UNIVERSITÀ DEGLI STUDI DI PADOVA

Dipartimento di Fisica e Astronomia “Galileo Galilei”

Corso di Laurea in Fisica

Tesi di Laurea

Deep Neural Networks for Electron-Positron discrimination in the JUNO experiment

Relatore

Alberto Garfagnini

Correlatori

Yury Malyshkin

Francesco Manzali

Laureando

Giulio Vicentini

Anno Accademico 2018/2019

Abstract

Neutrino physics has always been an important area of research in particle physics, especially since the discovery of neutrino oscillations. The Jiangmen Underground Neutrino Observatory (JUNO) is a new large liquid scintillator detector that aims to solve the neutrino mass hierarchy measuring reactor neutrino interaction in the detector via inverse beta decay. One of the most relevant background effects is given by the presence of electrons which, even if they don't take part in the decay reaction, leave a trace very similar to that of positrons in the liquid scintillator. High energies experimental physics has always had to deal with the management and observation of large amounts of data; for this reason, nowadays, the development of algorithms and the use of computer techniques are some of the most important skills that form the background preparation of a physicist. With the advent of the so-called *big data* and the rapid development of hardware components, the field of artificial intelligence has made numerous progresses in recent years. In this thesis the electrons-positrons discrimination in JUNO experiment are investigated through the use of artificial neural networks.

The work is organized by first introducing the main features of JUNO and the inverse beta decay; then a brief illustration of the modern deep learning techniques is given. Finally, the data set is presented, together with the development of the techniques, the data analysis and the obtained results.

Contents

Chapter 1

Introduction

1.1 The JUNO Experiment

The Jiangmen Underground Neutrino Observatory (JUNO) [1][2] is a large liquid scintillator detector currently under construction near Kaiping, Jiangmen, South China. Its main goal is to determine the *neutrino mass hierarchy* (MH).
In fact, the mass of the neutrino has never been directly measured due to its very small dimension; however, the differences between the squares of the masses of the various flavors were managed to be measured [3]. As it is known, neutrinos occur in 3 different eigenstates: electronic $|\nu_e\rangle$, muonic $|\nu_\mu\rangle$ and tauonic $|\nu_\tau\rangle$. However, other experiments such as Super-Kamiokande [4] have shown that neutrinos oscillate between mass eigenstates $|\nu_1\rangle$, $|\nu_2\rangle$ and $|\nu_3\rangle$ (different from those of flavor). As a matter of fact, the time evolution operator acts differently on the three masses and this causes the phenomenon of *neutrino oscillation*, where a neutrino of a certain flavor ν_i can be detected as another flavor ν_f after a certain distance traveled. More precisely, the relationship between the mass eigenstates $|\nu_i\rangle$ and the flavor eigenstates $|\nu_\alpha\rangle$ is given by the Maki-Nakagawa-Sakata-Pontecorvo (MNSP) matrix [5][6][7]:

$$\begin{pmatrix} \nu_e \\ \nu_\mu \\ \nu_\tau \end{pmatrix} = \begin{pmatrix} U_{e1} & U_{e2} & U_{e3} \\ U_{\mu 1} & U_{\mu 2} & U_{\mu 3} \\ U_{\tau 1} & U_{\tau 2} & U_{\tau 3} \end{pmatrix} \begin{pmatrix} \nu_1 \\ \nu_2 \\ \nu_3 \end{pmatrix} \tag{1.1}$$

which can be parameterized using 3 mixing angles θ_{12}, θ_{13}, θ_{23} and a parameter δ called *Dirac CP-violating phase*.
The order of the masses of the first two eigenstates was determined $m_2 > m_1$ [3].

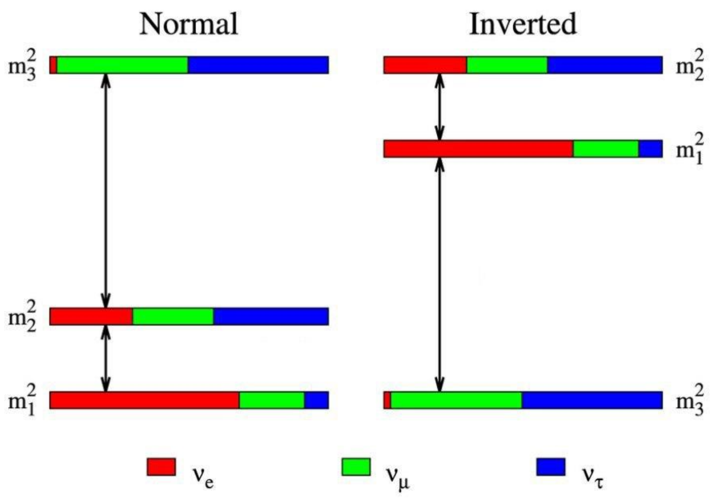


Figure 1.1: Neutrino mass hierarchies. Leftside normal, rightside iverted hierarchy. The colors denote the individual flavor contribution to each mass eigenstate [1]

However, the position of the third mass eigenstate remains an open question, and there are two possible scenarios:

- **Normal Mass Hierarchy (NH)** $m_3 > m_2 > m_1$.
- **Inverted Mass Hierarchy (IH)** $m_2 > m_1 > m_3$.

In order to study MH, JUNO is located near two nuclear power plants, equidistant 53 Km from each of the two. The reactors of the plants act as a source of electronic antineutrinos ($\bar{\nu}_e$). To study these products, their probability of survival is taken into account

$$P(\bar{\nu}_e \to \bar{\nu}_e) = 1 - sin^2(2\theta_{12})c_{13}^4 sin^2\left(\frac{\Delta m_{12}^2 L}{4\mathcal{E}}\right) - sin^2(2\theta_{13})\left[c_{12}^2 sin^2\left(\frac{\Delta m_{13}^2 L}{4\mathcal{E}}\right) + s_{12}^2 sin^2\left(\frac{\Delta m_{32}^2 L}{4\mathcal{E}}\right)\right] \tag{1.2}$$

where $s_{ij} = sin\theta_{ij}$, $c_{ij} = cos\theta_{ij}$, $\mathcal{E}$ is the neutrino energy, L the distance traveled and $\Delta m_{ij}^2 = m_i^2 - m_j^2$.

