation sets.

One trace		
S_{total}	**Val. Acc.%**	**Test Acc.%**
FFNN1	82.11(0.27)	82.12(0.65)
FFNN2	84.18(0.17)	84.14(0.43)
CNN	88.74(0.14)	87.23(0.22)
S_μ	**Val. Acc.%**	**Test Acc.%**
FFNN1	78.55(0.37)	78.15(0.074)
FFNN12	78.61(0.37)	77.92(0.074)
CNN	79.05(0.34)	79.05(0.12)

Three traces		
S_{total}	**Val Acc.%**	**Test Acc.%**
FFNN1	86.20(0.68)	84.03(1.40)
FFNN2	88.28(0.58)	85.79(0.80)
CNN	90.95(0.58)	88.48(0.02)
S_μ	**Val. Acc.%**	**Test Acc.%**
FFNN1	86.98(0.65)	87.08(0.41)
FFNN2	87.86(0.95)	86.27(0.41)
CNN	88.02(0.60)	86.07(0.34)

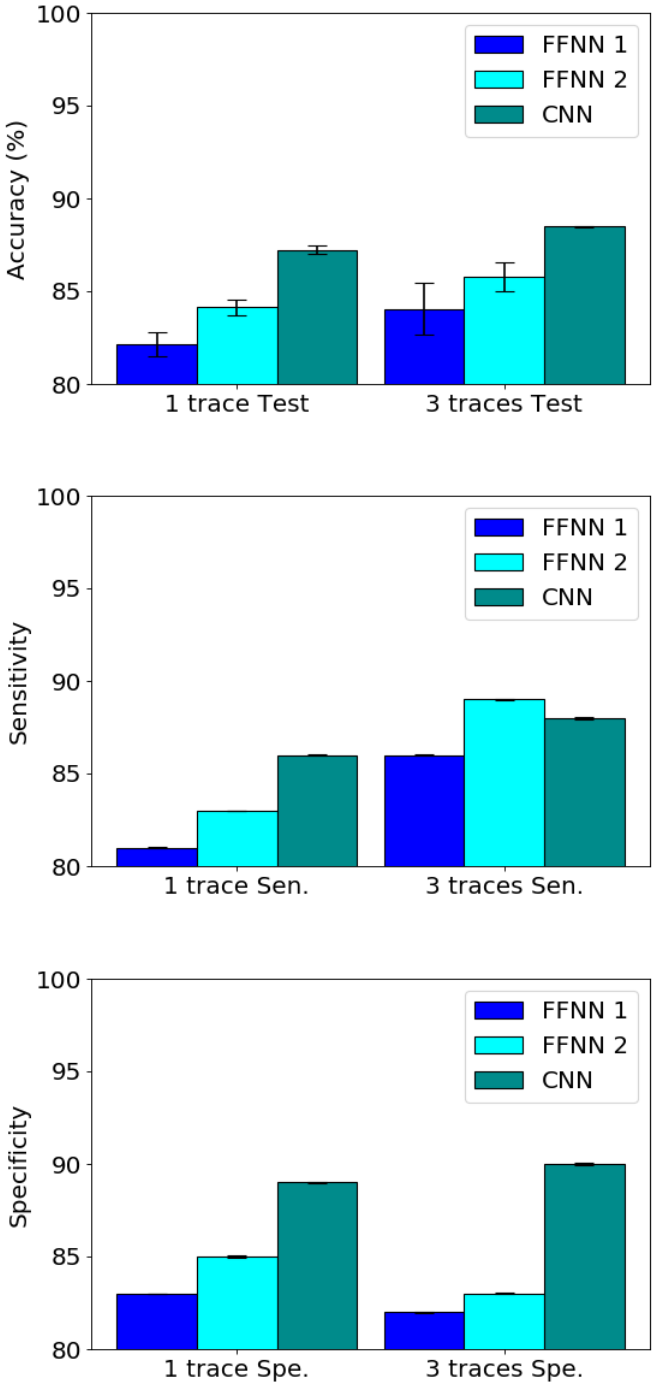
Table 5.2: Mean Sensitivity and specificity for each FFNN and CNN topologies using one and three traces in the test set. Results are presented for S_{total} and for S_{μ}.

One trace		
S_{total}	**Sensitivy%**	**Specificity%**
FFNN1	81(0.9)	83(0.8)
FFNN2	83(0.7)	85(0.7)
CNN	86(1.5)	88(0.9)
S_{μ}	**Sensitivy%**	**Specificity%**
FFNN1	76(0.5)	80(0.6)
FFNN2	77.5(0.7)	78(1)
CNN	76(0.9)	80(1.2)
Three traces		
S_{total}	**Sensitivy%**	**Specificity%**
FFNN1	86(2)	82(3)
FFNN2	89(2)	83(3)
CNN	88(6)	90(4)
S_{μ}	**Sensitivy%**	**Specificity%**
FFNN1	86(0.9)	88(1.7)
FFNN2	87(2)	86(4)
CNN	82(5)	94(3)

As it can be observed, best accuracy is obtained using the aggregation of three traces per event for each architecture (see table 5.1). The second architecture for the FFNN, the one with two two layers of one hundred neurons each, improves results over the first FFNN architecture. However, none of them surpasses the results obtained with the CNN using the information of three traces. Results obtained using total signal instead of muonic signal are superior for the CNN architecture. Nonetheless, for FFNN architectures superior results are obtained with muonic signal using three traces, but very close to results obtained with total signal. FFNNs using one trace have a superior performance when using total signal.

In relation to the sensitivity and specificity(see table 5.2), results are superior in the CNN architecture. Again, using three traces improves results over using the information of one trace only. This was expected, since using three traces add more information of the event to classify. Also using total signal instead of muonic signal led to better results in both sensitivity and specificity. A graphical representation of the results for each architecture can be observed in Figure 5.1.

Figure 5.1: In Figure 5.1a mean accuracy for all architectures using total signal is presented. As observed, CNN outperforms FFNN architectures' results. A baseline model, where we could use a weak classifier which always predict the majority class, could obtain near 50% of accuracy, therefore all the three presented models here outperform this weak classifier. In 5.1b mean sensitivity in the test set using total signal is presented. In Figure 5.1c mean specificity in the test set using total signal is presented. Y axis has been ranged between 80% and 100% in order to see the differences between different results.



5.2 Classification results obtained when using traces' components

Tables 5.3 and 5.4 show results for CNN using S_T and SVM using computed variables. Table 5.5 shows the Specificity and Sensitivity in the test set for each algorithm described above.

The combination of all three traces' components is denoted as S_T where $S_T = S_{total} + S_{\mu} + S_{em}$. Computed variables are denoted as V_c. They were described in Section 3.1.

Table 5.3: Mean Accuracy for CNNs and SVMs using one and three SDs for binary classification. Results are presented for both for test and validation sets. The combination of all three traces is denoted as S_T where $S_T = S_{total} + S_{\mu} + S_{em}$. Computed variables are denoted as V_c.