To give a solution to the MH problem, JUNO has been designed to have an energy resolution of at least $3\%/\sqrt{\text{Mev}}$. To achieve such a resolution, as shown in figure 1.2 the detector is composed of a scintillating liquid contained within an acrylic sphere with a diameter of 35.4 m. To collect the light produced by the excited scintillator, the container is covered with a structure that supports about 18000 large photomultipliers (∅20 inch) alternating with 25600 small PMTs (∅3 inch) [8]. The central core is immersed in a tank full of instrumented water which acts as a water Cherenkov, which allows to reduce background events. Finally, further layers of plastic scintillators, the top tracker, were placed at the upper end to detect other background particles, mainly atmospheric muons.

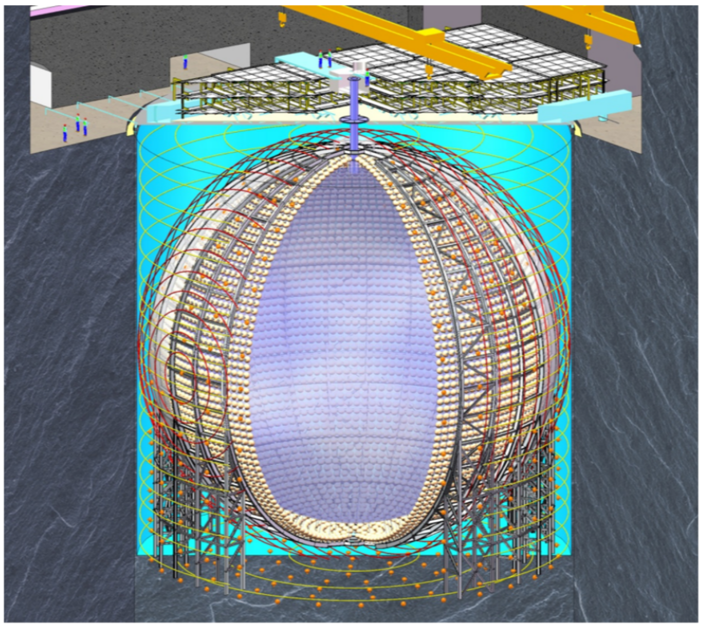


Figure 1.2: Structure of the JUNO experiment [1][2]

1.2 Inverse Beta Decay

The golden channel for reactor anti-neutrino detection in JUNO is given by the Inverse Beta Decay (IBD)

$$\bar{\nu}_e + p \longrightarrow n + e^+ \tag{1.3}$$

This type of event can be schematized in figure 1.3. The electronic antineutrino interacts with a proton in the scintillator through weak interaction producing a positron and a neutron.

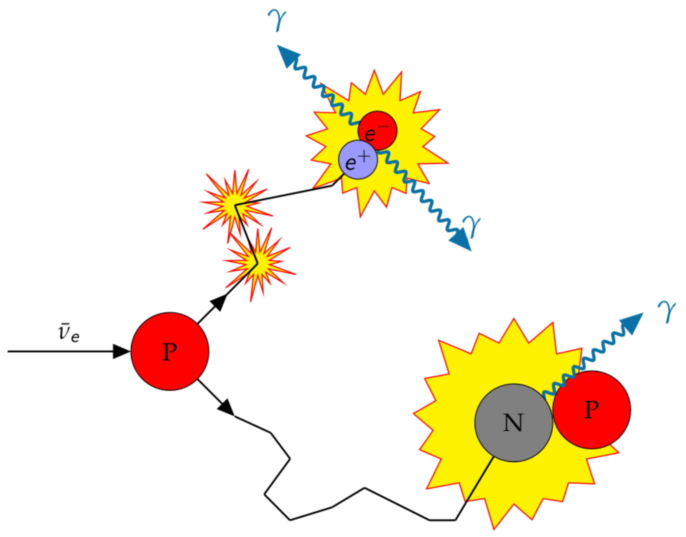


Figure 1.3: Reactor neutrino signature in liquid scintillator [9]

The positron reacts with the scintillator medium by depositing energy and the latter releases a cascade of scintillation light. The light produced propagates through the liquid until it reaches the PMTs. When the positron has lost most of its kinetic energy it can annihilate with one of the electrons contained in the medium or produce unstable positronium. This annihilation leads to the creation of two γ rays with an energy of 511 keV each which, in turn, interact with the medium, producing other charged particles and giving rise to further scintillation light. Taking into account this electron-positron annihilation it is possible to reconstruct the initial energy of the neutrino [10]

$$\begin{aligned} E_{\bar{\nu}_e} &\approx E^k_{e^+} + m_e + m_n - m_p \\ &\approx E_{vis} - m_e + m_n + m_p, \qquad \text{with } E_{vis} = E^k_{e^+} + 2m_e \\ &\approx E_{vis} + 0.8\text{MeV} \end{aligned} \tag{1.4}$$

On the other hand, the neutron, moderates in the liquid scintillator until it is absorbed by a nucleus, emitting 2.2 MeV gamma rays.

1.3 Electron-Positron discrimination

Even if the inverse beta decay leaves a distinct mark in the detector, there are some events that produce a very similar trace and therefore must be interpreted as background signals for the measurement of MH. Most background events actually consist of an electron instead of a positron.
Indeed, the electrons generate a signal very similar to the positrons in the scintillator liquid, as figure 1.4 shows. The only difference is the lack of annihilation, as there are no free positrons in the medium.

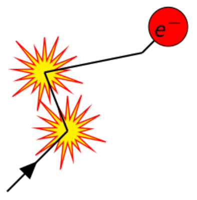


Figure 1.4: Electron signal in LS [9]

After the positron has lost most of its kinetic energy, it annihilates directly with a free electron in the SL or forms metastable positronium. Positronium can appear in two different states, para-positronium and ortho-positronium [11].
Therefore, the differences in the electron and positron traces can be summarized as follows:

- For direct annihilation and para-positronium decay the main difference is the spatial distribution of scintillation light due to the propagation of the two generated γ's.
- For ortho-positronium there is a difference in spatial distribution.
- The relations between initial particle energy and visual energy are:

$$\begin{aligned} E_{vis}^{e^-} &\approx E_k \\ E_{vis}^{e^+} &\approx E_k + 2E_\gamma \end{aligned} \tag{1.5}$$

This leads to three possible principles of discrimination, based on:

- Pulse shape differences due to the different time arrival of optical photons to the PMT photocathodes.
- Spatial differences via track reconstruction (3D).
- Spatial differences via light pattern on detector surfaces (2D).

In this thesis the first principle of discrimination (*time profile* pattern) is investigated using Convolutional Neural Networks (CNNs) [12].

Chapter 2

Deep Learning

2.1 Machine Learning

Deep Learning ia a specific kind of machine learning. Therefore, to understand deep learning, a good understanding of the basic principles of machine learning is required.
In general, a machine learning algorithm is an algorithm that is able to learn from data or, more precisely, an algorithm that is capable of improving a computer program's performance at some task via experience [12]. On a more concrete level it is useful to give an example of a simple machine learning algorithm: linear regression.

Linear Regression

As the name implies, linear regression solves a regression problem. In other words, the goal is to build a system that can take a vector $\boldsymbol{x} \in \mathbb{R}^n$ as input and predict the value of a scalar $y \in \mathbb{R}$ as its output. Hence, the output of linear regression is a linear function of the input.

$$y = \boldsymbol{w}^\top \boldsymbol{x} \tag{2.1}$$

where $\boldsymbol{w} \in \mathbb{R}^n$ is a vector of parameters.
Parameters are values that control the behavior of the system. In this case, w_i is the coefficient that multiplies the feature x_i before summing up the contributions from all the features. In other words, $\boldsymbol{w}$ is a set of weights that determines how each feature affects the predictions. Thus, in this case, the definition of the task is clear: to predict y from $\boldsymbol{x}$ by calibrating $\boldsymbol{w}$.
In reality the term linear regression is often used to refer to a slightly more sophisticated model with one additional parameter – an intercept term b. In this model

$$y = \boldsymbol{w}^\top \boldsymbol{x} + b \tag{2.2}$$

so the mapping from feature to predictions is now an affine function, where the b parameter is often called the bias.

Linear regression is of course an extremely simple and limited learning algorithm, but it provides a clear example of how a learning algorithm can work.
However, the central challenge in machine learning is that our algorithm must perform well on new, previously unseen inputs – not just those on which the model was trained. The ability to perform well on a previously unobserved inputs is called generalization. The first – most important – causes for the poor performance of machine learning algorithms are **underfitting** and **overfitting**.

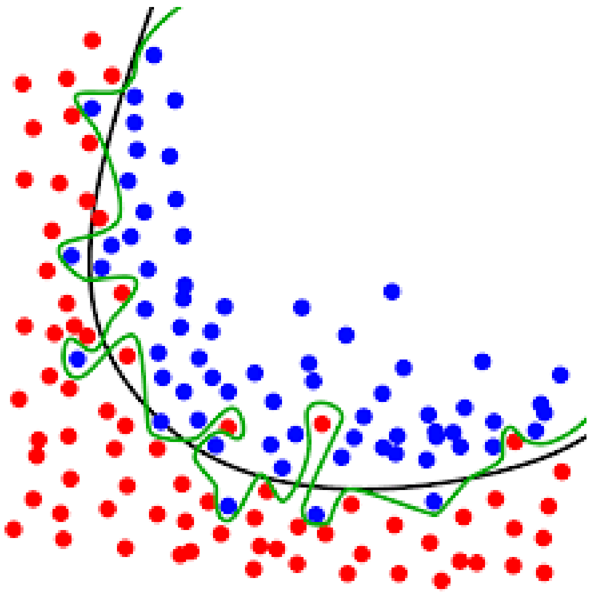


Figure 2.1: Example of an overfitted model.

- Underfitting occurs when a model is too simple – informed by too few features or regularized too much – which makes it inflexible in learning from dataset.
- Overfitting is the opposite problem, makes the model focusing too much on the input set and learns complex relations which may not be valid in general from new data.

Both cases are undesirable, and can be corrected by adding more instances or modifying the model. Therefore the process requires many arbitrary choices. This is made by setting the so-called *hyper-parameters*, through a *trial-and-error* process called *fine-tuning*.

2.2 Binary classificaton

Discrimination between electrons and positrons is a perfect example of the so-called *binary classification task* [12]. Given two distinct categories, to solve this type of task, the algorithm must be able to specify which category the input data belongs to. To do this, the model is generally asked to construct a function $y = f(\boldsymbol{x})$ which assigns an a identified numeric code y to each category, described by a vector $\boldsymbol{x}$.
To distinguish between two different classes of data, it is very useful to have a set of input data already labeled, so that the algorithm can learn from features that are already classified. In this case we speak of **supervised learning**. The most important parameters to define, in order to complete this task, are the following:

1. **Data specification.** A set of labeled data $D_{train} = \{\boldsymbol{x}^{(i)}, y^{(i)}\}_{i=1,...,m}$. Each pair i is called an *instance*, and contains all the relevant information about the i-th event. The vector $\boldsymbol{x}^{(i)}$ represents the *features* of that event, that is a set of chosen functions of the available experimental data. On the other hand, $y^{(i)}$ represents the category it belongs to.
2. **Model specification.** A *model* $f_{\boldsymbol{W}}$, that is a function which goes from the vectors space of features to that of labels. f is completely specified by a set $\boldsymbol{W}$ of parameters.
3. **Training**. The parameters are initially randomly chosen from a distribution. A cost function, called *loss*, measures how far the predictions fall from the known values y. Then the parameters are tweaked by an *optimizer* in order to lower the loss value. The process is iterated many times, until a good candidate for $f_{\boldsymbol{W}}$ has been found.
4. **Validation.** If the process of supervised learning succeeds, $f_{\boldsymbol{W}}$ should be able to generalize, that is predict accurate labels for features that were not part of D_{train}. This can be checked by evaluating the model's performance on another labeled set D_{val} (validation set).

5. **Testing.** The general performance of the model is evaluated one last time on a labeled set D_{test}, different from all the previous.

2.3 Deep Neural Networks

Deep learning provides powerful tools for dealing with supervised learning.
The Deep Neural Networks (DNNs) are a large class of models used for pattern recognition which, in recent years, have been applied with great success in various areas, especially thanks to the recent development of high-performing hardwares. Indeed, artificial neural networks were conceived in the early 60s [13], taking inspiration from the structures of the human brain.
The basic unit of a neural network is the neuron. In artificial neural networks, a neuron can be schematized as in figure 2.1, where different signals transmit information through numerical values. To distinguish the importance of the different inputs, each of them is suitably weighted through a *weight* parameter. Before transmitting the output through an activation function, an offset or *bias* is added to the signal.

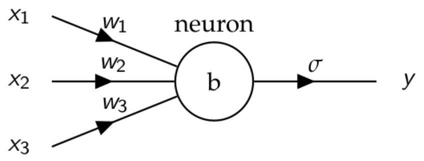


Figure 2.2: Artificial neuron.