Photon vs Hadron		
One trace		
	Val. Acc.%	**Test Acc.%**
CNN S_T	92.70(0.21)	92.77(0.27)
SVM V_c	90.28(0.001)	90.42(0.002)
Three traces		
	Val Acc.%	**Test Acc.%**
CNN S_T	96.94(0.6)	97.09(0.28)
SVM V_c	95.99(0.005)	95.78(0.005)

Table 5.4: Mean Accuracy for CNNs and SVMs using one and three SDs for multi-class classification. Results are presented for both for test and validation sets. The combination of all three traces is denoted as S_T where $S_T = S_{total} + S_{\mu} + S_{em}$. Computed variables are denoted as V_c.

Five classes		
One trace		
	Val. Acc.%	**Test Acc.%**
CNN S_T	41.74(0.71)	41.92(0.63)
SVM V_c	40.61(0.002)	40.70(0.001)
Three traces		
	Val Acc.%	**Test Acc.%**
CNN S_T	44.54(0.44)	44.94(1.08)
SVM V_c	43.44(0.007)	43.43(0.01)

Table 5.5: Mean Sensitivity and specificity for CNNs and SVM using one and three SDs in the test set for binary classification. The combination of all three traces is denoted as S_T where $S_T = S_{total} + S_{\mu} + S_{em}$. Computed variables are denoted as V_c.

Photon vs Hadron		
One trace		
	Sensitivy%	**Specificity%**
CNN S_T	92.87(0.011)	92.67(0.016)
SVM V_c	93(0.003)	87.86(0.001)
Three traces		
	Sensitivy%	**Specificity%**
CNN S_T	96.63(0.008)	97.51(0.006)
SVM V_c	96.45(0.008)	95.09(0.003)

Binary classification

Based on the results presented in Table 5.3, we can observed that the performance obtained by the CNN architecture using S_T is superior to those obtained by using the SVM with the computed variables. This points out to that the model proposed is able to extract more information from the particle that caused the shower than manually extracted features given by the physicists for this same goal.

Moreover, as observed in the previous section, the results obtained when combining the information of more than one SD improves results over the information of each SD separately. CNN using S_T outperform the results obtained by the SVM when using the information from three SDs as well. These results also show that the combination of the different components of the trace benefits the performance when classifying the primary causing the shower. Similar conclusions can be drawn based on the results obtained in Table 5.5, which shows the sensitivy and specificity of each model.

In Figure 5.2 a graphical comparison between the results obtained with the CNN and the SVM for both one and three SDs can be observed. In Figure 5.3 the confusion matrix for the CNN experiment using S_T can be observed for one and three SDs.

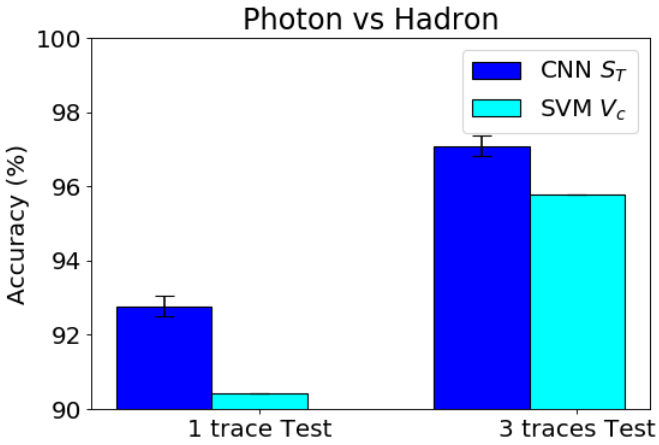
Multi-class classification

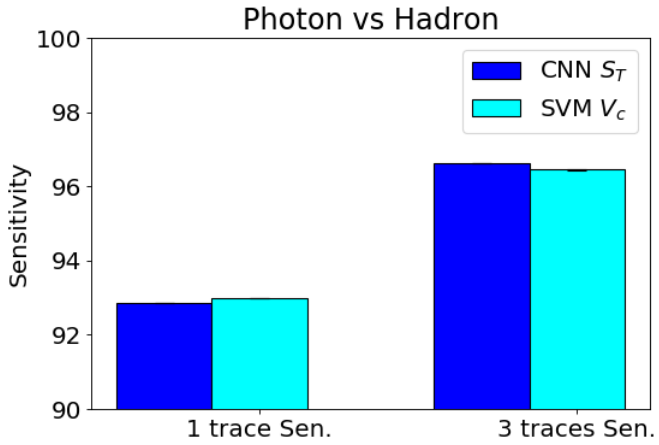
Regarding the results obtained for multi-class classification, better performance is obtained when CNNs. It must be mentioned that these are preliminary results, and the CNN architecture could be tuned in order to get the most out of the data. Nevertheless, results using V_c with SVM in this

problem point out that not real relevant improvement could be expected for this multi-class problem and under the given experimentation. Even though, as it has been observed before, the automatic feature extraction from the traces performed by CNNs outperform the variables extracted from them by physicists.

In Figure 5.4 the confusion matrix for the CNN experiment using S_T can be observed for one and three SDs.

Figure 5.2: Graphical results obtained for the Photon vs Hadron classification. In Figure 5.2a mean accuracy in the test set for both CNN and SVM is presented. As observed, CNN outperforms SVM results for both one and three SDs. A baseline model, where we could use a weak classifier which always predict the majority class, could obtain near 50% of accuracy, therefore all the three presented models here outperform this weak classifier. In Figure 5.2b mean sensitivity in the test set for both CNN and SVM is presented. In Figure 5.2c mean specificity in the test set for both CNN and SVM is presented. Y axis has been ranged between 90% and 100% in order to see the differences between different results.





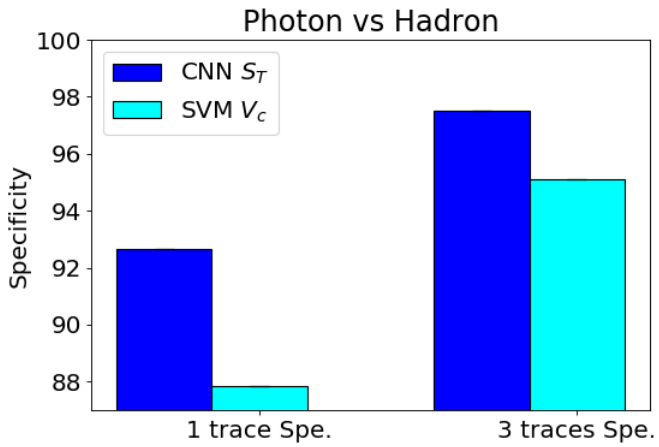


Figure 5.3: Confusion matrix obtained for CNN using S_T, for binary classification, when using the information of one SD. Figure 5.3a, and three SDs in Figure 5.3b.