Therefore, the model is interpreted through a function of this type:

$$y = \sigma\left(\sum_i w_i x_i + b\right) \tag{2.3}$$

where x_i is the i-th input signal, w_i the weight corresponding to x_i, b is the bias, σ is the activation function and y is the neuron output.
In this thesis, the so-called *rectified linear unit* (ReLU) was used as activation function:

$$\text{ReLU}(x) = \max(0, x) \tag{2.4}$$

In general, the connection between biological neurons can be very complex. For simplicity, artificial neural networks are made up of layers of neurons, so that neurons in a certain layer receive inputs only from neurons in the previous layer. The simplest version of this concept is the dense layer, sometimes called *fully connected*, where each neuron receives signals from each neuron of the previous state.

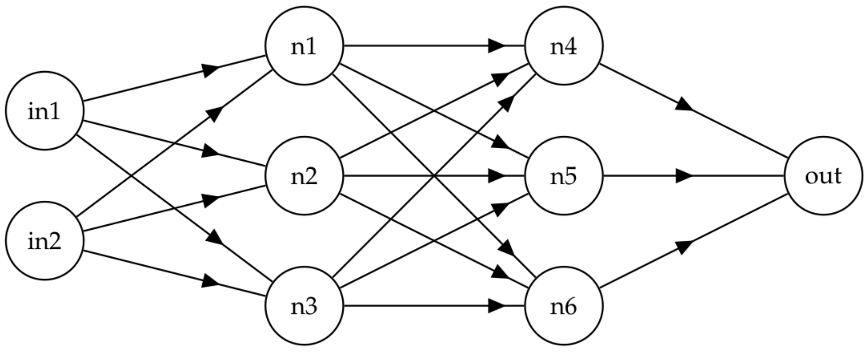


Figure 2.3: Simple Neural Network with two dense layers.

In the case of a deep neural network, consisting of many layers and many nodes, the number of parameters can become enormous, and the determination of them must be automated. In analogy with biological neural networks this process is called "training". During this learning step it is useful to monitor how the model learns. To do this, the *loss function* must be minimized during the process. In this case, since it is a binary classification task, the *binary cross entrophy* [14] has been chosen:

$$\text{BCE} = -\frac{1}{N_{\text{samples}}} \sum_{\text{samples}} \sum_{i}^{\#\text{nodes}} s_i^{\text{true}} \ln(s_i^{\text{pred}}) + (1 - s_i^{\text{true}}) \ln(1 - s_i^{\text{pred}}) \tag{2.5}$$

where s_i is the i-th neuron output.
Another more intuitive indicator of the performance of the classification algorithm is *accuracy*:

$$\text{accuracy} = \frac{\#\text{correct predictions}}{N_{\text{samples}}} \tag{2.6}$$

A typical task for which deep learning is used is that of image classification. In this case the model will have to be built to address several issues:

- High input dimension.
- Interesting features could be found only in some areas of the image.
- A type of a class can be composed of different characteristics, which can also be contained in other classes. DNNs must be able to decompose each specific characteristic.

The models that provide solutions for all these needs are the **Convolutional Neural Networks (CNNs)**, formed by convolutional layers. The principle of operation of a convolutional layer consists in applying a filter called *kernel*, smaller than the size of the image, to all the pixels that make up the image. A filter consists of a small number of values with which each pixel is weighed. Each of these values is added together before going through an activation function. Then, the filter returns a map of values called *feature map*.

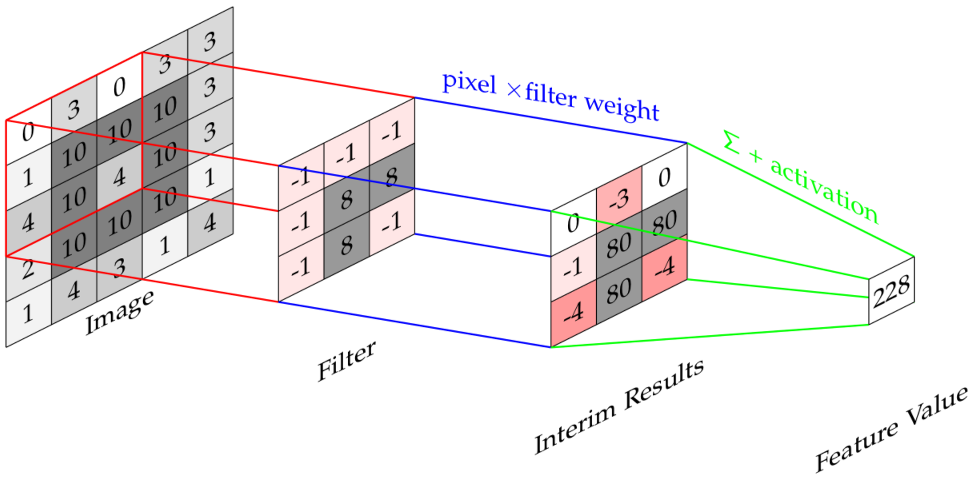


Figure 2.4: Example of convulational filter.

In particular in this work 1D convolutional filters have been applied, which take one-dimensional arrays as input.

Chapter 3

Analysis

3.1 Monte Carlo data

One of the fundamental requirements for the use of deep learning techniques concerns the acquisition of a large amount of data to train the model. The JUNO collaboration provides a Monte Carlo simulation software[1] able to emulate the detector and provide "high level" data, that consist in physical events reconstructed observable, such as position and energy. All data are provided in the form of TTree objects of the ROOT software [16] that contain different types of information, such as the "true information" of the events or the features detected by the PMTs of different sizes. Here we use:

- one million events for e^+ and one million for e^- with uniformly distributed energies $E_k \in [0, 10]$ MeV that form the training set D_{train} and the validation set D_{val}.
- 10 sets each with $2 \cdot 10^3$ instances and discrete energy $E_k \in \{0, 1, \ldots, 9\}$ MeV, used as test set D_{test}.

In order to simplify the analysis, the following restrictions are made:

- Only large PMTs (∅20 inch) are considered.
- The *Dark Noise* is totally removed.

As mentioned before, in this thesis a discrimination approach based on **time profile** has been chosen, for this reason the subsequent preparation of the data is based on the selection of particular features. In addition to the filters already applied, only the information contained in the first hit time for each PMT was taken into consideration. The times were then normalized so that the scale started from the arrival time of the first hit PMT. Before proceeding, as late hits tend to undergo multiple scatterings, a time cut of < 300 ns was applied.

3.2 Data preparation

Electron-positron discrimination is a difficult problem [9]. The physical differences of these events are modest and only a week discrimination has been achieved with traditional analysis methods. For this reason an approach with neural networks has been adopted. Focusing on the time profile, different types of input forms are investigated, trying to single out the best method.

[1]SNiPER framework [15]

3.2.1 Mollweide Projection

Since convolutional neural networks generally require input images, the first approach was to transform data into images. However CNNs are generally used for Euclidean geometries, the spherical arrangement of the PMTs in the detector surface must be projected into a rectangle so that it can be used as network input.
The projection was made thanks to the methods provided by the Healpy library [17]. It is a library available in Python 3.6 that allows to apply the Hierarchical Equal Area and iso-Latitude Pixelation of the sphere (HEALPix), developed by NASA for astrophysical research [18]. The HEALPix scheme starts by dividing the spherical surface into 12 equivalent quadrilaterals of different shapes, organized into three rings - two around the poles and one around the equator (fig. 3.1). Each pixel is then further divided into four equivalent areas, and the process repeats until the desired resolution is reached. The number of divisions made for each of the 12 starting pixels are decided through the parameter N_{side} which, to preserve the hierarchical structure, must be a power of 2. Therefore, the total number of partitions will be $N_{pix} = 12N_{side}^2$

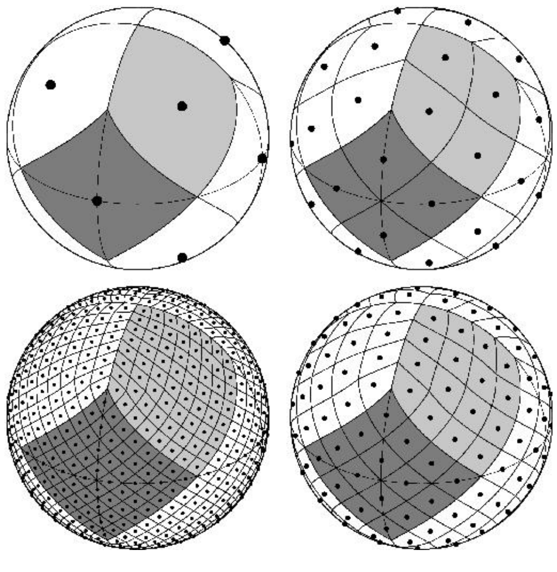


Figure 3.1: Scheme of the HEALPix pixelation [18]

With the aim of analyzing the time profile, the image is constructed starting from the **first hit time**: each spherical pixel is associated with the minimum hit time of the corresponding PMT, where 0 ns corresponds to the first PMT fired. In an attempt to represent as much information as possible, a $N_{side} = 32$ has been chosen, corresponding to a number of pixels $N_{pix} = 12218$, so that it have about 1.5 PMTs per pixel.
The Healpy library provides different solutions to represent and project spherical data. For this model, in particular, the **Mollweide projection** [19] was chosen. It is a map, famous for being mainly used in geographical maps, whose main characteristic is that of leaving the areas unchanged. Through the following transformations to the Cartesian coordinates:

$$x = R\frac{2\sqrt{2}}{\pi}(\lambda - \lambda_0)cos\theta \qquad y = R\sqrt{2}sin\theta \tag{3.1}$$

where θ is an auxiliary angle defined by $2\theta + sin2\theta = \pi sin\phi$, λ the longitude, λ_0 the central meridian, ϕ the latitude and R the radius of the sphere.
It is a equal-area maps that preserve area measure, generally distorting shapes in order to do that. It allows to give priority to the information contained in a certain surface, rather than to the shape of this, so the signal per unit area is shown in correct proportion.

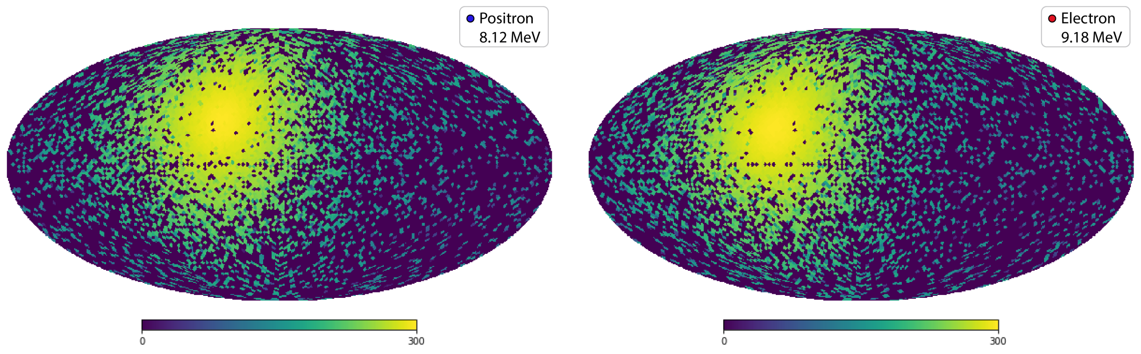


Figure 3.2: Mollweide Projections of two events with similar energies: positron on the left and electron on the right.

Figure 3.2 shows a positron and electron sample, respectively, with approximately the same energy ($E^{e^+}_{\text{true}} = 8.12$ MeV and $E^{e^-}_{\text{true}} = 9.18$ MeV)[2]. Nowadays, the most performing neural network architectures, while achieving excellent results in various fields of application, are not much better than humans in completing a classification task. Therefore, starting from the assumption that we are good classifiers, it is impossible at first sight to recognize patterns in the images that allow us to significantly distinguish the two particles. For this reason the path of building images has been discarded.

3.2.2 Power Spectrum

The representation of the features through the images could have led to a loss of information. For this reason, a method was sought to encode the same information obtained from the HEALPix subdivision into a different form. The idea, borrowed from cosmology, is to go through the **power spectrum**.
Spherical harmonics form a complete orthonormal base of the two-dimensional sphere $L^2(S^2, d\Omega)$ and are defined by

$$Y_{lm} = \sqrt{\frac{2l+1}{4\pi}\frac{(l-m)!}{(l+m)!}}e^{im\phi}P_l^m(cos\theta) \tag{3.2}$$

where $l = 0, \dots, \infty$, $-l \leq m \leq l$ and P_l^m are the Legendre polynomials.
Therefore, it is possible to write any function on the sphere as a linear combination of spherical harmonics expanding the represented signal by the following functions:

$$\psi(\theta,\phi) = \sum_{l=0}^{l=\infty}\sum_{m=-l}^{l} a_{lm}Y_{lm}(\theta,\phi) \qquad \text{with} \quad a_{lm} = \int_{\theta=-\pi}^{\pi}\int_{\phi=0}^{2\pi}\psi(\theta,\phi)Y^*_{lm}(\theta,\phi)d\Omega \tag{3.3}$$

Similarly to what is done in the Fourier space, we define the power spectrum C_l of these fluctuations starting from the harmonic coefficients

$$C_l = \frac{1}{2l+1}\sum_{m=-l}^{l}|a_{lm}|^2 \tag{3.4}$$

The `anafast` function of the Healpy library returns just these coefficients.
Then, this same expansion was adopted to analyze the temporal profile mapped by Mollweide's projection. Taking advantage of a typical convention of the study of the Comsmic Microwave Background (CMB) radiation field [20], two graphs are proposed in which $l(l+1)C_l$ is plotted as a function of l.