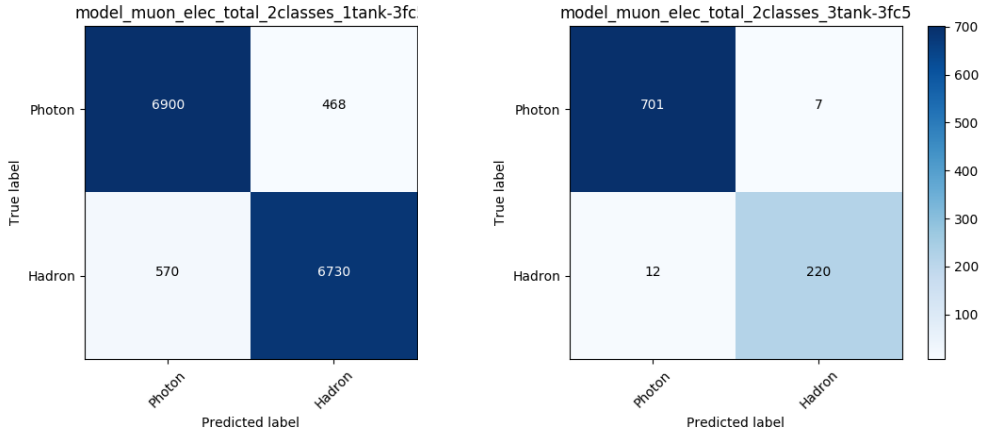
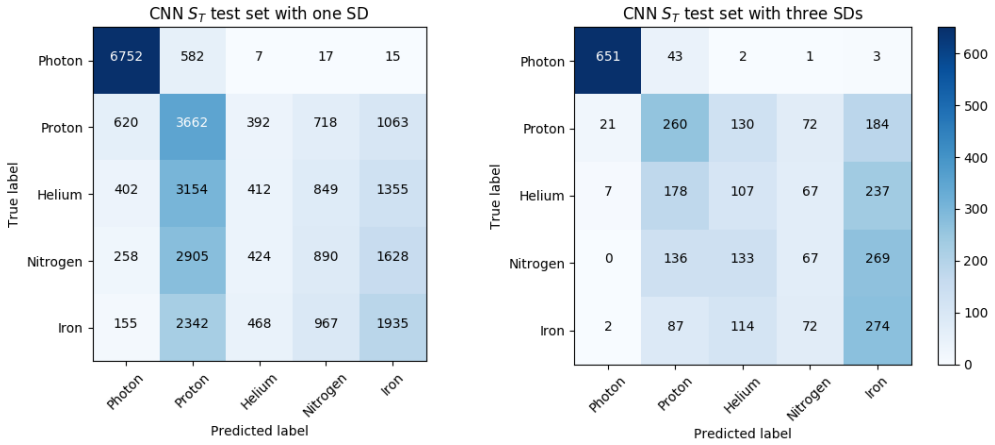


Figure 5.4: Confusion matrix obtained for CNN using S_T, for multi-class classification, when using the information of one SD. Figure 5.4a, and three SDs in Figure 5.4b.



5.3 Classification results obtained when using traces' components, computed and external variables combined

In Tables 5.6 and 5.7 the mean accuracy in the validation and test sets for binary classification and multi-class classification can be observed, when using the trace's information in addition with variables obtained from the traces by physicists, V_c, and information measured from the event, V_e. Which variables correspond to each group was explained in Section 3.1. Table 5.8 shows the mean sensitivity and specificity in the test set.

Table 5.6: Mean Accuracy for CNNs and SVMs using one and three SDs for binary classification. Results are presented for both for test and validation sets. The combination of all three traces is denoted as S_T where $S_T = S_{total} + S_{\mu} + S_{em}$. Computed variables are denoted as V_c. External variables are denoted as V_e. Reader can refer to Table 5.3 to recall classification accuracy when using CNN S_T and SVM V_c.

Photon vs Hadron		
One SD		
	Val. Acc.%	**Test Acc.%**
CNN S_T + V_c	92.43(0.47)	92.38(0.29)
CNN S_T + V_e	97.02(0.11)	97.05(0.11)
CNN S_T + V_c + V_e	96.65(0.75)	96.56(0.68)
SVM V_e	91.60(0.002)	91.80(0.002)
SVM V_c + V_e	97.18(0.008)	97.08(0.001)
Three SDs		
	Val Acc.%	**Test Acc.%**
CNN S_T + V_c	97.98(0.38)	98.00(0.24)
CNN S_T + V_e	98.27(0.38)	98.64(0.33)
CNN S_T + V_c + V_e	98.60(0.16)	98.87(0.17)
SVM V_e	96.83(0.006)	97.23(0.004)
SVM V_c + V_e	98.34(0.001)	98.19(0.001)

Table 5.7: Mean Accuracy for CNNs and SVMs using one and three SDs for multi-class classification. Results are presented for both for test and validation sets. The combination of all three traces is denoted as S_T where $S_T = S_{total} + S_\mu + S_{em}$. Computed variables are denoted as V_c. External variables are denoted as V_e. Reader can refer to Table 5.4 to recall classification accuracy when using CNN S_T and SVM V_c.

Five classes		
One trace		
	Val. Acc.%	**Test Acc.%**
CNN $S_T + V_c$	42.70(0.28)	42.81(0.17)
CNN $S_T + V_e$	53.34(0.16)	53.66(0.13)
CNN $S_T + V_c + V_e$	52.74(0.59)	53.17(0.65)
SVM V_e	49.57(0.001)	49.65(0.002)
SVM $V_c + V_e$	52.84(0.002)	53.11(0.003)
Three traces		
	Val. Acc.%	**Test Acc.%**
CNN $S_T + V_c$	44.89(0.80)	45.33(1.33)
CNN $S_T + V_e$	52.52(1.21)	53.05(0.43)
CNN $S_T + V_c + V_e$	52.65(1.40)	53.34(1.29)
SVM V_e	49.57(0.001)	49.65(0.002)
SVM $V_c + V_e$	50.16(0.006)	50.82(0.005)

Table 5.8: Mean Sensitivity and specificity for CNNs and SVM using one and three SDs in the test set for binary classification. The combination of all three traces is denoted as S_T where $S_T = S_{total} + S_{\mu} + S_{em}$. Computed variables are denoted as V_c. External variables are denoted as V_e.

Photon vs Hadron		
One trace		
	Sensitivy%	**Specificity%**
CNN $S_T + V_c$	89.20(0.013)	95.58(0.009)
CNN $S_T + V_e$	96.64(0.008)	97.46(0.007)
CNN $S_T + V_c + V_e$	92.83(0.015)	96.87(0.06)
SVM V_e	93.76(0.003)	89.86 (0.004)
SVM $V_c + V_e$	97.27(0.003)	96.88(0.001)
Three traces		
	Sensitivy%	**Specificity%**
CNN $S_T + V_c$	97.47(0.011)	98.51(0.01)
CNN $S_T + V_e$	98.63(0.003)	98.65(0.004)
CNN $S_T + V_c + V_e$	99(0.001)	98.74(0.003)
SVM V_e	96.51(0.009)	97.96(0.002)
SVM $V_c + V_e$	98.17(0.002)	98.22(0.002)

Binary classification

When it comes to binary classification different conclusions can be drawn from Table 5.6.

Using S_T along with V_c worsens the results obtained with S_T alone, as observed in Table 5.3. Therefore, it seems that adding these features extracted by physicists to the features extracted by the CNN from the traces worsen the model's generalization.

Using V_e by itself for classification does not improve results obtained when using traces' information. Reader could observed this in Tables 5.3 and 5.6. Henceforth, the information from the traces seems crucial in order to understand which has been the particle that have generated the air shower.

However, results are improved when using the information from the traces plus the addition of V_e. Although, V_e can not always be measured, what gives results obtained in Table 5.3 more importance, where CNNs obtained better results.

In addition, it seems that the classifier's performance, both for CNNs and SVMs, reach its limit when adding the information of V_e, as it can be observed for results obtained in Table 5.6.

A graphical comparison of the results described above can be observed

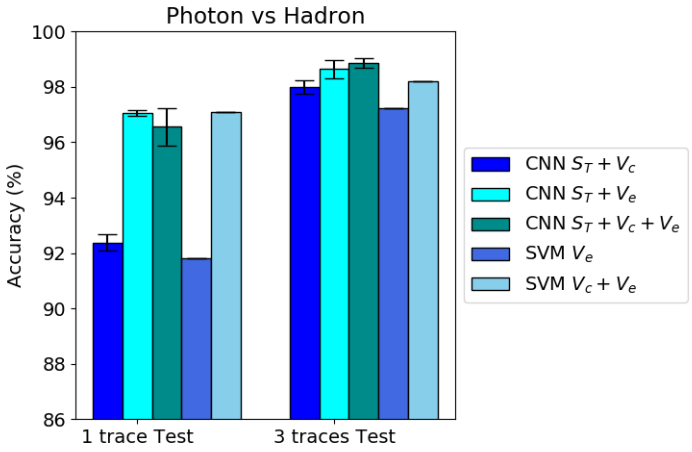
in Figure 5.5. In Figure 5.2 a graphical comparison between the results obtained with the CNN and the SVM for both one and three SDs can be observed. In Figure 5.6 the confusion matrix for the CNN experiment using $S_T + V_c + V_e$ can be observed for one and three SDs.

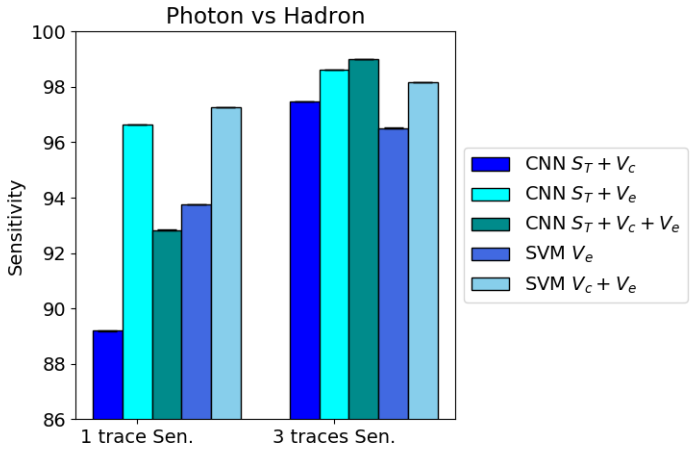
Multi-class classification

Regarding the results obtained for multi-class classification, same conclusions can be drawn as previously explained. CNNs outperform results obtained by SVM with the direct comparison between CNN $S_T + V_e$ and SVM $V_c + V_e$. However in this case the usage of the external variables with one SD provides a better performance than before. This could be due to the fact that the external variables represent better the differentiation between the fifth particles than when we are grouping four of them into a group. AS it been said before, these are preliminary results, and the CNN architecture could be tuned in order to get the most out of the data.

In Figure 5.7 the confusion matrix for the CNN experiment using $S_T + V_c + V_e$ can be observed for one and three SDs.

Figure 5.5: Graphical results obtained for the Photon vs Hadron classification. In Figure 5.5a mean accuracy in the test set for all the variations of CNN and SVM is presented. In Figure 5.5b mean sensitivity in the test set for both CNN and SVM is presented. In Figure 5.5c mean specificity in the test set for both CNN and SVM is presented. Y axis has been ranged between 86%-90% and 100% in order to see the differences between different results.





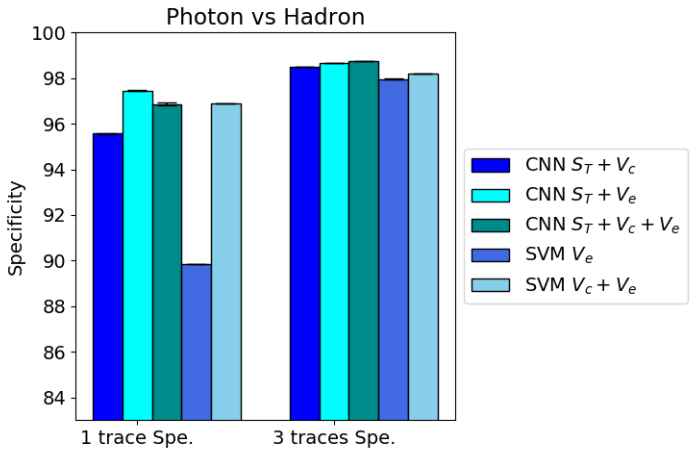


Figure 5.6: Confusion matrix obtained for CNN using $S_T + V_c + V_e$, for binary classification, when using the information of one SD. Figure 5.6a, and three SDs in Figure 5.6b.