[2]As showed in section 1.3, for positrons only, there is a difference between the visible and the kinetic energy: $E^{e^+}_{vis} - E^{e^+}_{k} \approx 2E_\gamma \sim 1$ MeV

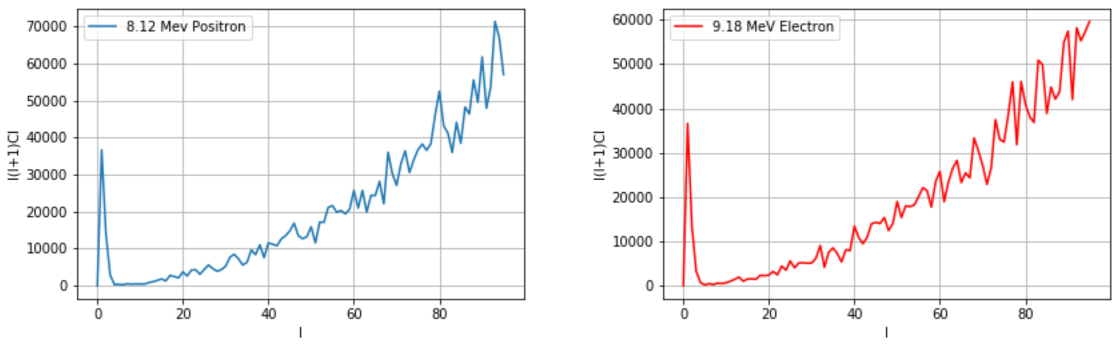


Figure 3.3: Power Spectrum of two events with similar energies: positron (left) and electron (right).

Figure 3.3 compares the power spectrum of the projections of an electron and a positron with almost the same energy.
In order to give the information about the spatial distribution of the signal, a first 1D model was trained with arrays containing the C_l coefficients; but the network seems to have learned nothing from this type of input.

3.2.3 Time Distribution

The projection from spherical geometry to Cartesian coordinates, together with the discretizzation in pixels, may not allow to maximally encode the information contained in the features of the event. For this reason a simpler solution has been tried.
With the intention of representing time profiles of the events, histograms that collect the information of the hit times have been produced. In order to find the best way to represent this type of information, the response of the model trained with various forms of histograms was studied. In addition to changing the number of bins, different bins distributions have also been tried (such as the logarithmic one). To achieve the task and to make the distribution along the bins more uniform, the histograms with 100 equal size bins turned out to be the best choice. By deepening the use of time histograms this strategy has been expanded by investigating the response of the model to datasets divided according to different physical characteristics of the events. A division based on the energy of the event was tried.
For a first comparison, the figure shows the 100 bins histograms of the usual sample events.

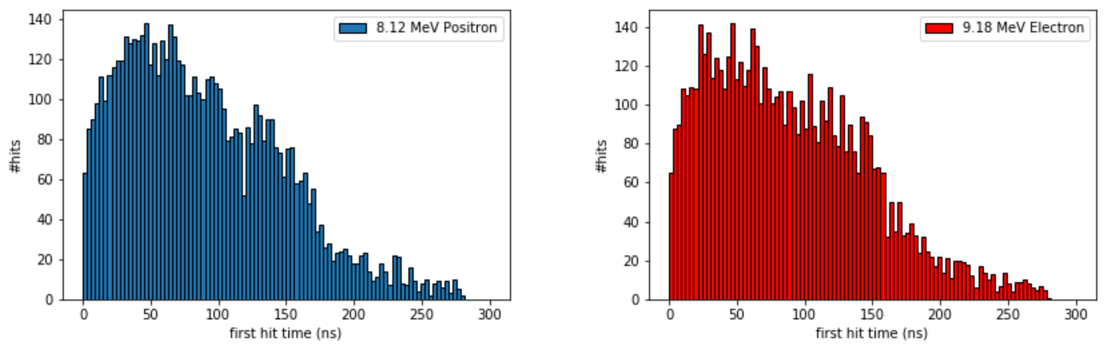


Figure 3.4: Time Profile of two events with similar energies: positron (left) and electron (right).

3.3 Model

The different types of prepared data were used to train the same type of model. There are several libraries that implement algorithms for neural networks. A popular open source library is *Keras* API (Application Programming Interface) [21], written for and in Python. This library was chosen because it has a user friendly interface with many standard methods for deep learning.

3.3.1 VGG architecture

A good starting point is the general architectural principles of the VGG[3] model [22]. This is a good starting point because they achieved top performance in the ILSVRC 2014 competition [23] and because the modular structure of the architecture is easy to understand and implement.
The architecture involves stacking convolutional layers followed by a max pooling layer. Together, these layers form a block, and these blocks can be repeated where the number of filters in each block is increased with the depth of the network as figure 3.5 shows. Each layer uses the ReLU activation function.