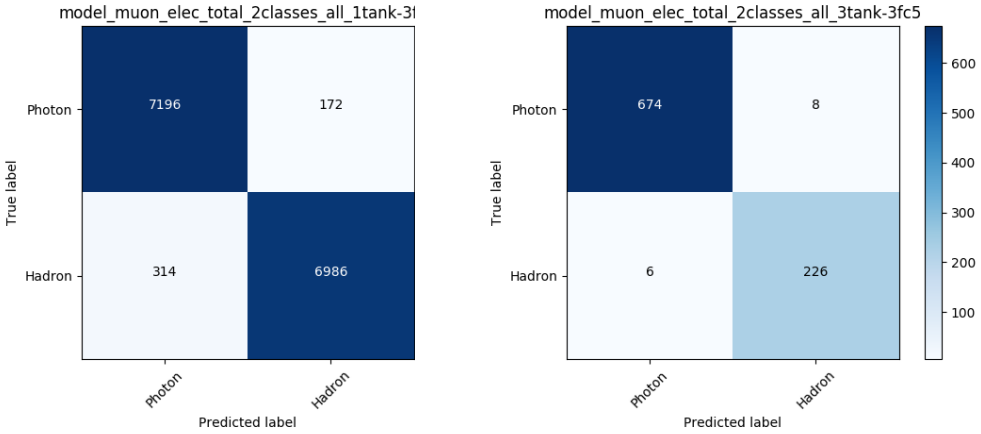
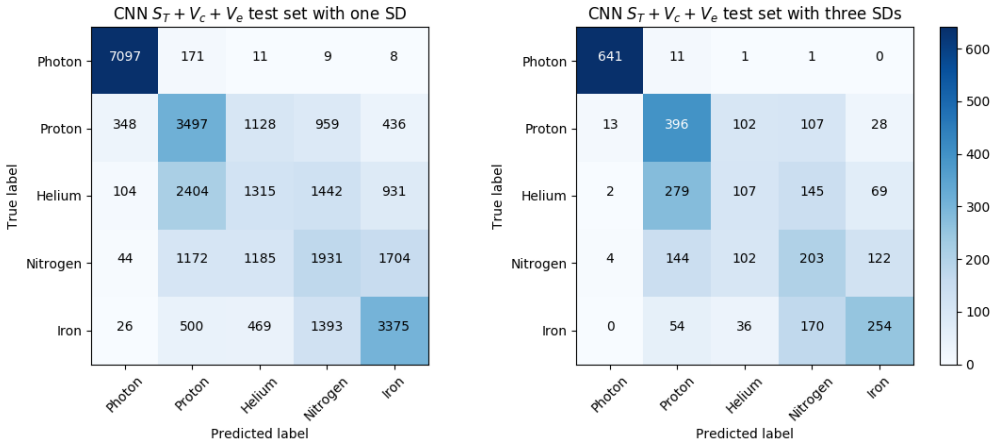


Figure 5.7: Confusion matrix obtained for CNN using $S_T+V_c+V_e$, for multi-class classification, when using the information of one SD. Figure 5.7a, and three SDs in Figure 5.7b.



Chapter 6

Conclusions and future work

Experiments performed have answered the questions that were stated in the aims of this thesis, as well as some others arisen after the observation of the experiments' results.

Firstly it has been shown in Sections 5.1, 5.2 and 5.3 that tackling the problem of classifying particles that have generated extensive air showers is possible. Very successful results are obtained both for the CNNs and SVMs when classifying between Photon and Hadron.

Continuing with the performance of the models, it has been shown that CNNs are able to properly extract the spatial information intrinsic in the traces and using it for the classification. CNNs have been remarkably successful when it comes to extracting important information for the classification, as it can be observed in Tables 5.3 and 5.6 and in Figures 5.2 and 5.5.

In Section 5.2 experiments were performed in order to answer the question regarding if using the automatically extracted features from the traces by the CNN could compete with the features extracted by hand by physicists. It has been shown in Table 5.3 and Figure 5.2 that the features extracted by the CNN can outperform the results obtained by the features proposed by physicists.

It has been shown in Tables 5.1, 5.3 and 5.6, that the combination of information between measurements of different SDs hit by the same event improved remarkably the classification's performance.

Experiments where external variables were added to the features extracted from the traces were performed in Section 5.3. They shown that using external variables in combination with the features extracted from traces improved the performance of the CNN over using features from traces isolatedly. However, they could not always be measured. Therefore,the re-

sults obtained in Section 5.2 are more important. Experiments using only external variables inputs were performed. However, as it been shown, this information isoletedly is not sufficient in order to understand which has been the particle that caused the extensive air shower. Furthermore, they could not always be measured.

To sum up, it has been shown that Deep Learning algorithms are able to extract information from the gathered traces. This information can be applied to the classification problem, obtaining fantastic results. In addition, the combination of trace's information with external measurements improves the models' results, achieving the best performance when they are included. However, they could not always be measured and worse results are obtained when using the external variables only. Given the intrinsic difficulty for multi-class classification, the performance obtained for this task is not the best one, showing that additional information is needed to improve the discerning capabilities of the evaluated types of models in the classification of more specific types of hadrons.

6.1 Future work

In order to improve the architectures' proposed, a genetic algorithm is wanted to be used for optimizing and making the architecture as simple as possible. In addition, this could help to improve the understanding of the problem and to discern if the architectures presented in this work are the optimal ones. The genetic algorithm could be also optimized for multi-class classification.

Once the model has been optimized, the interpretability of the features obtained from the traces by CNNs wanted to be studied. This was a task wanted to be done, but,unfortunately, with all the work that was required for the knowledge extraction portion of this master thesis, it will be performed in the future.

Applying the algorithms proposed in this work to actual measurements would be of great interest. Both the architectures presented in this work plus the ones obtained by the usage of genetic algorithms could improve physicists' knowledge about the nature of cosmic rays.

In addition, developing a pipeline for newer observatories that are under construction right know is a proposed goal. Given the right tools, as the algorithms presented in this work, the work of physicist in this area could be hugely accelerated. Once a new event has been recorded, the information could go through the algorithm to obtain a label, and compare it with expert knowledge at the moment. Later, this newer events could be used to improve the performance of the network.

Chapter 7

Appendix: Master Thesis's Budget

In this appendix the budget needed for the master thesis is going to be itemise.

Firstly, for the experiments a Graphic Processing Unit (GPU) is needed. One that is affordable and powerful enough is the Nvidia GeForce GTX 1080 [65]. Also, different components for a sever are need, such a CPU, for example a Intel Xeon [66], and at least 8 GB of RAM.

For the Junior Researcher, working hours can be set in the morning, since the rest of the day was set for master's classes. Around 180 hours have been used for this thesis by the Junior Researcher. When it comes to the two Senior Researchers they worked for around 30 hours each.

The budget can be observed in Table 7.1.

Table 7.1: Master thesis's budget in euros.

	Price
Equipment (GPU: Nvidia 1080 GTX)	600 €
Server (Intel Xeon, 16 GB RAM)	500 €
Junior Researcher (35 €/hour)	6300 €
2 Senior Researchers (45 €/hour)	2700 €
Total	10100 €

Bibliography

[1] L. Bret. Auger reveals subtlety in cosmic rays. https://www.symmetrymagazine.org/article/november-2014/auger-reveals-subtlety-in-cosmic-rays, 2014. [Online; accessed 3 de Mayo 2019].

[2] Dariusz Gora, Pierre Auger Collaboration, et al. The pierre auger observatory: review of latest results and perspectives. Universe, 4(11):128, 2018.

[3] Argonne National Laboratory. Cherenkov radiation glowing in the core of the Advanced Test Reactor. https://en.wikipedia.org/wiki/Cherenkov_radiation#/media/File:Advanced_Test_Reactor.jpg, 2019. [Online; accessed 6 de Mayo 2019].

[4] Tony Fischetti. The diagonal decision boundary. https://www.oreilly.com/library/view/data-analysis-with/9781788393720/7ef9ff0a-000c-44e1-ad6a-d352e0744c74.xhtml, 2018. [Online; accessed 6 de Mayo 2019].