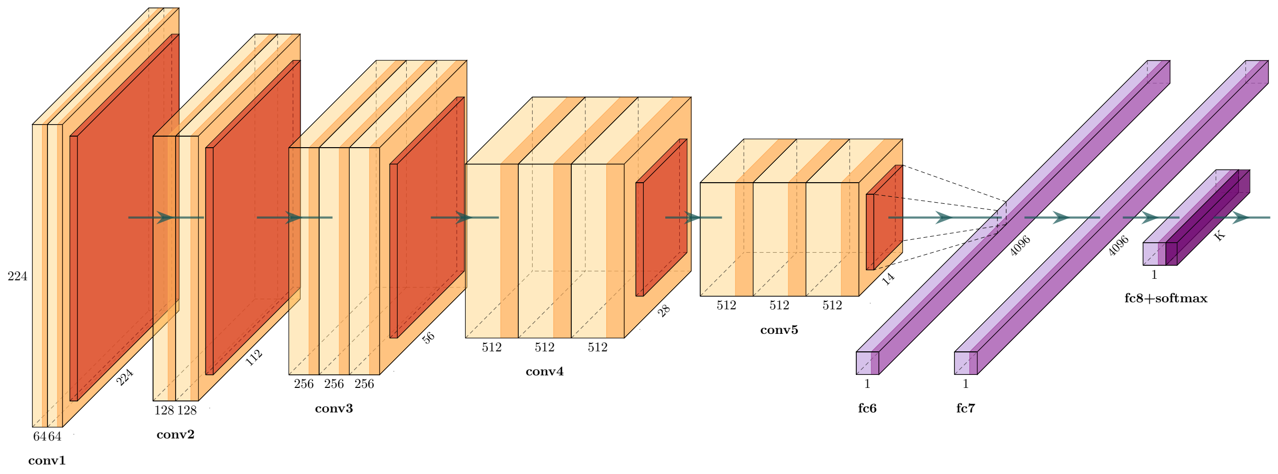


Figure 3.5: Scheme of the VGG16 architecture. Convolutional layers are in yellow, maxpool in red and fully connected in purple.

This architecture is usually implemented to receive images as input, in this case it has been adapted accordingly to be trained through one-dimensional arrays. Therefore, the network used consists of 3 VGG blocks, each consisting of a 1D convolutional layer which presents an increasing number of filters, specifically $32, 64, 128$.
Furthermore, since it is a binary classification problem, it is important to build the models so that the result of the prediction returns 0 or 1 corresponding to the type of class. In this sense, at the end of the VGG blocks, two dense layers – with a single output node – were implemented, activated by the sigmoid function. Consistently, the model has been optimized through the binary cross-entropy loss function.
In general, convolutional neural networks are implemented to recognize spatially related patterns by applying filters only to small portions of the input. For this reason they achieve the best results in image recognition. In this case small filters has been tried but the type of input does not require to be investigated to recognize local patterns, instead it must be totally interpreted. As a matter of fact, the time histograms represent a sort of "integral" features. To do this, the model was therefore constructed so that each convolutional layer applies a filter exactly as large as the input size, in order to interpret the histogram as a whole. Precisely, a *kernel size* = 100 has been set.

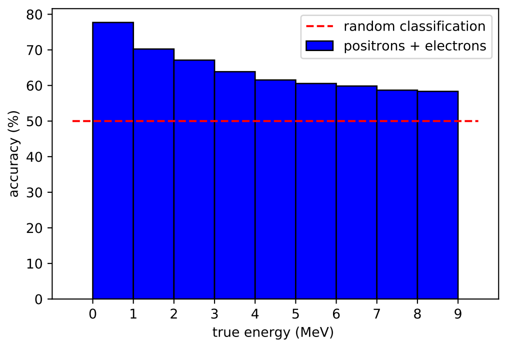
3.3.2 GPU usage

Keras runs with *Tensorflow* backend API [24] that is used to optimize the calculations of large numbers of tensor operations, which are performed during the training of a neural network. To speed up the process it is useful to use GPUs, a hardware that allows you to perform multiple parallel calculations simultaneously.
In order to do this, the work was done on a virtual machine hosted on Cloud Veneto [25], using a Nvidia Tesla T4 GPU with a 15 cores CPU, 90 GB of RAM and 500 GB of available storage on a SSD.

[3]Visual Geometry Group of Oxford University

3.4 Results

In this section the results obtained with the 100 equally spaced bins histograms are showed.
As differences between electrons and positrons should become smaller for higher energies, as the annihilation signal becomes relatively small, an energy depended analysis is required as well. The two million events have been divided into nine energy ranges. Following the uniform distributions each subset contains about 200k events where electrons and positrons are compared with the usual $\Delta E^{e^+} \sim 1$Mev. The data are then divided into 80% training set and 20% validation set. In figure 3.6 accuracies are plotted against the true visible energies.

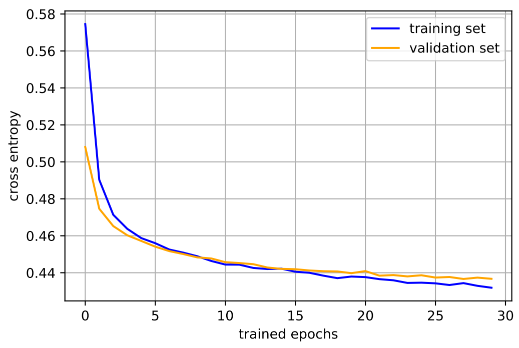


energy range (MeV)	accuracy (%)
$[0,1]$	77.70
$[1,2]$	70.24
$[2,3]$	67.12
$[3,4]$	63.87
$[4,5]$	61.54
$[5,6]$	60.55
$[6,7]$	59.84
$[7,8]$	58.86
$[8,9]$	58.34

Figure 3.6: Accuracy as function of energy. Energy ranges are those of protons.

As expected, with increasing energy, the discrimination task becomes more difficult and results provided by the model tend to get worse. However, it is fair to point out that a network that classifies randomly would produce an accuracy of 50%, for this reason even the 58% achieved for the highest energies on a validation set of about 40k events is still significantly better than a random classification.

Figure 3.7 shows the cross entropy and the accuracy against the epochs for the first energy range.



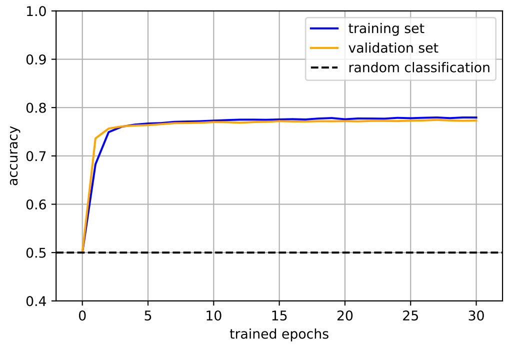


Figure 3.7: Loss function and accuracy against epochs for lowest energy. Plotted for training set in blue and validation set in orange.

This is the best result obtained, where the model managed to reach 78% accuracy on the validation set. The loss function for training set could be minimized even further but this would mean an overtraining of the network, as the validation loss function is already in its minimum. Stopping after 30 epochs seems reasonably.

Chapter 4

Conclusions

The goal of this thesis was to examine machine learning approaches for electron-positron discrimination in the JUNO experiment. Three different methods have been investigated, which are:

- the production of images with the HEALPix pixellation scheme,
- the distinction of the signal spatial distribution through the power spectrum,
- the difference between time profile patterns via time histogram building.