[5] Sagar Sharma. What the Hell is a Perceptron? https://towardsdatascience.com/what-the-hell-is-perceptron-626217814f53, 2017. [Online; accessed 8 de Mayo 2019].

[6] Gengiskanhg. Red Neuronal Multicapa. https://es.wikipedia.org/wiki/Perceptr%C3%B3n_multicapa#/media/File:RedNeuronalArtificial.png, 2019. [Online; accessed 8 de Mayo 2019].

[7] Unkown. Rectifier(neural networks). https://en.wikipedia.org/wiki/Rectifier_(neural_networks)#/media/File:Rectifier_and_softplus_functions.svg, 2017. [Online; accessed 19 June 2019].

[8] Qef. Sigmoid Function. https://en.wikipedia.org/wiki/Sigmoid_function#/media/File:Logistic-curve.svg, 2008. [Online; accessed 19 June 2019].

[9] Peter Roelants. Softmax Function. https://peterroelants.github.io/posts/cross-entropy-softmax/, Unknown. [Online; accessed 19 June 2019].

[10] Towards Data Science. Applied Deep Learning Part 4. https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2, 2014. [Online; accessed 18 June 2019].

[11] Andrej Kaparthy. CS231n Convolutional Neural Networks for Visual Recognition. https://cs231n.github.io/convolutional-networks/, 2017. [Online; accessed 18 June 2019].

[12] Ghiles. Overfitting. https://en.wikipedia.org/wiki/Overfitting#/media/File:Overfitted_Data.png, 2016. [Online; accessed 19 June 2019].

[13] Geoffrey E Hinton, Alexander Krizhevsky, Ilya Sutskever, and Nitish Srivastva. System and method for addressing overfitting in a neural network, August 2 2016. US Patent 9,406,017.

[14] Shiyu Ji. Kernel trick idea. https://en.wikipedia.org/wiki/Kernel_method#/media/File:Kernel_trick_idea.svg, 2017. [Online; accessed 18 June 2019].

[15] Sklearn. SVMs. https://scikit-learn.org/stable/modules/svm.html, 2019. [Online; accessed 18 June 2019].

[16] John Linsley, Livio Scarsi, and Bruno Rossi. Extremely energetic cosmic-ray event. Physical Review Letters, 6(9):485, 1961.

[17] Antoine Letessier-Selvon and Todor Stanev. Ultrahigh energy cosmic rays. Reviews of modern physics, 83(3):907, 2011.

[18] Ralf Ulrich, Ralph Engel, and Michael Unger. Hadronic multiparticle production at ultrahigh energies and extensive air showers. Physical Review D, 83(5):054026, 2011.

[19] Pierre Auger, Paul Ehrenfest, Roland Maze, Jean Daudin, and Robley A Fréon. Extensive cosmic-ray showers. Reviews of modern physics, 11(3-4):288, 1939.

[20] Alan A Watson. The discovery of cherenkov radiation and its use in the detection of extensive air showers. Nuclear Physics B-Proceedings Supplements, 212:13–19, 2011.

[21] Martin Erdmann, Jonas Glombitza, and David Walz. A deep learning-based reconstruction of cosmic ray-induced air showers. Astroparticle Physics, 97:46–53, 2018.

[22] Nicholas Choma, Federico Monti, Lisa Gerhardt, Tomasz Palczewski, Zahra Ronaghi, Prabhat Prabhat, Wahid Bhimji, Michael Bronstein, Spencer Klein, and Joan Bruna. Graph neural networks for icecube signal classification. In 2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA), pages 386–391. IEEE, 2018.

[23] Mirco Huennefeld. Deep learning in physics exemplified by the reconstruction of muon-neutrino events in icecube. PoS, page 1057, 2017.

[24] A. Guillén, A. Bueno, J.M. Carceller, J.C. Martínez-Velázquez, G. Rubio, C.J. Todero Peixoto, and P. Sanchez-Lucas. Deep learning techniques applied to the physics of extensive air showers. Astroparticle Physics, 2019.

[25] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. The elements of statistical learning, volume 1. Springer series in statistics New York, 2001.

[26] Olivier Chapelle, Bernhard Scholkopf, and Alexander Zien. Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews]. IEEE Transactions on Neural Networks, 20(3):542–542, 2009.

[27] Jürgen Schmidhuber. Deep learning in neural networks: An overview. Neural networks, 61:85–117, 2015.

[28] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. Psychological review, 65(6):386, 1958.

[29] Steven J Cooper. Donald o. hebb's synapse and learning rule: a history and commentary. Neuroscience & Biobehavioral Reviews, 28(8):851–874, 2005.

[30] David E Rumelhart, Geoffrey E Hinton, Ronald J Williams, et al. Learning representations by back-propagating errors. Cognitive modeling, 5(3):1, 1988.

[31] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. Neural networks, 2(5):359–366, 1989.

[32] Kunihiko Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. Biological cybernetics, 36(4):193–202, 1980.

[33] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11):2278–2324, 1998.

[34] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. nature, 521(7553):436, 2015.

[35] Christopher G Harris, Mike Stephens, et al. A combined corner and edge detector. In Alvey vision conference, volume 15, pages 10–5244. Citeseer, 1988.

[36] Irwin Sobel and Gary Feldman. A 3x3 isotropic gradient operator for image processing. a talk at the Stanford Artificial Project in, pages 271–272, 1968.

[37] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. Deep learning. MIT press, 2016.

[38] Dan Cireşan, Ueli Meier, and Jürgen Schmidhuber. Multi-column deep neural networks for image classification. arXiv preprint arXiv:1202.2745, 2012.

[39] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In Proceedings of the 27th international conference on machine learning (ICML-10), pages 807–814, 2010.

[40] Kazuyuki Hara, Daisuke Saito, and Hayaru Shouno. Analysis of function of rectified linear unit used in deep learning. In 2015 International Joint Conference on Neural Networks (IJCNN), pages 1–8. IEEE, 2015.

[41] Herbert Robbins and Sutton Monro. A stochastic approximation method. The annals of mathematical statistics, pages 400–407, 1951.

[42] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980, 2014.

[43] Quoc V Le, Jiquan Ngiam, Adam Coates, Abhik Lahiri, Bobby Prochnow, and Andrew Y Ng. On optimization methods for deep learning. In Proceedings of the 28th International Conference on International Conference on Machine Learning, pages 265–272. Omnipress, 2011.

[44] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. Deep Learning. MIT Press, 2016. `http://www.deeplearningbook.org`.

[45] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167, 2015.

[46] Mercedes Fernández-Redondo and Carlos Hernández-Espinosa. Weight initialization methods for multilayer feedforward. In ESANN, pages 119–124, 2001.

[47] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In Proceedings of the thirteenth international conference on artificial intelligence and statistics, pages 249–256, 2010.

[48] Christopher M Bishop. Pattern recognition and machine learning. springer, 2006.