As the physics required, an analysis for different energy ranges was also discussed.
In the end the simplest approach of building histogram for data preparation has been adopted. Even if the final result remains inconclusive, the obtained results managed to reach a significant value anyway. As a matter of fact electron-positron separation is much more challenging than e.g. alpha/beta separation, which is successively achieved with traditional pulse-shape discrimination methods.
The work showed that features obtained from hit time information alone are not sufficient to allow the network to complete the classification at high accuracy level. For this reason, adding new features could be an interesting field of study. Future works could implement more information such as the reconstructed energy, the vertex position and/or spatial distribution of hits.

Bibliography

[1] Fengpeng An et al. “Neutrino Physics with JUNO”. In: *J. Phys.* G43.3 (2016), p. 030401. DOI: 10.1088/0954-3899/43/3/030401. arXiv: 1507.05613 [physics.ins-det] (cit. on pp. 7, 8).

[2] Zelimir Djurcic et al. “JUNO Conceptual Design Report”. In: (2015). arXiv: 1508.07166 [physics.ins-det] (cit. on pp. 7, 8).

[3] P. F. de Salas et al. “Status of neutrino oscillations 2018: 3σ hint for normal mass ordering and improved CP sensitivity”. In: *Phys. Lett.* B782 (2018), pp. 633–640. DOI: 10.1016/j.physletb.2018.06.019. arXiv: 1708.01186 [hep-ph] (cit. on p. 7).

[4] Y. Fukuda et al. “Evidence for oscillation of atmospheric neutrinos”. In: *Phys. Rev. Lett.* 81 (1998), pp. 1562–1567. DOI: 10.1103/PhysRevLett.81.1562. arXiv: hep-ex/9807003 [hep-ex] (cit. on p. 7).

[5] B. Pontecorvo. “Neutrino Experiments and the Problem of Conservation of Leptonic Charge”. In: *Sov. Phys. JETP* 26 (1968). [Zh. Eksp. Teor. Fiz.53,1717(1967)], pp. 984–988 (cit. on p. 7).

[6] Ziro Maki, Masami Nakagawa, and Shoichi Sakata. “Remarks on the unified model of elementary particles”. In: *Prog. Theor. Phys.* 28 (1962). [,34(1962)], pp. 870–880. DOI: 10.1143/PTP.28.870 (cit. on p. 7).

[7] S. M. Bilenky. “Bruno Pontecorvo and the neutrino”. In: *Usp. Fiz. Nauk* 184.5 (2014), pp. 531–538. DOI: 10.3367/UFNe.0184.201405g.0531 (cit. on p. 7).

[8] Miao He. “Double Calorimetry System in JUNO”. In: *Proceedings of International Conference on Technology and Instrumentation in Particle Physics 2017 (TIPP2017), 2 vol.: Beijing, China, May 22-26, 2017.* 2017. arXiv: 1706.08761 [physics.ins-det] (cit. on p. 8).

[9] T. Birkenfeld. “Electron-Positron Discrimination in the Jiangmen Underground Neutrino Observatory with Deep Neural Networks”. MA thesis. Fakultät für Mathematik, Informatik und Naturwissenschaften der RWTH Aachen, 2018 (cit. on pp. 9, 15).

[10] P. Vogel. “Analysis of the Anti-neutrino Capture on Protons”. In: *Phys. Rev.* D29 (1984), p. 1918. DOI: 10.1103/PhysRevD.29.1918 (cit. on p. 9).

[11] Mario Schwarz et al. “Measurements of the lifetime of orthopositronium in the LAB-based liquid scintillator of JUNO”. In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 922 (Dec. 2018). DOI: 10.1016/j.nima.2018.12.068 (cit. on p. 9).

[12] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning.* http://www.deeplearningbook.org. MIT Press, 2016 (cit. on pp. 10–12).

[13] Juergen Schmidhuber. “Deep Learning in Neural Networks: An Overview”. In: *Neural Networks* 61 (Apr. 2014). DOI: 10.1016/j.neunet.2014.09.003 (cit. on p. 13).

[14] Dirk P. Kroese Reuven Y. Rubinstein. *The Cross-Entropy Method: A Unified Approach to Combinatorial Optimization, Monte-Carlo Simulation and Machine Learning.* Springer-Verlag New York, 2004. DOI: 10.1007/978-1-4757-4321-0 (cit. on p. 14).

[15] Tao Lin et al. “The Application of SNiPER to the JUNO Simulation”. In: *Journal of Physics: Conference Series* 898 (Feb. 2017). DOI: 10.1088/1742-6596/898/4/042029 (cit. on p. 15).

[16] I. Antcheva et al. “ROOT - A C++ Framework for Petabyte Data Storage, Statistical Analysis and Visualization”. In: *Computer Physics Communications* 180 (June 2011), pp. 2499–2512. DOI: 10.1016/j.cpc.2009.08.005 (cit. on p. 15).

[17] Andrea Zonca et al. “healpy: equal area pixelization and spherical harmonics transforms for data on the sphere in Python”. In: *Journal of Open Source Software* 4 (Mar. 2019), p. 1298. DOI: `10.21105/joss.01298` (cit. on p. 16).

[18] Krzysztof Gorski et al. “The HEALPix Primer”. In: (June 1999) (cit. on p. 16).

[19] Miljenko Lapaine. “Mollweide map projection”. In: *KoG* (Jan. 2011) (cit. on p. 16).

[20] Douglas Scott and George Smoot. “Cosmic Microwave Background Mini-Review”. In: (May 2010) (cit. on p. 17).

[21] François Chollet et al. *Keras*. `https://keras.io`. 2015 (cit. on p. 18).

[22] Karen Simonyan and Andrew Zisserman. “Very Deep Convolutional Networks for Large-Scale Image Recognition”. In: *International Conference on Learning Representations*. 2015 (cit. on p. 19).

[23] Olga Russakovsky et al. “ImageNet Large Scale Visual Recognition Challenge”. In: *International Journal of Computer Vision (IJCV)* 115.3 (2015), pp. 211–252. DOI: `10.1007/s11263-015-0816-y` (cit. on p. 19).

[24] Martın Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: `https://www.tensorflow.org/` (cit. on p. 19).

[25] INFN Padova. *Cloud Veneto: cloud infrastructure jointly developed by Padova University and INFN Padova and Legnaro*. `http://www.cloudveneto.it`. 2016 (cit. on p. 19).