[49] Xavier Bertou, PS Allison, C Bonifazi, P Bauleo, CM Grunfeld, M Aglietta, F Arneodo, D Barnhill, JJ Beatty, NG Busca, et al. Calibration of the surface array of the pierre auger observatory. Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment, 568(2):839–846, 2006.

[50] JG Gonzalez, IceCube Collaboration, et al. Measurement of the muon content of air showers with icetop. In Journal of Physics: Conference Series, volume 718, page 052017. IOP Publishing, 2016.

[51] Balázs Kégl. Measurement of the muon signal using the temporal and spectral structure of the signals in surface detectors of the pierre auger observatory. In 33rd International Cosmic Ray Conference (ICRC2013), 2013.

[52] D Garcia-Gamez. Measurement of atmospheric production depths of muons with the pierre auger observatory. In EPJ Web of Conferences, volume 53, page 04008. EDP Sciences, 2013.

[53] I Valiño, Pierre Auger Collaboration, et al. Measurements of the muon content of air showers at the pierre auger observatory. In Journal of Physics: Conference Series, volume 632, page 012103. IOP Publishing, 2015.

[54] Sergey Ostapchenko. Monte carlo treatment of hadronic interactions in enhanced pomeron scheme: Qgsjet-ii model. Physical Review D, 83(1):014018, 2011.

[55] D. Heck, J. Knapp, J. N. Capdevielle, G. Schatz, and T. Thouw. CORSIKA: a Monte Carlo code to simulate extensive air showers. February 1998.

[56] A. Aab, P. Abreu, M. Aglietta, I. Al Samarai, I. F. M. Albuquerque, I. Allekotte, A. Almela, J. Alvarez Castillo, J. Alvarez-Muñiz, G. A. Anastasi, L. Anchordoqui, B. Andrada, S. Andringa, C. Aramo, F. Arqueros, N. Arsene, H. Asorey, P. Assis, J. Aublin, G. Avila, A. M. Badescu, A. Balaceanu, F. Barbato, R. J. Barreira Luz, J. J. Beatty, K. H. Becker, J. A. Bellido, C. Berat, M. E. Bertaina, X. Bertou, P. L.

Biermann, J. Biteau, S. G. Blaess, A. Blanco, J. Blazek, C. Bleve, M. Boháčová, D. Boncioli, C. Bonifazi, N. Borodai, A. M. Botti, J. Brack, I. Brancus, T. Bretz, A. Bridgeman, F. L. Briechle, P. Buchholz, A. Bueno, S. Buitink, M. Buscemi, K. S. Caballero-Mora, L. Caccianiga, A. Cancio, F. Canfora, L. Caramete, R. Caruso, A. Castellina, F. Catalani, G. Cataldi, L. Cazon, A. G. Chavez, J. A. Chinellato, J. Chudoba, R. W. Clay, A. Cobos, R. Colalillo, A. Coleman, L. Collica, M. R. Coluccia, R. Concei ç ao, G. Consolati, F. Contreras, M. J. Cooper, S. Coutu, C. E. Covault, J. Cronin, S. D'Amico, B. Daniel, S. Dasso, K. Daumiller, B. R. Dawson, R. M. de Almeida, S. J. de Jong, G. De Mauro, J. R. T. de Mello Neto, I. De Mitri, J. de Oliveira, V. de Souza, J. Debatin, O. Deligny, M. L. Díaz Castro, F. Diogo, C. Dobrigkeit, J. C. D'Olivo, Q. Dorosti, R. C. dos Anjos, M. T. Dova, A. Dundovic, J. Ebr, R. Engel, M. Erdmann, M. Erfani, C. O. Escobar, J. Espadanal, A. Etchegoyen, H. Falcke, J. Farmer, G. Farrar, A. C. Fauth, N. Fazzini, F. Fenu, B. Fick, J. M. Figueira, A. Filipčič, O. Fratu, M. M. Freire, T. Fujii, A. Fuster, R. Gaior, B. García, D. Garcia-Pinto, F. Gaté, H. Gemmeke, A. Gherghel-Lascu, P. L. Ghia, U. Giaccari, M. Giammarchi, M. Giller, D. Głas, C. Glaser, G. Golup, M. Gómez Berisso, P. F. Gómez Vitale, N. González, A. Gorgi, P. Gorham, A. F. Grillo, T. D. Grubb, F. Guarino, G. P. Guedes, R. Halliday, M. R. Hampel, P. Hansen, D. Harari, T. A. Harrison, J. L. Harton, A. Haungs, T. Hebbeker, D. Heck, P. Heimann, A. E. Herve, G. C. Hill, C. Hojvat, E. Holt, P. Homola, J. R. Hörandel, P. Horvath, M. Hrabovský, T. Huege, J. Hulsman, A. Insolia, P. G. Isar, I. Jandt, J. A. Johnsen, M. Josebachuili, J. Jurysek, A. Kääpä, O. Kambeitz, K. H. Kampert, B. Keilhauer, N. Kemmerich, E. Kemp, J. Kemp, R. M. Kieckhafer, H. O. Klages, M. Kleifges, J. Kleinfeller, R. Krause, N. Krohm, D. Kuempel, G. Kukec Mezek, N. Kunka, A. Kuotb Awad, B. L. Lago, D. LaHurd, R. G. Lang, M. Lauscher, R. Legumina, M. A. Leigui de Oliveira, A. Letessier-Selvon, I. Lhenry-Yvon, K. Link, D. Lo Presti, L. Lopes, R. López, A. López Casado, R. Lorek, Q. Luce, A. Lucero, M. Malacari, M. Mallamaci, D. Mandat, P. Mantsch, A. G. Mariazzi, I. C. Mariş, G. Marsella, D. Martello, H. Martinez, O. Martínez Bravo, J. J. Masías Meza, H. J. Mathes, S. Mathys, J. Matthews, J. A. J. Matthews, G. Matthiae, E. Mayotte, P. O. Mazur, C. Medina, G. Medina-Tanco, D. Melo, A. Menshikov, K.-D. Merenda, S. Michal, M. I. Micheletti, L. Middendorf, L. Miramonti, B. Mitrica, D. Mockler, S. Mollerach, F. Montanet, C. Morello, M. Mostafá, A. L. Müller, G. Müller, M. A. Muller, S. Müller, R. Mussa, I. Naranjo, L. Nellen, P. H. Nguyen, M. Niculescu-Oglinzanu, M. Niechciol, L. Niemietz, T. Niggemann, D. Nitz, D. Nosek, V. Novotny, L. Nožka, L. A. Núñez, L. Ochilo, F. Oikonomou, A. Olinto, M. Palatka, J. Pallotta, P. Papenbreer,

G. Parente, A. Parra, T. Paul, M. Pech, F. Pedreira, J. Pekala, R. Pelayo, J. Peña Rodriguez, L. A. S. Pereira, M. Perlin, L. Perrone, C. Peters, S. Petrera, J. Phuntsok, R. Piegaia, T. Pierog, M. Pimenta, V. Pirronello, M. Platino, M. Plum, C. Porowski, R. R. Prado, P. Privitera, M. Prouza, E. J. Quel, S. Querchfeld, S. Quinn, R. Ramos-Pollan, J. Rautenberg, D. Ravignani, J. Ridky, F. Riehn, M. Risse, P. Ristori, V. Rizi, W. Rodrigues de Carvalho, G. Rodriguez Fernandez, J. Rodriguez Rojo, D. Rogozin, M. J. Roncoroni, M. Roth, E. Roulet, A. C. Rovero, P. Ruehl, S. J. Saffi, A. Saftoiu, F. Salamida, H. Salazar, A. Saleh, F. Salesa Greus, G. Salina, F. Sánchez, P. Sanchez-Lucas, E. M. Santos, E. Santos, F. Sarazin, R. Sarmento, C. Sarmiento-Cano, R. Sato, M. Schauer, V. Scherini, H. Schieler, M. Schimp, D. Schmidt, O. Scholten, P. Schovánek, F. G. Schröder, S. Schröder, A. Schulz, J. Schumacher, S. J. Sciutto, A. Segreto, A. Shadkam, R. C. Shellard, G. Sigl, G. Silli, O. Sima, A. Śmiałkowski, R. Šmída, B. Smith, G. R. Snow, P. Sommers, S. Sonntag, R. Squartini, D. Stanca, S. Stanič, J. Stasielak, P. Stassi, M. Stolpovskiy, F. Strafella, A. Streich, F. Suarez, M. Suarez Durán, T. Sudholz, T. Suomijärvi, A. D. Supanitsky, J. Šupík, J. Swain, Z. Szadkowski, A. Taboada, O. A. Taborda, V. M. Theodoro, C. Timmermans, C. J. Todero Peixoto, L. Tomankova, B. Tomé, G. Torralba Elipe, P. Travnicek, M. Trini, R. Ulrich, M. Unger, M. Urban, J. F. Valdés Galicia, I. Valiño, L. Valore, G. van Aar, P. van Bodegom, A. M. van den Berg, A. van Vliet, E. Varela, B. Vargas Cárdenas, G. Varner, R. A. Vázquez, D. Veberič, C. Ventura, I. D. Vergara Quispe, V. Verzi, J. Vicha, L. Villaseñor, S. Vorobiov, H. Wahlberg, O. Wainberg, D. Walz, A. A. Watson, M. Weber, A. Weindl, L. Wiencke, H. Wilczyński, C. Wileman, M. Wirtz, D. Wittkowski, B. Wundheiler, L. Yang, A. Yushkov, E. Zas, D. Zavrtanik, M. Zavrtanik, A. Zepeda, B. Zimmermann, M. Ziolkowski, Z. Zong, and F. Zuccarello. Inferences on mass composition and tests of hadronic interactions from 0.3 to 100 eev using the water-cherenkov detectors of the pierre auger observatory. Phys. Rev. D, 96:122003, Dec 2017.

[57] A Guillen, A Bueno, JM Carceller, JC Martinez-Velazquez, G Rubio, CJ Todero Peixoto, and P Sanchez-Lucas. Deep learning techniques applied to the physics of extensive air showers. Astroparticle Physics, 2019.

[58] Francisco Carrillo-Perez, Luis Javier Herrera, Juan Miguel Carceller, and Alberto Guillén. Improving classification of ultra-high energy cosmic rays using spacial locality by means of a convolutional dnn. In International Work-Conference on Artificial Neural Networks, pages 222–232. Springer, 2019.

[59] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[60] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine Learning in Python . Journal of Machine Learning Research, 12:2825–2830, 2011.

[61] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. ACM Transactions on Intelligent Systems and Technology, 2:27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[62] Wes McKinney. Data structures for statistical computing in python. In Stéfan van der Walt and Jarrod Millman, editors, Proceedings of the 9th Python in Science Conference, pages 51 – 56, 2010.

[63] StÃ©fan van der Walt, S. Chris Colbert, and GaÃ«l Varoquaux. The numpy array: A structure for efficient numerical computation. Computing in Science & Engineering, 13(2):22–30, 2011.

[64] J. D. Hunter. Matplotlib: A 2d graphics environment. Computing in Science & Engineering, 9(3):90–95, 2007.

[65] NVIDIA. GEFORCE GTX 1080. `https://www.nvidia.com/es-es/geforce/products/10series/geforce-gtx-1080/`, 2019. [Online; accessed 8 de Mayo 2019].

[66] Intel. Intel Xeon. `https://www.intel.es/content/www/es/es/products/processors/xeon.html`, 2019. [Online; accessed 8 de Mayo 2019].

[67] The Pierre Auger Cosmic Ray Observatory. Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment, 798:172 – 213, 2015.

[68] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In Geoffrey Gordon, David Dunson, and Miroslav Dudík, editors, Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, volume 15, pages 315–323, 2011.

[69] Alexander Aab et al. Depth of maximum of air-shower profiles at the Pierre Auger Observatory. I. Measurements at energies above $10^{17.8}$eV. Phys. Rev.D, 90(12):122005, 2014.

[70] A. Aab et al. Depth of maximum of air-shower profiles at the Pierre Auger Observatory. II. Composition implications. Phys. Rev. D, 90(12):122006, 2014.

[71] M. Aartsen et al. Search for correlations between the arrival directions of IceCube neutrino events and ultrahigh-energy cosmic rays detected by the Pierre Auger Observatory and the Telescope Array. Journal of Cosmology and Astroparticle Physics, 2016(1):037–037, 2016.

[72] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. CoRR, abs/1412.6980, 2014.

[73] Institute for Nuclear Physics (IKP). Corsika – an air shower simulation program. `https://www.ikp.kit.edu/corsika/index.php`, 2018.

[74] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. Neural computation, 1(4):541–551, 1989.

[75] 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(Oct):2825–2830, 2011.

[76] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems, pages 1097–1105, 2012.

[77] Nik Khadijah Nik Aznan, Stephen Bonner, Jason Connolly, Noura Al Moubayed, and Toby Breckon. On the classification of ssvep-based dry-eeg signals via convolutional neural networks. In 2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC), pages 3726–3731. IEEE, 2018.

[78] Stefano Argiro, SLC Barroso, J Gonzalez, Lukas Nellen, T Paul, TA Porter, L Prado Jr, M Roth, R Ulrich, and D Veberič. The offline software framework of the pierre auger observatory. Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment, 580(3):1485–1496, 2007.

[79] Sylvain Gugger and Jeremy Howard. AdamW and Super-convergence is now the fastest way to train neural nets. `https://www.fast.ai/2018/07/02/adam-weight-decay/`, 2018. [Online; accessed 19 June 2019].

[80] José Carlos Martínez Velázquez. Aplicación de técnicas de Aprendizaje Automático al análisis de Rayos Cósmicos de Ultra Alta Energía. Master's thesis, Universidad de Granda, Spain, 2018.